Communication-Based Semantics for Recursive Session-Typed Processes

Ryan Kavanagh

CMU-CS-21-141
September 2021

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Stephen Brookes (Co-chair)
Frank Pfenning (Co-chair)
Jan Hoffmann
Luís Caires (Universidade Nova de Lisboa)
Gordon Plotkin (University of Edinburgh)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research was sponsored by National Science Foundation award CCF1017011; by Microsoft Corporation award 5005283; by a Carnegie Mellon University School of Computer Science Presidential Fellowship; and by a Natural Sciences and Engineering Research Council of Canada Postgraduate Doctoral Scholarship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

2020 *Mathematics Subject Classification.* Primary: 68Q55; Secondary: 03B70, 06B35, 18C50.

CCS Concepts. Theory of computation → *Program semantics*; Computing methodologies → *Concurrent programming languages*.

Key words and phrases. Session types, program equivalence, general recursion, observed communication semantics, testing equivalence, denotational semantics.

Abstract. Communicating systems are ubiquitous, and they bring human lives inestimable value. Despite this, they often go wrong, sometimes with severe consequences. They are hard to get right or reason about because of their inherent complexity. To tame this complexity, we can use various formalisms and semantic techniques to model, implement, and reason about communicating systems. Notable among these are session-typed programming languages and process calculi. Session types [Hon93; HVK98] are a typing discipline for communicating systems. They encode communication protocols to specify communications, analogously to how data types specify values in functional programs. Importantly, session-typed programming languages guarantee various desirable properties of communicating systems.

Many techniques exist for reasoning about session-typed programming languages and their programs. These include linear logical relations [Pér+14; Ton15], game semantics [CY19], denotational semantics [Atk17; KMP19], bisimulations [KPY17], and run-time monitoring [GJP18]. Few prior approaches have treated inductive and co-inductive session types [Ton15; LM16; DP19] or general recursive types [KPY17], or considered higher-order languages that integrate functional features and code transmission. Moreover, many prior techniques are not compositional.

In this dissertation, we present novel semantics and reasoning techniques for *Polarized SILL* [TCP13; PG15], a higher-order session-typed programming language. Polarized SILL coherently integrates functional programming with asynchronous session-typed message-passing concurrency. It supports recursive communication protocols, value transmission (including code transmission), choices (a form of branching), and synchronization. Our contributions are unified by their commitment to the process abstraction: communication is the only phenomenon of processes. As a result, our semantics define the meaning of processes in terms of their communications. Together, they support the following thesis:

> *Communication-based semantics elucidate the structure of session-typed languages and allow us to reason about programs written in these languages.*

Concretely, we give Polarized SILL three communication-based semantics: an *observed communications semantics*, a communication-based framework for *testing equivalences*, and a *denotational semantics*.

Our observed communication semantics defines the meaning of processes to be the communications we observe during their executions. Ours is the first to support rich protocols like recursion, code transmission, and synchronization.

We use our observed communication semantics to define extensional notions of program equivalence. They are given by a testing equivalences framework. Testing equivalence is a technique for defining process equivalence. It deems processes to be equivalent whenever they are indistinguishable through experimentation. Classical approaches to testing equivalences [DH84; Hen83; De 85] define experiment outcomes in terms of states. In contrast, we define experiment outcomes in terms of observed communications. We show that one of the testing equivalences captured by our framework coincides with barbed congruence, the canonical notion of process equivalence.

Our denotational semantics defines the meaning of communicating processes to be stable continuous functions between dI-domains of session-typed communications. Importantly, our denotational semantics is *compositional*, and we can reason modularly about programs. Our semantics is an instance of *CYO semantics*, a novel kind of semantics that adapts ideas from Kahn semantics for dataflow networks [Kah74] to handle bidirectional communications. Our denotational semantics is sound relative to barbed congruence.

To support our work, we make two contributions to the mathematical foundations of programming languages semantics. First, we introduce the first notions of fairness for substructural operational semantics and multiset rewriting systems, and we study their properties. These fairness results are essential to ensuring that our observed communications semantics is well-defined in the presence of non-terminating processes. Second, we introduce techniques for reasoning about parametrized fixed points of functors, and we study their 2-categorical properties. These results underlie our denotational interpretation of recursive session types.

Acknowledgments

I am grateful to have spent the past six years in the company of truly wonderful people at Carnegie Mellon and in Pittsburgh.

First, I would like to thank my advisors, Steve Brookes and Frank Pfenning, for their mentoring. You have taught me how to be a better researcher, how to be a better writer and speaker, and countless technical skills. Thank you for your unwavering support and encouragement these past six years. I would also like to thank my committee members, Gordon Plotkin, Jan Hoffmann, and Luís Caires, for having read my thesis, for their technical advice, and for supporting my professional development. More broadly, thank you to the Principles of Programming group for having taught me so much about programming languages. I enjoyed all of our PLunch talks, reading group discussions, and seminars. Thank you to Deb Cavlovich for your support throughout my PhD.

To my officemates in GHC 6207, Chris Fallin, Costin Badescu, and Anson Kahn: thank you for our endless office discussions, for the laughs, and for your friendship. Thank you to Catherine Copetas for your logistical support and for ensuring our social events ran smoothly. More importantly, thank you for your endless mischief and for having been an honourable opponent in our office feud. You definitely kept me on my toes!

Thank you to Andrew Carlisle for having made me a better piper, to the CMU Pipe Band for the camaraderie and for making great music together, and to the Pittsburgh Piping Society for the many evenings of musical fun.

Thank you to my family for having believed in me and for having encouraged me to pursue my dreams. I could not have done this without you.

Finally, thank you to my friends: you made Pittsburgh home for me. I will forever cherish the memories that we made together: the endless Avalon and brunch and board games; our Saturday night outings to IBC; our backyard barbecues, bonfires, and parties; the karaoke nights; skating, biking, kayaking, and hiking together; our many outings to the Pittsburgh Symphony Orchestra and the Pittsburgh Opera; the laughs at SIGBOVIK. Though we are dispersing around the globe, I eagerly await our future adventures.

Contents

List of Figures

CHAPTER 1

Introduction

Communicating systems are ubiquitous: we find them in everything from the cars we drive, to the smartphones in our pockets, to the computers on our desks. In all likelihood, this very document has been conveyed to you through a sequence of dozens of computer systems, each communicating with the next. Not only are communicating systems ubiquitous, but their importance to our lives cannot be understated. They allow us to communicate with loved ones across great distances and to access troves of information that would have unimaginable mere generations ago, and they support modern commerce. Despite their importance, communicating systems often go wrong, sometimes with severe consequences. For example, routing errors led to a large portion of the internet to be inaccessible for hours in the Northeastern United States on July 24, 2019 [Str19]. In 2015, the Heartbleed vulnerability allowed attackers to access private data on an estimated 24–55% of websites [Dur+14], and early estimates put its cost to industry at $500 million [Ker14]. It was due to an incorrect implementation of a communication protocol.

Communicating systems are hard to get right because of their inherent complexity. We can tame this complexity by abstracting away inessential details. The two fundamental abstractions of communicating systems are *processes* and *communication*. A process is a computational agent (broadly construed) that interacts with its environment solely through communication. In particular, the only phenomenon of processes is their communications: we cannot observe their internal states or workings. Communication is a sequence of atomic observable events ("messages"), each caused by a process and potentially observed by one or more other processes. A communicating system is then a collection of processes that interact through communication. To describe the inner workings of communicating systems, we need a third abstraction: *protocols*. A protocol is a specification of permitted communications.

To concretize these abstractions, consider the communicating system formed by a vending machine and an office worker. These two processes interact through communication: the office worker can insert a coin or push a button, and the vending machine can dispense a snack. Various protocols are possible. For example, we could allow the worker to arbitrarily insert coins or push buttons, but require that the vending machine dispense a snack whenever the office worker inserts a coin and then pushes a button. If we felt generous, we could instead insist that the vending machine dispense a snack whenever a button is pushed, regardless of whether a coin was inserted.

We can make these abstractions mathematically rigorous by appealing to various formalisms. *Process formalisms* describe the operational behaviour of processes. Examples include Milner's [Mil80] *Calculus of Communicating Systems*, Hoare's [Hoa85] *Communicating Sequential Processes*, and the π-calculi of Milner, Parrow, and Walker [MPW92a; MPW92b] and Sangiorgi [San92]. Similarly, various *protocol formalisms* have been used to specify protocols. These include state transition models [Boc78] and Petri nets [CAA84]. Though these two lines of research were long disconnected, they are united by session-typed programming languages and process calculi. *Session types* [Hon93; HVK98] specify protocols and classify communications, analogously to how data types classify values. Importantly, programs written in session-typed programming languages are guaranteed to respect the communication protocols specified by their session types.

Syntactic formalisms may describe and specify communicating systems, but they are not enough. For a formalism to be meaningful, it must be endowed with a semantics. We also need techniques to reason about formalized systems that are sound relative to their intended semantics.

In this dissertation, we study the semantics of session-typed programming languages and their associated reasoning techniques. In particular, we investigate semantics and reasoning techniques for the language *Polarized SILL* [TCP13; PG15].

Polarized SILL provides an ideal setting in which to study these topics: its feature set is sufficiently rich to study interactions between many desirable real-world features, while being sufficiently restricted to remain tractable. It coherently integrates functional programming with session-typed message-passing concurrency. Its functional programming layer is the simply-typed λ-calculus with a fixed-point operator, and it includes quoted processes as a base type. Its process layer is based on a proofs-as-processes correspondence between intuitionistic linear logic and the session-typed π-calculus. Protocols supported by Polarized SILL include value transmission (including quoted processes), choices (a form of branching), and synchronization (its communication layer is asynchronous). Importantly, it supports general recursive protocols. Recursive protocols are essential for modelling real-world systems, but in contrast to inductively and coinductively defined protocols, their semantics has been largely unstudied.

At the core of many techniques for reasoning about programs is the question of *program equivalence*. It asks: when are two programs in some sense "equivalent"? The answer to this question has many real-world applications, including compiler correctness and program verification. Indeed, "program equivalence is arguably one of the most interesting and at the same time important problems in formal verification" [Lah+18]. Program equivalence is an inherently semantic question: its answer depends on the semantics of our language. We answer it for Polarized SILL using two broad classes of semantic approaches, namely, operational and denotational semantics, and we relate the notions of equivalence that they induce.

Operational semantics describe the operational or run-time behaviour of programs. Polarized SILL's operational behaviour is specified by a substructural operational semantics [Sim12], a form of multiset rewriting with strong logical underpinnings [CS09]. Alone, this substructural operational semantics is insufficient for defining an extensional notion of equivalence, i.e., an equivalence where programs are equivalent if we cannot distinguish them through experimentation. To define such an equivalence, we must first define notions of observation and of experimentation. We take seriously the premise that we can only interact with processes through communication, and that communication is their sole phenomenon. This leads us to define an *observed communication semantics* [Atk17] for Polarized SILL, where the meaning of a process is the collection of communications we observe during its execution. To define experiments on processes, we adapt classical ideas on testing equivalences [DH84; Hen83; De 85] to the setting of observed communication semantics. We deem processes equivalent if we cannot differentiate them through communication. We relate testing equivalence to the canonical notion of process equivalence: barbed congruence [MS92].

Denotational semantics abstract away the concrete operational behaviour of programs and define the meaning of a program to be an object in a mathematical universe. This object is called the program's *denotation*, and programs are equivalent if they denote the same object. A defining characteristic of denotational semantics is that denotations are defined by induction on the program's structure. As a result, denotational semantics are automatically compositional. Advantageously, this means that we can reason modularly about programs instead of having to reason about whole programs at a time. We give Polarized SILL a denotational semantics where processes denote continuous functions between complete partial orders of communications. It is an instance of a novel style of semantics called *CYO semantics*.[1] By construction, it guarantees various desirable computational properties of processes. Subject to mild simplifying assumptions, our denotational equivalence implies barbed congruence and testing equivalence.

Our semantics are all designed to be faithful to the process abstraction: they are defined in terms of a process's communications. Together, they support the following thesis statement:

> *Communication-based semantics elucidate the structure of session-typed languages and allow us to reason about programs written in these languages.*

[1] In reference to the *Choose Your Own Adventure* book series.

To help further situate our communication-centric approach, we briefly contrast it with pre-existing approaches for reasoning about session-typed languages. Atkey [Atk17] gave a denotational semantics for Classical Processes [Wad15], a proofs-as-processes interpretation of classical linear logic with synchronous communication. Atkey's semantics coincides with the relational semantics of proofs in classical linear logic [Bar91]. In it, session types denote sets of communications and processes denote relations over these. It does not treat recursion or value transmission. In contrast, we draw inspiration from Kahn's work [Kah74] and use complete partial orders and continuous functions because they provide an ideal setting for investigating Polarized SILL's recursive protocols and processes. They also let us capture properties like continuity that we deem essential for a semantics for programs working with infinite data. Castellan and Yoshida [CY19] introduced a game semantics interpretation of the session π-calculus with recursion. Session types denote event structures that encode games, and processes denote continuous maps that describe strategies. Their semantics supports recursion and it is fully abstract relative to barbed congruence, i.e., it fully characterizes barbed congruence. However, it does not consider desirable language features like value transmission or functional programming. Kokke, Montesi, and Peressotti [KMP19] gave a denotational semantics using Brzozowski derivatives to a proofs-as-processes interpretation between classical linear logic and the π-calculus. It does not handle recursion or the transmission of functional values. Pérez et al. [Pér+12; Pér+14] gave a theory of logical relations for session-typed processes. In the broader setting of semantics for processes or communicating systems, Kahn [Kah74] gave a denotational semantics for dataflow networks. Communication in dataflow networks consists of unidirectional streams of values of a fixed simple type. In contrast, our semantics handle bidirectional communication and rich communication protocols. De Nicola and Hennessy [DH84], Hennessy [Hen83], and De Nicola [De 85] introduced testing equivalences, where programs are equivalent if they reach success states under all experiments. In our communication-based approach, programs are equivalent if they produce the same communications under all experiments.

1.1. Outline of Dissertation

Part 1 (chapters 2 to 4) is dedicated to the mathematical structures that underlie our work on Polarized SILL. It contains both an overview of the structures we use to study of Polarized SILL, as well as required contributions to the mathematical foundations of programming languages semantics.

Chapter 2 provides a brief survey of the mathematical concepts used in this dissertation. Its contents are mostly standard. Its primary purposes are to define our notation, and to present known results (or mild generalizations thereof) that will be used repeatedly in later chapters.

Chapter 3 contains the first study of fairness for multiset rewriting systems. As mentioned above, Polarized SILL's operational behaviour is defined using a multiset rewriting system. Rewrite rules in multiset rewriting systems can be applied non-deterministically: we may use a given rule whenever its conditions of use are satisfied. This non-determinism unfortunately means that a process in a communicating system might never make progress, even if it is able to do so. These "unfair" executions are undesirable in practice, and they make it difficult to give well-defined program equivalences. Accordingly, we restrict our attention to *fair* executions [Par80] of communicating systems. To do so, we introduce and study fairness for multiset rewriting systems. We discover that there are several reasonable and independent notions of fairness. We construct a fair scheduler, we give sufficient conditions for traces to be fair, and we study the effects of permutations on traces. We also introduce a novel notion of trace equivalence that will be essential to the observed communication semantics in chapter 6. This chapter builds on work presented at EXPRESS/SOS 2020 [Kav20a].

Chapter 4 studies fixed points of functors, and its results underlie our denotational account of recursive session types. In particular, we study the 2-categorical properties of fixed points of functors. We generalize existing techniques for computing and reasoning about parametrized fixed points of functors. In the process, we define a dagger operation [BÉ95; BÉ96] on a suitable category of categories, and we show that it satisfies the (cartesian) Conway identities. These identities imply

many other identities [BÉ96, § 3.3] useful for semantic reasoning, such as Bekič's rule, and they are also independent interest. This chapter builds on work presented at MFPS XXXVI [Kav20b].

We turn our attention to Polarized SILL in part 2 (chapters 5 to 9). It is here that we present our various semantics for Polarized SILL and their associated reasoning techniques. We start by presenting Polarized SILL, its statics, and its dynamics in chapter 5. The remaining chapters are our contributions. Each features an introductory section that gives a more detailed overview of the problems it seeks to solve and their associated challenges.

In chapter 6, we introduce our observed communication semantics. Given a process execution, it defines the meaning of the process to be the communications observed on its free channels. Our observed communication semantics is the first to handle recursive processes and protocols, and code transmission. We define a notion of equivalence on observed communications. To do so, we must address various challenges caused by code transmission. This communication equivalence is used in chapter 7 to define process equivalence. A key observation is that we observe the same communications across all fair executions of a given process. This faithfully captures the confluence property enjoyed by Polarized SILL and by other session-typed languages. This chapter builds on work presented at EXPRESS/SOS 2020 [Kav20a].

Chapter 7 introduces our testing equivalence framework. Testing equivalence frameworks [DH84; Hen83; De 85] use experiments to determine if processes are equivalent. Classical experiments can end in "success" states, and two processes are equivalent if they succeed the same experiments. In our setting, we cannot observe process states: we can only observe process communications. Our experiments communicate with tested processes, and we deem two processes to be equivalent if they produce equivalent communications under each experiment. There is a certain latitude in deciding which communication channels to observe during experimentation. One possibility is to observe the channels between an experiment and a process, leading to "internal" notions of equivalence à la Darondeau [Dar82] and Atkey [Atk17]. A second possibility is to imagine that an experiment uses some of its free channels to communicate with processes, and that it reports its findings on its remaining free channels. Observing these remaining free channels leads to an "external" notion of equivalence, and we show that this equivalence is a congruence. We also use our framework to define observational preorders and precongruences. We relate our observational equivalences to each other and to barbed congruence.

Our denotational semantics is given in chapter 8. It is inspired by "wave"-style geometry of interaction constructions [AJ94; Abr96, § 4.4] and the ideas underlying Kahn's [Kah74] semantics for dataflow networks, and it significantly generalizes the latter. Dataflow networks consist of processes communicating along unidirectional channels (lines or wires carrying messages in only one direction), and communications are sequences of values. In Kahn's semantics, processes denote continuous functions between complete partial orders of sequences of values. We generalize this to account for bidirectional communications and for the richer collection of protocols enjoyed by Polarized SILL. In particular, we introduce *CYO semantics*, a novel style of denotational semantics for systems with bidirectional communication. In CYO semantics, protocols denote partial orders of communications related by an embedding.[2] Concretely, they denote a complete partial order (cpo) of complete (bidirectional) communications permitted by the protocol and two cpos of unidirectional communications (one for each direction), and the embedding specifies the decomposition of complete communications into pairs of unidirectional communications. A process denotes a continuous function from the unidirectional input on its channels to the whole communications obtained by filling in the gaps in its input with its output. Subject to some simplifying assumptions, our denotational semantics is sound relative to barbed congruence and external testing equivalence, i.e., denotational equivalence implies barbed congruence and external testing equivalence.

Finally, in chapter 9, we illustrate our techniques with various case studies. In particular, we show that the forwarding process (the computational of the identity rule of intuitionistic linear logic) and the process interpretation of the identity expansion theorem for intuitionistic linear

[2]Embeddings are injective monotone functions with nice order-theoretic properties.

logic are equivalent. We also study processes on bit streams and binary representations of natural numbers. These results illustrate our semantics ability to reason about recursive processes and session types.

A reader who is primarily interested in operational notions of equivalence may prefer to take the following path through the dissertation: section 3.1.2 and chapters 5 to 7, while referring back to chapter 3 and sections 2.1.3, 2.4.2 and 2.5 as needed. A reader primarily interested in our denotational semantics could read chapters 5 and 8, while referring back to chapters 2 and 4 as needed. For convenience, all typing rules for Polarized SILL are collected in section 5.A, all multiset rewriting rules for Polarized SILL are collected in section 5.B, and all semantic clauses of the denotational semantics are collected in section 8.8. We have also included a glossary of symbols, and an index with major definitions and results.

Notational Conventions

When pattern matching against tuples, we often care only about certain components. To reduce the cognitive burden caused by unneeded names, we adopt a convention found in programming languages like Standard ML, Prolog, and Coq, where we notate irrelevant values as underscores "_". For example, instead of writing "the frobnication of n is $(n, p, q, 2 * n)$ for some p and q" when p and q are not used below, we write "the frobnication of n is $(n, \_, \_, 2 * n)$". Similarly, instead of writing $f(w, x, y, z) = x + y$, we may write $f(\_, x, y, \_) = x + y$.

When an expression is too long to fit on a single line, we typically break it at an operator or relation symbol, and we follow the Oxford custom [CBB54, pp. 37–38] of repeating this symbol on the following line.[3] For example, we write

$$f : \{\cdots\} \to \\ \to \{\cdots\}$$

when the domain and codomain of a function declaration cannot both fit onto a single line.

We often define terms inline to avoid breaking the flow of text. We use various typographical conventions to make definitions easier to locate definitions and to indicate their importance. The definitions of core concepts and objects are given in numbered environments. Important terms are given in **bold sans serif**, while less important terms appear in *emphasis*.

[3] This convention is also found in many ex-Soviet republics. It serves to connect two parts of expression, and it makes clear that the expression is incomplete.

Part 1

Mathematical Foundations

CHAPTER 2

Mathematical Preliminaries

For ease of reference, we define various mathematical concepts that will be used throughout this thesis.

Section 2.1 gives an account of various category-theoretic notions that underlie our denotational semantics. We use these notions to reason about our semantics itself in chapter 8, but also when using our semantics reason about processes in chapter 9. Virtually all concepts appearing in section 2.1 are standard. Our denotational semantics makes extensive use of order theory and fixed-point operators, which we survey in sections 2.2 and 2.3. Their contents are standard, apart from a few technical lemmas in section 2.3. These lemmas will be useful for reasoning about recursive processes.

In **??**, I give an account general binding trees. These extend abstract syntax trees and abstract binding trees [Chu40; Har16] to handle bound symbols. We use general binding trees to formally processes in part 2. In particular, we use their generalized binding structure to capture the fact that bound channel names appearing in processes can be freely renamed.

Finally, in section 2.5 we discuss inductively and coinductively defined judgments. In particular, we examine *parametric* judgments, which capture the structural properties of symbols appearing in judgments. Parametric judgments will be important part 2, where we use symbols to represent the names of communication channels appearing in process typing judgments.

2.1. Category Theory

There are a number of excellent introductory (and not so introductory) texts on category theory [AL91; BW99; Mac98; Rie16], of which Riehl's [Rie16] stands out. Our purpose here is not to give a primer on category theory, so much as to fix our definitions and notation. Especially in chapters 4 and 8 and section 2.2, we expect the reader to be familiar with the following notions: category, functor, natural transformation, product, and colimit.

We generally use upright boldface for categories. For example, we write **Set** for the category of sets and functions. We write $\mathbf{C}^{\mathrm{op}}$ for the opposite category of $\mathbf{C}$. We write $\mathsf{ob}(\mathbf{C})$ for the collection of objects in $\mathbf{C}$ and $\mathsf{mor}(\mathbf{C})$ for its collection of morphisms. A category is **small** if its objects and morphisms both form a set. It is **locally small** if its morphisms form a set. We write **Cat** for the category of small categories, and **CAT** for the category of locally small categories.

Remark 2.1.1. Many categories of interest are not small, so they are not objects in **Cat**. We can work around size issues by using a hierarchy of universes [Sch72, § 3] to treat them as small categories.

A subcategory $\mathbf{C}$ of $\mathbf{D}$ is **full** if for each pair of objects in $\mathbf{C}$, $\mathbf{C}$ contains all morphisms that are between them in $\mathbf{D}$. It is **wide** if it contains all of the objects in $\mathbf{D}$.

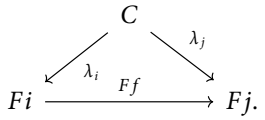
Definition 2.1.2. Let $F, G : \mathbf{C} \to \mathbf{D}$ be functors. A **natural transformation** $\eta : F \Rightarrow G : \mathbf{C} \to \mathbf{D}$, usually written $\eta : F \Rightarrow G$, is a family of morphisms $(\eta_X : FX \to GX)_X$ indexed by objects of $\mathbf{C}$ such that for all morphisms $f : X \to Y$ of $\mathbf{C}$, the following diagram commutes:

$$\begin{array}{ccc} FX & \xrightarrow{\eta_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow{\eta_Y} & GY \end{array}$$

◂

If $\mathbf{D}$ is a locally small category, then we write $\mathbf{D}(-,-) : \mathbf{D}^{\mathrm{op}} \times \mathbf{D} \to \mathbf{Set}$ for the **hom functor**. Given objects D and E of $\mathbf{D}$, $\mathbf{D}(D,E)$ is the set of morphisms from D to E. If a category $\mathbf{E}$ has an **internal hom**, then we write $\mathbf{E}[- \to -]$ or just $[- \to -]$ for it. Given small categories $\mathbf{C}$ and $\mathbf{D}$, write $\mathrm{diag}_{\mathbf{C}} : \mathbf{D} \to \mathbf{CAT}[\mathbf{C} \to \mathbf{D}]$ for the **diagonal functor**. Concretely, $\mathrm{diag}_{\mathbf{C}}\, D$ is the constant functor onto the object D.

Definition 2.1.3. A **diagram** of shape $\mathbf{J}$ in a category $\mathbf{C}$ is a functor $F : \mathbf{J} \to \mathbf{C}$. If C is an object of $\mathbf{C}$, then a **cone** on F with summit C is a natural transformation $\lambda : \mathrm{diag}_{\mathbf{J}}\, C \Rightarrow F$. Concretely, this is a $\mathrm{ob}(\mathbf{J})$-indexed family of morphisms $\lambda_i : C \to Fi$ such that for all $f : i \to j$ in $\mathbf{J}$, the following diagram commutes:



Dually, a **cocone** on F with nadir C is a natural transformation $\kappa : F \Rightarrow \mathrm{diag}_{\mathbf{J}}\, C$. ◂

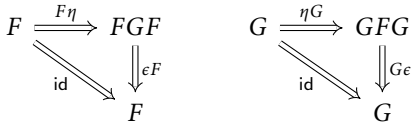
The **initial object** of a category $\mathbf{C}$, if it exists, is an object $\bot_{\mathbf{C}}$ such that for every object X of $\mathbf{C}$, there exists a unique morphism $\bot_{\mathbf{C}} \to X$ in $\mathbf{C}$. We often write $\bot$ for $\bot_{\mathbf{C}}$. We also write $\bot$ for the unique cone $\bot_{\mathbf{C}} \Rightarrow \mathrm{id}_{\mathbf{C}}$ witnessing the initiality of $\bot_{\mathbf{C}}$. The **terminal objects** $\top_{\mathbf{C}}$ of a category $\mathbf{C}$ is dually defined.

If $\mathbf{C}$ has a terminal object isomorphic to its initial object, we call the initial object the **zero object** $\mathrm{o}_{\mathbf{C}}$. $\mathbf{C}$ has **zero morphisms** if for all objects A and D there exists a fixed morphism $\mathrm{o}_{AD} : A \to D$, and if this family of morphisms satisfies $\mathrm{o}_{BD} \circ f = \mathrm{o}_{AD} = g \circ \mathrm{o}_{AC}$ for all morphisms $f : A \to B$ and $g : C \to D$. $\mathbf{C}$ has zero morphisms whenever it has a zero object: $\mathrm{o}_{AB} = A \to \mathrm{o} \to B$.

Definition 2.1.4. Given a functor $F : \mathbf{C} \to \mathbf{Set}$, the **category of elements** $\int F$ has as objects pairs (x, X) such that X is an object of $\mathbf{C}$ and $x \in FX$. Morphisms $f : (x, X) \to (y, Y)$ are morphisms $f : X \to Y$ in $\mathbf{C}$ such that $F(f)(x) = y$. ◂

Given a small category $\mathbf{J}$, a locally small category $\mathbf{C}$, and a functor $F : \mathbf{J} \to \mathbf{C}$, the **cocone functor** [Rie16, Definition 3.1.5] $\mathsf{Cone}(F, -) : \mathbf{C} \to \mathbf{Set}$ takes objects C of $\mathbf{C}$ to the set of cocones on F with summit C. Given a morphism $f : C \to C'$ and a cocone $(\lambda : F \Rightarrow C) \in \mathsf{Cone}(F, C)$, $\mathsf{Cone}(F, f)(\lambda) = f \circ \lambda$. Given a diagram $F : \mathbf{J} \to \mathbf{C}$, the **category of cocones on** F is the category of elements $\int \mathsf{Cone}(F, -)$. Its objects are pairs (α, A) where $\alpha \in \mathsf{Cone}(F, A)$. Morphisms $f : (\alpha, A) \to (\beta, B)$ are morphisms $f : C \to D$ in $\mathbf{C}$ such that $\mathsf{Cone}(F, f)(\alpha) = \beta$, i.e., such that $f \circ \alpha = \beta$. The **colimit** [Rie16, Definition 3.1.6] of F is the initial object of $\int \mathsf{Cone}(F, -)$. **Cone functors** $\mathsf{Cone}(-, F) : \mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$, the **category of cones on** F, and **limits** are dually defined.

An **adjunction** $F \dashv G$ is a pair of functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ equipped with an isomorphism $\mathbf{D}(F(C), D) \cong \mathbf{C}(C, G(D))$ natural in C and D. Equivalently, an adjunction is a pair of functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ equipped with natural transformations $\eta : \mathsf{id} \Rightarrow GF$ and $\epsilon : FG \Rightarrow \mathsf{id}$ satisfying the triangle indentities:



We call F the **left adjoint**, G the **right adjoint**, η the **unit**, and ϵ the **counit**.

A **two-variable adjunction** [Rie16, Definition 4.3.7] is given by a triple of functors $F : \mathbf{A} \times \mathbf{B} \to \mathbf{C}$, $G : \mathbf{A}^{\mathrm{op}} \times \mathbf{C}$, and $H : \mathbf{B}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{A}$ equipped with a natural isomorphism

$$\mathbf{C}(F(A,B), C) \cong \mathbf{B}(B, G(A,C)) \cong \mathbf{A}(A, H(B,C)).$$

We call G and H the **left** and **right closures** of F. We say that F is **closed** whenever G and H are naturally isomorphic.

A category is **discrete** if every morphism is an identity. A **product** is the limit of a diagram whose shape is a discrete category. We say that a product is finite if this discrete category has finitely many objects. A category is **cartesian** if it has all finite products. If $\mathbf{C}$ has binary products, then the **product bifunctor** $\times : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ assigns to pairs (A, B) of objects of $\mathbf{C}$ their product in $\mathbf{C}$. A cartesian category $\mathbf{C}$ is **cartesian closed** if its product bifunctor is closed. Its left and right closures define the **exponential objects** of $\mathbf{C}$.

It is often useful to have notation to identify the components of a product. Consider a discrete category with objects $d_1, \ldots, d_n$ and a diagram $F(d_i) = D_i$. We write $(d_1 : D_1) \times \cdots \times (d_n : D_n)$ or $\prod_{d_i} D_i$ for limit of F: it is the d_i-indexed product of the D_i. Given morphisms $f_i : C \to D_i$, we write $\langle d_1 : f_1, \ldots, d_n : f_n \rangle$ for the mediating morphism $C \to \prod_{d_i} D_i$. Given an indexed product $\prod_{i \in I} D_i$ and a subset $J \subseteq I$, we write π^I_J or π_J for the projection $\prod_{i \in I} D_i \to \prod_{j \in J} D_j$. If $\prod_{d_i} D_i$ is an object in a category of sets with structure and $\delta_i \in D_i$ for $1 \leq i \leq n$, then we write $(d_1 : \delta_1, \ldots, d_n : \delta_n)$ for the corresponding element of this product.

Coproducts are dually defined to products. We write $(d_1 : D_1) \oplus \cdots \oplus (d_n : D_n)$ or $\bigoplus_{d_i} D_i$ for the coproduct of the D_i indexed by the d_i. We write $\iota^{d_i} : D_i \to \bigoplus_{d_i} D_i$ for the injection of D_i into the coproduct.

Morphisms $\bigoplus_{i \in I} A_i \to \prod_{j \in J} B_j$ from coproducts to products are uniquely determined by morphisms $f_{(i,j)}$ for each $(i, j) \in I \times J$. This means that such morphisms can be conveniently represented by matrices whose (i, j)-th component is $f_{(i,j)}$ [Rie16, pp. 82f.]. When a category has zero morphisms and $I = J$, a collection of maps $f_i : A_i \to B_i$ for $i \in I$ determines a morphism $\mathrm{diag}(f_i)_{i \in I} : \bigoplus_{i \in I} A_i \to \prod_{i \in I} B_i$. It is represented by the matrix whose (i, i)-th component is f_i and whose (i, j)-th components for $i \neq j$ is the corresponding zero morphism.

Applications to semantics motivate functor algebras. Given a functor $F : \mathbf{C} \to \mathbf{C}$, an **$F$-algebra** is a pair (A, a) where A and a are respectively an object and a morphism $FA \to A$ in $\mathbf{C}$. A morphism $f : (A, a) \to (B, b)$ of F-algebras is a morphism $f : A \to B$ in $\mathbf{C}$ such that $f \circ a = b \circ Ff$. Such a morphism is called an **F-algebra homomorphism**. These objects and morphisms form a category $\mathbf{C}^F$ of F-algebras.

2.1.1. 2-Category Theory. Chapter 4 builds heavily on 2-category theory. Readers unfamiliar with 2-category theory may replace the words "2-category", "2-functor", and "2-natural transformation" by "category", "functor", and "natural transformation" throughout to obtain weaker forms of our results. Fiore [Fio94, Chapter 2] and Kelly and Street [KS74] give surveys of 2-category theory.

A **2-category** $\mathbf{C}$ has **objects** $A, B, \ldots$, **arrows** (horizontal morphisms) $f : A \to B$, and **2-cells** (vertical morphisms) $\alpha : f \Rightarrow g : A \to B$ drawn as:

$$A \;\underset{g}{\overset{f}{\Downarrow \alpha}}\; B$$

Objects and arrows form a category $\mathbf{C}_0$ called the **underlying category** of $\mathbf{C}$; we write $\circ$ for its composition. Each pair of objects A and B gives rise to a category $\mathbf{C}(A, B)$ whose objects are arrows $A \to B$ and whose morphisms are 2-cells between them; we call its composition operator "$\cdot$" **vertical composition**. Objects and 2-cells form a category $\mathbf{Cell}_\mathbf{C}$; we call its composition operator "$*$". Vertical and horizontal composition satisfy the **middle four interchange** and **identity** laws: whenever

$$A \;\underset{\Downarrow\beta}{\overset{\Downarrow\alpha}{\longrightarrow}}\; B \;\underset{\Downarrow\delta}{\overset{\Downarrow\gamma}{\longrightarrow}}\; C \qquad \text{and} \qquad A \;\underset{f}{\overset{f}{\Downarrow \mathsf{id}_f}}\; B \;\underset{g}{\overset{g}{\Downarrow \mathsf{id}_g}}\; C$$

we have $(\delta \cdot \gamma) * (\beta \cdot \alpha) = (\delta * \beta) \cdot (\gamma * \alpha)$ and $\mathsf{id}_g * \mathsf{id}_f = \mathsf{id}_{g \circ f}$, respectively. Thanks to the identity law, we can adopt the convention of writing f for the identity 2-cell $\mathsf{id}_f : f \Rightarrow f : A \to B$.

The prototypical 2-category is **Cat**, the category of small categories, where objects are small categories, horizontal morphisms are functors, and vertical morphisms are natural transformations. Given 2-cells $\epsilon : F \Rightarrow G : \mathbf{C} \to \mathbf{D}$ and $\eta : H \Rightarrow I : \mathbf{D} \to \mathbf{E}$ in **Cat**, their horizontal composition

$\eta * \epsilon : HF \Rightarrow IG$ is given by the equal natural transformations $I\epsilon \circ \eta F = \eta G \circ H\epsilon$. Given a morphism $f : K \to L$ in $\mathbf{C}$, we abuse notation and write $\eta * f : FK \to GL$ for the naturality square $FK \xrightarrow{Ff} FL \xrightarrow{\eta_L} GL = FK \xrightarrow{\eta_K} GK \xrightarrow{Gf} GK$.

Let $\mathbf{C}$ and $\mathbf{D}$ be 2-categories. A **2-functor** $F : \mathbf{C} \to \mathbf{D}$ sends objects of $\mathbf{C}$ to objects of $\mathbf{D}$, arrows of $\mathbf{C}$ to arrows of $\mathbf{D}$, and 2-cells of $\mathbf{C}$ to 2-cells of $\mathbf{D}$ while preserving all identities, compositions, domains, and codomains. A **2-natural transformation** $\eta : F \Rightarrow G : \mathbf{C} \to \mathbf{D}$ is a natural transformation $\eta : F \Rightarrow G$ that is 2-natural, i.e., such that for each 2-cell $\alpha : f \Rightarrow g : A \to B$ in $\mathbf{C}$, we have the following equality in $\mathbf{D}$:

$$FA \underbrace{\overset{Ff}{\Downarrow F\alpha}}_{Fg} FB \xrightarrow{\eta_B} GB = FA \xrightarrow{\eta_A} GA \underbrace{\overset{Gf}{\Downarrow G\alpha}}_{Gg} GB.$$

A **modification** $\rho : \alpha \to \beta : F \Rightarrow G : \mathbf{C} \to \mathbf{D}$ is a morphism of 2-natural transformations. It assigns to each object A of $\mathbf{C}$ a 2-cell $\rho_A : \alpha_A \Rightarrow \beta_A$ such that for all $f : A \to B$ we have the following equality in $\mathbf{D}$:

$$FA \underbrace{\overset{\alpha_A}{\Downarrow \rho_A}}_{\beta_A} GA \xrightarrow{Gf} GB = FA \xrightarrow{Ff} FB \underbrace{\overset{\alpha_B}{\Downarrow \rho_B}}_{\beta_B} GB.$$

Various constructions give new 2-categories from old. The **opposite** 2-category $\mathbf{C}^{\mathrm{op}}$ of a 2-category $\mathbf{C}$ is determined by $\mathbf{C}^{\mathrm{op}}(A, B) = \mathbf{C}(B, A)$, where arrows are reversed but not 2-cells between them. The **product 2-category** $\mathbf{C} \times \mathbf{D}$ is given by the usual product-category construction, where objects are pairs (C, D) of objects C of $\mathbf{C}$ and D of $\mathbf{D}$, and all morphisms, compositions, and identities are given component-wise.

Every 2-category $\mathbf{C}$ is equipped with a **hom 2-functor** $\mathbf{C}(-,-) : \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{CAT}$,[1] where $\mathbf{CAT}$ is the 2-category of locally small categories. It takes objects (A, B) to categories $\mathbf{C}(A, B)$, arrows $(f, g) : (A, B) \to (A', B')$ to functors $g \circ - \circ f : \mathbf{C}(A, B) \to \mathbf{C}(A', B')$, and 2-cells $(\alpha, \beta) : (f, g) \Rightarrow (f', g') : (A, B) \to (A', B')$ to natural transformations $\alpha * \mathsf{id}_- * \beta : g \circ - \circ f \Rightarrow g' \circ - \circ f' : \mathbf{C}(A, B) \to \mathbf{C}(A', B')$.

A 2-category $\mathbf{C}$ is **2-cartesian closed** if $\mathbf{Cell}_{\mathbf{C}}$ is cartesian closed [BÉ95, p. 97]. In elementary terms [Fio94, p. 24], this means that $\mathbf{C}$ has a terminal object, binary 2-products, and 2-exponentials, where

- the **terminal object** of $\mathbf{C}$ is an object 1 of $\mathbf{C}$ with a 2-natural isomorphism $\mathbf{C}(-, 1) \cong \Delta\mathbf{1}$, where $\mathbf{1}$ is the terminal category;
- the **2-product** of objects A and B of $\mathbf{C}$ is an object $A \times B$ of $\mathbf{C}$ with a 2-natural isomorphism $\mathbf{C}(-, A) \times \mathbf{C}(-, B) \cong \mathbf{C}(-, A \times B)$;
- the **2-exponential** of objects A and B of $\mathbf{C}$ is an object $\mathbf{C}[A \to B]$ of $\mathbf{C}$ with a 2-natural isomorphism $\mathbf{C}(- \times A, B) \cong \mathbf{C}(-, \mathbf{C}[A \to B])$.

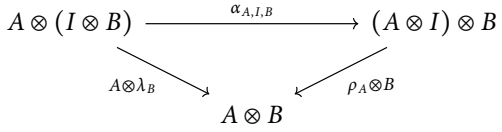
We can generalize functor algebras to algebras of horizontal morphisms in arbitrary 2-cartesian categories. Given a horizontal morphism $f : A \times B \to B$ in a 2-cartesian category, an **f-algebra** [BÉ95, Definition 2.3] is a pair (g, u) where $g : A \to B$ is a horizontal morphism and $u : f \circ \langle \mathsf{id}_A, g \rangle \Rightarrow g$ is vertical. An **f-algebra homomorphism** $(g, u) \to (h, v)$ is a vertical morphism $w : g \Rightarrow h$ such that $w \circ u = v \circ (f * \langle \mathsf{id}_A, w \rangle)$. These f-algebras and f-algebra homomorphisms form a category.

2.1.2. Monoidal Categories.

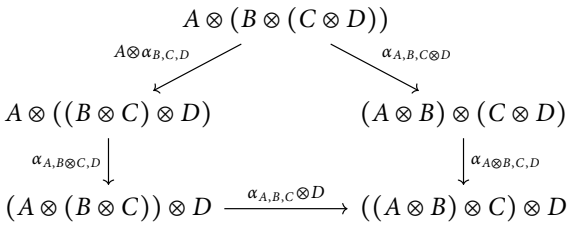
Definition 2.1.5. A **monoidal category** is a sextuple $(\mathbf{M}, \otimes, I, \lambda, \rho, \alpha)$ satisfying the axioms of fig. 2.1, where

- $\mathbf{M}$ is a category
- $\otimes : \mathbf{M} \times \mathbf{M} \to \mathbf{M}$ is a bifunctor (called a tensor) on $\mathbf{M}$

[1] It will be clear from context whether $\mathbf{C}(-,-)$ is the hom 2-functor into $\mathbf{CAT}$ or the usual hom functor into $\mathbf{Set}$.

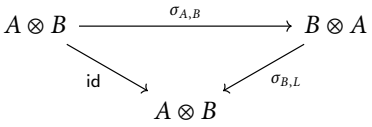


(A) The triangle axiom

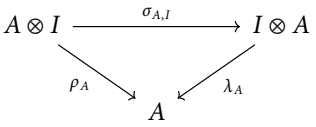


(B) The pentagon axiom

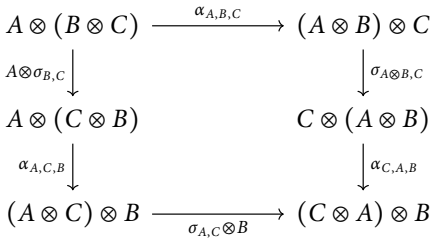
FIGURE 2.1. Axioms for monoidal categories



(A) Symmetry



(B) Preservation of identity



(C) Interaction with associativity

FIGURE 2.2. Axioms for symmetric monoidal categories

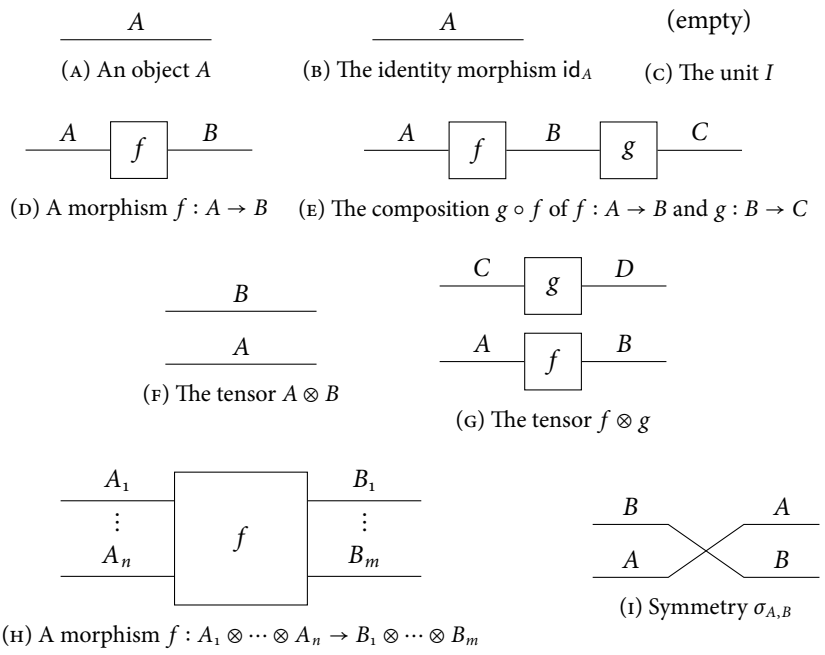
- I is the unit of the tensor
- $\lambda : I \otimes A \Rightarrow A$ is a natural isomorphism witnessing that I is the left unit
- $\rho : A \otimes I \Rightarrow A$ is a natural isomorphism witnessing that I is the right unit
- $\alpha : (A \otimes B) \otimes C \Rightarrow A \otimes (B \otimes C)$ is a natural isomorphism witnessing the associativity of the tensor $\otimes$.

A monoidal category is **symmetric** if it is additionally equipped with a natural isomorphism $\sigma : A \otimes B \Rightarrow B \otimes A$ satisfying the axioms of fig. 2.2. ◀

For more details on monoidal and symmetric monoidal categories, we refer the reader to the expositions by Etingof et al. [Eti+15, chap. 2], Barr and Wells [BW99, chap. 16], and Riehl [Rie16, § E.2].

Example 2.1.6. Every Cartesian category is symmetric monoidal. The tensor product is given by the Cartesian product, and the terminal object $\top$ is the unit. ◀

String diagrams provide a convenient graphical notation for reasoning about monoidal categories. Instead of reasoning about morphisms through algebraic manipulations, we can reason



(A) An object A (B) The identity morphism id_A (C) The unit I

(D) A morphism $f : A \to B$ (E) The composition $g \circ f$ of $f : A \to B$ and $g : B \to C$

(F) The tensor $A \otimes B$ (G) The tensor $f \otimes g$

(H) A morphism $f : A_1 \otimes \cdots \otimes A_n \to B_1 \otimes \cdots \otimes B_m$ (I) Symmetry $\sigma_{A,B}$

FIGURE 2.3. String diagram notation for (symmetric) monoidal categories

about them by manipulating diagrams made up of boxes and wires. We refer the reader to [Sel11; JSV96; JS91; Mal10, § 2.8] for more detailed expositions of this style of graphical language.

The graphical notation used herein is found in [Sel11; Mal10, § 2.8]. It depicts objects as wires, morphisms as boxes, identity morphisms as wires, and composition as connecting wires. The tensor operation is represented by juxtaposition, while the unit object I is the empty diagram. Symmetry is represented by crossing wires. We adopt the convention that diagrams flow from left to right: wires entering a box from the left denote inputs to a morphism, while wires exiting a box on the right denote outputs of a morphism. These are summarized by fig. 2.3.

THEOREM 2.1.7 ([Sel11, Theorem 3]). *A well-formed equation between morphism terms in the language of monoidal categories follows from the axioms of monoidal categories if and only if it holds, up to planar isotopy,*[2] *in the graphical language.*

THEOREM 2.1.8 ([Sel11, Theorem 7]). *A well-formed equation between morphisms in the language of symmetric monoidal categories follows from the axioms of symmetric monoidal categories if and only if it holds, up to isomorphism of diagrams,*[3] *in the graphical language.*

2.1.3. Multicategories, Polycategories, And Pluricategories. Multicategories generalize categories to allow for morphisms with multiple inputs. The following definition is for the original presentation of multicategory due to Lambek [Lam69, pp. 103ff.]. It differs from modern presentations, e.g., [Lei04], by allowing only two morphisms to be composed together, instead of requiring that all inputs receive a composition partner.

Given a set X, we write X^* for the free monoid on X, and ε for its unit. We write its elements as lists and composition as concatenation. We use capital Greek letters to range over these. If $\Phi = x_1, \cdots, x_n \in X^*$, then we write Φ_i for the i-th element x_i in the concatenation.

Definition 2.1.9. A **multicategory M** consists of the following data:

[2]Informally, planar isotopy means equivalence up to continuous deformation without allowing any boxes or wires to cross each other or be detached.

[3]An isomorphism of diagrams is a bijection between wires and boxes that preserves the structure of the graph.

- a class $\mathsf{ob}(\mathbf{M})$ of **objects**;
- a class $\mathsf{mor}(\mathbf{M})$ of **multimaps** or **morphisms**;
- a function $\mathrm{dom} : \mathsf{mor}(\mathbf{M}) \to \mathsf{ob}(\mathbf{M})^*$;
- a function $\mathrm{cod} : \mathsf{mor}(\mathbf{M}) \to \mathsf{ob}(\mathbf{M}))$;
- a **substitution** or **composition** operator $\mathrm{subst} : (\biguplus_{n\in\mathbb{N}} \mathsf{mor}(\mathbf{M}) \times_n \mathsf{mor}(\mathbf{M})) \to \mathsf{mor}(\mathbf{M})$ that is the unique morphism out of the disjoint union determined by a family of composition operators

$$\mathrm{subst}_n : \mathsf{mor}(\mathbf{M}) \times_n \mathsf{mor}(\mathbf{M}) \to \mathsf{mor}(\mathbf{M}),$$

where

$$\mathsf{mor}(\mathbf{M}) \times_n \mathsf{mor}(\mathbf{M}) = \{(g, f) \in \mathsf{mor}(\mathbf{M}) \times \mathsf{mor}(\mathbf{M}) \mid \mathrm{cod}(f) = (\mathrm{dom}(g))_n\}.$$

We write $f : A_1, \ldots, A_n \to B$ when $\mathrm{dom}(f) = A_1, \ldots, A_n$ and $\mathrm{cod}(f) = B$. We write $g \circ_i f$ for $\mathrm{subst}_i(g, f)$. It is convenient to write compositions in tree-form, with morphism names above the arrows:

$$\frac{\Psi \xrightarrow{f} A_i \quad A_1, \ldots, A_i, \ldots, A_n \xrightarrow{g} B}{A_1, \ldots, \Psi, \ldots, A_n \xrightarrow{g\circ_i f} B}$$

These data must satisfy the following axioms:

(1) if $f : \Psi \to A_i$ and $g : A_1, \ldots, A_n \to B$, then $g \circ_i f : A_1, \ldots, \Psi, \ldots, A_n \to B$;
(2) for all objects A, there exists an **identity morphism** $\mathsf{id}_A : A \to A$;
(3) the identity morphism is the left unit, i.e., the following multimap is equal to f:

$$\frac{A \xrightarrow{\mathsf{id}_A} A \quad \Phi, A, \Psi \xrightarrow{f} B}{\Phi, A, \Psi \longrightarrow B}$$

(4) the identity morphism is the right unit, i.e., the following multimap is equal to g:

$$\frac{\Lambda \xrightarrow{g} A \quad A \xrightarrow{\mathsf{id}_A} A}{\Lambda \xrightarrow{A}}$$

(5) composition is associative, i.e., the following multimaps are equal:

$$\frac{\Lambda \longrightarrow A \quad \Phi, A, \Psi \longrightarrow B}{\dfrac{\Phi, \Lambda, \Psi \longrightarrow B \qquad \Gamma, B, \Delta \longrightarrow C}{\Gamma, \Phi, \Lambda, \Psi, \Delta \longrightarrow C}}$$

$$\frac{\dfrac{\Phi, A, \Psi \longrightarrow B \quad \Gamma, B, \Delta \longrightarrow C}{\Lambda \longrightarrow A \qquad \Gamma, \Phi, A, \Psi, \Delta \longrightarrow C}}{\Gamma, \Phi, \Lambda, \Psi, \Delta \longrightarrow C}$$

(6) composition is partially commutative, i.e., the following multimaps are equal:

$$\frac{\Gamma \longrightarrow C \qquad \dfrac{\Delta \longrightarrow D \quad \Phi, C, \Theta, D, \Psi \longrightarrow B}{\Phi, C, \Theta, \Delta, \Psi \longrightarrow B}}{\Phi, \Gamma, \Theta, \Delta, \Psi \longrightarrow C}$$

$$\frac{\Delta \longrightarrow D \qquad \dfrac{\Gamma \longrightarrow C \quad \Phi, C, \Theta, D, \Psi \longrightarrow B}{\Phi, \Gamma, \Theta, D, \Psi \longrightarrow B}}{\Phi, \Gamma, \Theta, \Delta, \Psi \longrightarrow C}$$

◀

Polycategories [Sza75] generalize multicategories to allow for morphisms with multiple outputs.

Definition 2.1.10 ([Sza75])**.** A **polycategory P** is given by the following data:

- a class $\mathsf{ob}(\mathbf{M})$ of **objects**;
- a class $\mathsf{mor}(\mathbf{M})$ of **polymaps** or **morphisms**;
- functions $\mathrm{dom}, \mathrm{cod} : \mathsf{mor}(\mathbf{M}) \to \mathsf{ob}(\mathbf{M})^*$;

- a **substitution** or **composition** operator

$$\mathsf{subst} : \left(\biguplus_{n,m \in \mathbb{N}} \mathsf{mor}(\mathbf{M}) \times_{n,m} \mathsf{mor}(\mathbf{M}) \right) \to \mathsf{mor}(\mathbf{M})$$

that is the unique morphism out of the disjoint union determined by a family of composition operators

$$\mathsf{subst}_{n,m} : \mathsf{mor}(\mathbf{M}) \times_{n,m} \mathsf{mor}(\mathbf{M}) \to \mathsf{mor}(\mathbf{M}),$$

where

$$\mathsf{mor}(\mathbf{M}) \times_{n,m} \mathsf{mor}(\mathbf{M}) = \{(g, f) \in \mathsf{mor}(\mathbf{M}) \times \mathsf{mor}(\mathbf{M}) \mid (\mathrm{cod}(f))_m = (\mathrm{dom}(g))_n\}.$$

We write $f : A_1, \ldots, A_n \to B_1, \ldots, B_m$ when $\mathrm{dom}(f) = A_1, \ldots, A_n$ and $\mathrm{cod}(f) = B_1, \ldots, B_m$. We write $g \circ_{n,m} f$ for $\mathsf{subst}_{n,m}(g, f)$. It is convenient to write compositions in tree-form, with morphism names above the arrows:

$$\frac{\Psi \xrightarrow{f} \Lambda, A, \Phi \quad \Gamma, A, \Delta \xrightarrow{g} \Xi}{\Gamma, \Psi, \Delta \xrightarrow{g \circ_{n,m} f} \Lambda, \Xi, \Phi}$$

These data must satisfy the following axioms:

(1) if $f : \Psi \to \Lambda, A, \Psi$, $g : \Gamma, A, \Delta \to \Xi$, $(\mathrm{cod}(f))_n = A$, and $(\mathrm{dom}(g))_m = A$, then $g \circ_{n,m} f : \Gamma, \Psi, \Delta \to \Lambda, \Xi, \Phi$;
(2) for all objects A, there exists an **identity morphism** $\mathsf{id}_A : A \to A$;
(3) the identity morphism is the left unit, i.e., the following morphism is equal to f:

$$\frac{A \xrightarrow{\mathsf{id}_A} A \quad \Phi, A, \Psi \xrightarrow{f} \Xi}{\Phi, A, \Psi \longrightarrow \Xi}$$

(4) the identity morphism is the right unit, i.e., the following morphism is equal to g:

$$\frac{\Lambda \xrightarrow{g} \Gamma, A, \Delta \quad A \xrightarrow{\mathsf{id}_A} A}{\Lambda \xrightarrow{\Gamma}, A, \Delta}$$

(5) composition is associative, i.e., the following morphisms are equal:

$$\frac{\dfrac{\Gamma_1 \xrightarrow{f} \Gamma_2, A, \Gamma_3 \quad \Delta_1, A, \Delta_2 \xrightarrow{g} \Delta_3, B, \Delta_4}{\Delta_1, \Gamma_1, \Delta_2 \longrightarrow \Gamma_2, \Delta_3, B, \Delta_4, \Gamma_3} \quad \Phi_1, B, \Phi_2 \xrightarrow{h} \Phi_3}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Phi_3, \Delta_4, \Gamma_3}$$

$$\frac{\Gamma_1 \xrightarrow{f} \Gamma_2, A, \Gamma_3 \quad \dfrac{\Delta_1, A, \Delta_2 \xrightarrow{g} \Delta_3, B, \Delta_4 \quad \Phi_1, B, \Phi_2 \xrightarrow{h} \Phi_3}{\Phi_1, \Delta_1, A, \Delta_2, \Phi_2 \longrightarrow \Delta_3, \Phi_3, \Delta_4}}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Phi_3, \Delta_4, \Gamma_3}$$

(6) composition is partially commutative, i.e., the following morphisms are equal whenever at least one of Δ_2, Γ_2, and one of Δ_2, Γ_3 is empty:

$$\frac{\Delta_1 \xrightarrow{g} \Delta_2, B, \Delta_3 \quad \dfrac{\Gamma_1 \xrightarrow{f} \Gamma_2, A, \Gamma_3 \quad \Phi_1, A, \Phi_2, B, \Phi_3 \xrightarrow{h} \Phi_4}{\Phi_1, \Gamma_1, \Phi_2, B, \Phi_3 \longrightarrow \Gamma_2, \Phi_4, \Gamma_3}}{\Phi_1, \Gamma_1, \Phi_2, \Delta_1, \Phi_3 \longrightarrow \Delta_2, \Gamma_2, \Phi_4, \Gamma_3, \Delta_3}$$

$$\frac{\Gamma_1 \xrightarrow{f} \Gamma_2, A, \Gamma_3 \quad \dfrac{\Delta_1 \xrightarrow{g} \Delta_2, B, \Delta_3 \quad \Phi_1, A, \Phi_2, B, \Phi_3 \xrightarrow{h} \Phi_4}{\Phi_1, A, \Phi_2, \Delta_1, \Phi_3 \longrightarrow \Gamma_2, \Phi_4, \Gamma_3}}{\Phi_1, \Gamma_1, \Phi_2, \Delta_1, \Phi_3 \longrightarrow \Delta_2, \Gamma_2, \Phi_4, \Gamma_3, \Delta_3}$$

and the following morphisms are equal whenever at least one of Φ_1, Δ_1, and one of Φ_2, Δ_2, is empty:

$$\dfrac{\dfrac{\Gamma_1 \xrightarrow{h} \Gamma_2, A, \Gamma_3, B, \Gamma_4 \quad \Delta_1, A, \Delta_2 \xrightarrow{f} \Delta_3}{\Delta_1, \Gamma_1, \Delta_2 \longrightarrow \Gamma_2, \Delta_3, \Gamma_3, B, \Gamma_4} \qquad \Phi_1, B, \Phi_2 \xrightarrow{g} \Phi_3}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Gamma_3, \Phi_3, \Gamma_4}$$

$$\dfrac{\dfrac{\Gamma_1 \xrightarrow{h} \Gamma_2, A, \Gamma_3, B, \Gamma_4 \quad \Phi_1, B, \Phi_2 \xrightarrow{g} \Phi_3}{\Phi_1, \Gamma_1, \Phi_2 \longrightarrow \Gamma_2, A, \Gamma_3, \Phi_3, \Gamma_4} \qquad \Delta_1, A, \Delta_2 \xrightarrow{f} \Delta_3}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Gamma_3, \Phi_3, \Gamma_4}$$ ◂

Every polycategory contains a maximal multicategory.

Pluricategories[4] generalize polycategories to allow for composition along multiple objects. Its axioms associativity and partial commutativity axioms generalize those of polycategories in the obvious manner, where we replace single objects by a sequence of adjacent objects.

Definition 2.1.11. A **pluricategory P** is given by the following data:

- a class $\mathsf{ob}(\mathbf{M})$ of **objects**;
- a class $\mathsf{mor}(\mathbf{M})$ of **plurimaps** or **morphisms**;
- functions $\mathrm{dom}, \mathrm{cod} : \mathsf{mor}(\mathbf{M}) \to \mathsf{ob}(\mathbf{M})^*$;
- a **substitution** or **composition** operator

$$\mathrm{subst} : \left(\biguplus_{n,m,k \in \mathbb{N}} \mathsf{mor}(\mathbf{M}) \times_{n,m,k} \mathsf{mor}(\mathbf{M}) \right) \to \mathsf{mor}(\mathbf{M})$$

that is the unique morphism out of the disjoint union determined by a family of composition operators

$$\mathrm{subst}_{n,m,k} : \mathsf{mor}(\mathbf{M}) \times_{n,m,k} \mathsf{mor}(\mathbf{M}) \to \mathsf{mor}(\mathbf{M}),$$

where

$$\begin{aligned} &\mathsf{mor}(\mathbf{M}) \times_{n,m,k} \mathsf{mor}(\mathbf{M}) \\ &= \{(g, f) \in \mathsf{mor}(\mathbf{M}) \times \mathsf{mor}(\mathbf{M}) \mid \forall \mathsf{o} \leq i \leq k. (\mathrm{cod}(f))_{m+i} = (\mathrm{dom}(g))_{n+i}\}. \end{aligned}$$

We write $f : A_1, \ldots, A_n \to B_1, \ldots, B_m$ when $\mathrm{dom}(f) = A_1, \ldots, A_n$ and $\mathrm{cod}(f) = B_1, \ldots, B_m$. We write $g \circ_{n,m,k} f$ for $\mathrm{subst}_{n,m,k}(g, f)$. It is convenient to write compositions in tree-form, with morphism names above the arrows:

$$\dfrac{\Psi \xrightarrow{f} \Gamma, \Pi, \Phi \quad \Lambda, \Pi, \Delta \xrightarrow{g} \Xi}{\Lambda, \Psi, \Delta \xrightarrow{g \circ_{n,m,k} f} \Gamma, \Xi, \Phi}$$

These data must satisfy the following axioms:

(1) if $f : A_1, \ldots, A_m \to C_1, \ldots, C_n$, $g : B_1, \ldots, B_p \to D_1, \ldots, D_q$, then

$$\begin{aligned} \mathrm{dom}(g \circ_{r,s,t} f) &= B_1, \ldots, B_{r-1}, A_1, \ldots, A_m, B_{r+t+1}, \ldots, B_p \\ \mathrm{cod}(g \circ_{r,s,t} f) &= C_1, \ldots, C_{s-1}, D_1, \ldots, D_q, C_{s+t+1}, \ldots, C_q; \end{aligned}$$

(2) for all lists of objects objects Δ, there exists an **identity morphism** $\mathsf{id}_\Delta : \Delta \to \Delta$;

(3) identity morphisms commute with concatenation, i.e., the following composition is equal to $\mathsf{id}_{\Delta,\Psi}$:

$$\dfrac{\Delta \xrightarrow{\mathsf{id}_\Delta} \Delta \quad \Psi \xrightarrow{\mathsf{id}_\Psi} \Psi}{\Delta, \Psi \longrightarrow \Delta, \Psi}$$

[4]Introduced here as a convenient notation for representing compositions in symmetric monoidal categories. In particular, they simplify semantic reasoning by omitting the need to include extraneous identity morphisms when composing pairs of morphisms.

(4) the identity morphism is the left unit, i.e., the following morphism is equal to f:

$$\frac{\Delta \xrightarrow{\mathsf{id}_\Delta} \Delta \quad \Phi, \Delta, \Psi \xrightarrow{f} \Xi}{\Phi, \Delta, \Psi \longrightarrow \Xi}$$

(5) the identity morphism is the right unit, i.e., the following morphism is equal to g:

$$\frac{\Lambda \xrightarrow{g} \Gamma, \Delta, \Delta \quad \Delta \xrightarrow{\mathsf{id}_\Delta} \Delta}{\Lambda \xrightarrow{\Gamma}, \Delta, \Delta}$$

(6) composition is associative, i.e., the following morphisms are equal:

$$\frac{\dfrac{\Gamma_1 \xrightarrow{f} \Gamma_2, \Pi, \Gamma_3 \quad \Delta_1, \Pi, \Delta_2 \xrightarrow{g} \Delta_3, \Theta, \Delta_4}{\Delta_1, \Gamma_1, \Delta_2 \longrightarrow \Gamma_2, \Delta_3, \Theta, \Delta_4, \Gamma_3} \quad \Phi_1, \Theta, \Phi_2 \xrightarrow{h} \Phi_3}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Phi_3, \Delta_4, \Gamma_3}$$

$$\frac{\Gamma_1 \xrightarrow{f} \Gamma_2, \Pi, \Gamma_3 \quad \dfrac{\Delta_1, \Pi, \Delta_2 \xrightarrow{g} \Delta_3, \Theta, \Delta_4 \quad \Phi_1, \Theta, \Phi_2 \xrightarrow{h} \Phi_3}{\Phi_1, \Delta_1, \Pi, \Delta_2, \Phi_2 \longrightarrow \Delta_3, \Phi_3, \Delta_4}}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Phi_3, \Delta_4, \Gamma_3}$$

(7) composition is partially commutative, i.e., the following morphisms are equal whenever at least one of Δ_2, Γ_2, and one of Δ_2, Γ_3 is empty:

$$\frac{\Delta_1 \xrightarrow{g} \Delta_2, \Theta, \Delta_3 \quad \dfrac{\Gamma_1 \xrightarrow{f} \Gamma_2, \Pi, \Gamma_3 \quad \Phi_1, \Pi, \Phi_2, \Theta, \Phi_3 \xrightarrow{h} \Phi_4}{\Phi_1, \Gamma_1, \Phi_2, \Theta, \Phi_3 \longrightarrow \Gamma_2, \Phi_4, \Gamma_3}}{\Phi_1, \Gamma_1, \Phi_2, \Delta_1, \Phi_3 \longrightarrow \Delta_2, \Gamma_2, \Phi_4, \Gamma_3, \Delta_3}$$

$$\frac{\Gamma_1 \xrightarrow{f} \Gamma_2, \Pi, \Gamma_3 \quad \dfrac{\Delta_1 \xrightarrow{g} \Delta_2, \Theta, \Delta_3 \quad \Phi_1, \Pi, \Phi_2, \Theta, \Phi_3 \xrightarrow{h} \Phi_4}{\Phi_1, \Pi, \Phi_2, \Delta_1, \Phi_3 \longrightarrow \Gamma_2, \Phi_4, \Gamma_3}}{\Phi_1, \Gamma_1, \Phi_2, \Delta_1, \Phi_3 \longrightarrow \Delta_2, \Gamma_2, \Phi_4, \Gamma_3, \Delta_3}$$

and the following morphisms are equal whenever at least one of Φ_1, Δ_1, and one of Φ_2, Δ_2, is empty:

$$\frac{\dfrac{\Gamma_1 \xrightarrow{h} \Gamma_2, \Pi, \Gamma_3, \Theta, \Gamma_4 \quad \Delta_1, \Pi, \Delta_2 \xrightarrow{f} \Delta_3}{\Delta_1, \Gamma_1, \Delta_2 \longrightarrow \Gamma_2, \Delta_3, \Gamma_3, \Theta, \Gamma_4} \quad \Phi_1, \Theta, \Phi_2 \xrightarrow{g} \Phi_3}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Gamma_3, \Phi_3, \Gamma_4}$$

$$\frac{\dfrac{\Gamma_1 \xrightarrow{h} \Gamma_2, \Pi, \Gamma_3, \Theta, \Gamma_4 \quad \Phi_1, \Theta, \Phi_2 \xrightarrow{g} \Phi_3}{\Phi_1, \Gamma_1, \Phi_2 \longrightarrow \Gamma_2, \Pi, \Gamma_3, \Phi_3, \Gamma_4} \quad \Delta_1, \Pi, \Delta_2 \xrightarrow{f} \Delta_3}{\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2 \longrightarrow \Gamma_2, \Delta_3, \Gamma_3, \Phi_3, \Gamma_4}$$

◀

We can extract a maximal polycategory **P** from each pluricategory **Q**. The objects of **P** are those of **Q**, and the morphisms of **P** are those whose codomain consists of a single object.

Example 2.1.12. Every symmetric monoidal category induces a pluricategory by interpreting morphisms $f : A_1 \otimes \cdots \otimes A_n \to B_1 \otimes \cdots \otimes B_m$ as morphisms $f : A_1, \ldots, A_n \to B_1, \ldots, B_m$. ◀

2.2. Order Theory

Partial orders have long been applied to the semantics of programming languages. We briefly review the key definitions that we will use in this thesis, building primarily on [AJ95; Gie+03]. For a deeper introduction, we refer the reader to the wealth of expository works on order theory and domain theory, and on applications of domain theory to the semantics of programming languages: [AJ95; Gie+80; Gie+03; GM89; Gun92; Ten95].

Definition 2.2.1. A **partially ordered set** or **poset** $(P, \sqsubseteq)$ is a set P equipped with a relation $\sqsubseteq$ that is:

(1) **reflexive**: for all $x \in P$, $p \sqsubseteq p$;
(2) **transitive**: for all $x, y, z \in P$, $x \sqsubseteq y$ and $y \sqsubseteq z$ implies $x \sqsubseteq z$;
(3) **antisymmetric**: for all $x, y, z \in P$, $x \sqsubseteq y$ and $y \sqsubseteq x$ implies $x = y$. ◀

As usual in mathematics, we usually leave the structure on a set implicit, i.e., we write P for the poset $(P, \sqsubseteq)$.

Definition 2.2.2. A function $f : P \to Q$ between posets is **monotone** if for all $x, y \in A$, if $x \sqsubseteq y$, then $f(x) \sqsubseteq f(y)$. ◀

Partially ordered sets and monotone functions between them form a category **Poset**.

Example 2.2.3. Let P and Q be posets. The poset $\mathbf{Poset}[P \to Q]$ has as elements monotone functions $f : P \to Q$. Its ordering is given pointwise, i.e., $f \sqsubseteq g$ if and only if $f(x) \sqsubseteq g(x)$ for all $x \in P$. ◀

Definition 2.2.4 ([AJ95, Definition 2.1.3])**.** Let P be a poset and A a subset of P.

(1) The subset A is an **upper set** if $x \in A$ implies $y \in A$ for all $y \sqsupseteq x$. We write $\uparrow A$ for the least upper set containing A, and $\uparrow x$ for $\uparrow\{x\}$.
(2) An element $x \in P$ is an **upper bound** of A if $a \sqsubseteq x$ for all $a \in A$.
(3) The subset A is a **lower set** if $x \in A$ implies $y \in A$ for all $y \sqsubseteq x$. We write $\downarrow A$ for the least upper set containing A, and $\downarrow x$ for $\downarrow\{x\}$.
(4) An element $x \in P$ is a **lower bound** of A if $x \sqsubseteq a$ for all $a \in A$.
(5) The least upper bound of A, if it exists, is variously called the **lub**, **supremum**, or **join** of A. We write $\bigsqcup A$ for this element when it exists.
(6) The least element of P, if it exists, is called its **bottom element** and is denoted by $\bot$. In this case, we say that P is a **pointed poset**.
(7) The greatest lower bound of A, if it exists, is variously called the **glb**, **infimum**, or **meet** of A. We write $\bigsqcap A$ for this element when it exists.
(8) The greatest element of P, if it exists, is called its **top element** and is denoted by $\top$.
(9) If every pair of elements in P has a supremum and an infimum, then P is called a **lattice**. A **complete lattice** is a lattice with a supremum and an infimum for each of its subsets. ◀

Definition 2.2.5. A function $F : P \to Q$ of pointed posets is **strict** if $f(\bot) = \bot$. ◀

Given a category $\mathbf{P}$ of partially ordered sets, we write $\mathbf{P}_\bot$ for the full subcategory of $\mathbf{P}$ whose objects are pointed posets. We write $\mathbf{P}_{\bot!}$ for the wide subcategory of $\mathbf{P}_\bot$ whose morphisms are strict.

Definition 2.2.6. Let P be a poset and $f : P \to P$ a function. An element $x \in P$ is called a **fixed point** of f if $f(x) = x$, a **pre-fixed point** of f if $f(x) \sqsubseteq x$, and a **post-fixed point** of f if $x \sqsubseteq f(x)$. The least and greatest fixed points of f, if they exist, are respectively denoted $\mathrm{lfp}(f)$ and $\mathrm{gfp}(f)$. ◀

The fixed points of a monotone function on a complete lattices form a complete lattice. In particular, monotone functions on complete lattices enjoy least and greatest fixed points:

THEOREM 2.2.7 (Knaster-Tarski [Tar55]). *Let L be a complete lattice, $f : L \to L$ be a monotone function, and P the set of all fixed points of f. Then P is non-empty, a complete lattice, and*

$$\bigsqcap P = \bigsqcap\{x \mid f(x) \sqsubseteq x\},$$
$$\bigsqcup P = \bigsqcup\{x \mid x \sqsubseteq f(x)\}.$$

We can explicitly construct fixed points of ω-(co)continuous functions on complete lattices using the Kleene fixed-point theorem:

Definition 2.2.8. Let L be a complete lattice. We say that a function $f : L \to L$ is

(1) **ω-continuous** if $f(\bigsqcup_{i \in \mathbb{N}} x_i) = \bigsqcup_{i \in \mathbb{N}} f(x_i)$ for all increasing sequences $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ of points in L;

(2) ω-**cocontinuous** if $f(\bigsqcap_{i\in\mathbb{N}} x_i) = \bigsqcap_{i\in\mathbb{N}} f(x_i)$ for all decreasing sequences $\cdots \sqsubseteq x_1 \sqsubseteq x_0$ of points in L. ◀

THEOREM 2.2.9 (Kleene Fixed-Point [San12, Theorem 2.8.5]). *Let L be a complete lattice and $f : L \to L$ a function. If f is ω-continuous, then*

$$\operatorname{lfp}(f) = \bigsqcup_{n\in\mathbb{N}} f^n(\bot).$$

If f is ω-cocontinuous, then

$$\operatorname{gfp}(f) = \bigsqcap_{n\in\mathbb{N}} f^n(\top).$$

In chapter 4, we will generalize the above results from ω-cocontinuous functions on lattices to ω-cocontinuous functors on categories.

The definition of lattice is too strong for many applications, and functions on sets with less structure still have pleasant fixed-point properties.

Definition 2.2.10 ([AJ95, Definition 2.1.8])**.** A subset A of a poset P is **directed** if it is non-empty and each pair of elements of A has an upper bound in A. We write $\bigsqcup^{\uparrow} A$ for the supremum of a directed subset A, if it exists, and call it a **directed supremum**. ◀

The following proposition is useful for simplifying calculations involving directed suprema:

PROPOSITION 2.2.11 ([AJ95, Proposition 2.12.2]). *Let I be a directed poset and let $\alpha : I \times I \to P$ be a monotone function into a poset P. If the following directed suprema exist, then they are equal:*

$$\bigsqcup^{\uparrow}_{i,j\in I} \alpha(i,j) = \bigsqcup^{\uparrow}_{i\in I} \bigsqcup^{\uparrow}_{j\in I} \alpha(i,j) = \bigsqcup^{\uparrow}_{j\in I} \bigsqcup^{\uparrow}_{i\in I} \alpha(i,j) = \bigsqcup^{\uparrow}_{i\in I} \alpha(i,i).$$

Definition 2.2.12. A **directed-complete partial order** or **dcpo** is a poset whose every directed subset has a supremum. ◀

Remark 2.2.13. Some authors [Mit90, p. 394] require that dcpos be pointed, i.e., that they have a bottom element. Following Abramsky and Jung [AJ95], we do not require dcpos to be pointed.

Definition 2.2.14. Let C and D be dcpos. A function $f : C \to D$ is **(Scott-)continuous** if for all directed subsets A of C, $f\left(\bigsqcup^{\uparrow} A\right) = \bigsqcup f(A)$. ◀

PROPOSITION 2.2.15 ([AJ95, Exercise 2.3.9(12)]). *A function $f : C \to D$ between dcpos is continuous if and only if it is monotone and $f\left(\bigsqcup^{\uparrow} A\right) = \bigsqcup^{\uparrow} f(A)$ for all directed subsets A of C.*

Dcpos and continuous functions between them form a category **DCPO**.

One important feature of continuous functions on pointed dcpos is that they admit fixed points. The following proposition gives two explicit characterizations of these:

PROPOSITION 2.2.16. *Let D be a pointed dcpo, and let $f : D \to D$ be a continuous function. Then the fixed points of f form a pointed dcpo. In particular, f has a least fixed point* $\operatorname{lfp}(f) \in D$*, and it is equivalently constructed:*

(1) using the Kleene fixed-point theorem, with $\operatorname{lfp}(f) = \bigsqcup^{\uparrow}_{n\in\mathbb{N}} f^n(\bot_D)$*;*
(2) using a variant of the Knaster-Tarski theorem, with $\operatorname{lfp}(f) = \bigsqcap\{x \in D \mid f(x) \sqsubseteq x\}$.

Proof. The proofs are standard. We begin by showing the formulation given by the Kleene fixed-point theorem. Observe first that $\bot \sqsubseteq f(\bot_D)$, and induction on n shows that $f^n(\bot_D) \sqsubseteq f^{n+1}(\bot_D)$ for all n. It follows that $\{f^n(\bot_D) \mid n \in \mathbb{N}\}$ is a directed subset of D, so it has a directed supremum by definition of dcpo. Observe that this directed supremum is a fixed point of f:

$$f\left(\bigsqcup^{\uparrow}_{n\in\mathbb{N}} f^n(\bot_D)\right) = \bigsqcup^{\uparrow}_{n\in\mathbb{N}} f^{n+1}(\bot_D) = \bigsqcup^{\uparrow}_{n\in\mathbb{N}} f^n(\bot_D).$$

To see that it is the least fixed point, let d be any other fixed point of f. Then $\bot_D \sqsubseteq d = f(d)$, and by induction on n, we have $f^n(\bot_D) \sqsubseteq d$ for all n. So d is also an upper bound of $\{f^n(\bot_D) \mid n \in \mathbb{N}\}$. It follows that $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} f^n(\bot_D) \sqsubseteq d$, i.e., that $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} f^n(\bot_D)$ is truly the least upper bound of f.

Next, we show the Knaster-Tarski formulation. Set $F = \{x \in D \mid f(x) \sqsubseteq x\}$. Observe that $\mathrm{lfp}(f) \in F$, so it is sufficient to show that $\mathrm{lfp}(f)$ is the least element of F. Let $x \in F$ be arbitrary. Then $\bot \sqsubseteq x$. By monotonicity of f and the definition of F, $f(\bot) \sqsubseteq f(x) \sqsubseteq x$. By induction on n, we get $f^n(\bot_D) \sqsubseteq x$ for all n, so $\mathrm{lfp}(f) \sqsubseteq x$. It follows that $\mathrm{lfp}(f)$ is the least element of f, so the infimum $\bigsqcap F$ exists and is equal to $\mathrm{lfp}(f)$.

We know by the existence of $\mathrm{lfp}(f)$ that the fixed points of f form a pointed poset. It remains to show that this pointed poset is directed complete. Let X be any directed subset. We must show that $f(\bigsqcup^{\uparrow} X) = \bigsqcup^{\uparrow} X$. But this is immediate by continuity of f, and the fact that the elements of X are fixed points of f:

$$f\left(\bigsqcup^{\uparrow} X\right) = \bigsqcup^{\uparrow} f(X) = \bigsqcup^{\uparrow} X. \qquad \square$$

A key idea underlying category theory is that objects are best studied through the lens of their morphisms. Order theory enjoys a wide variety of morphisms with special properties, and we consider these here.

Definition 2.2.17. Let P and Q be posets. We say that monotone functions $l : P \leftrightarrows Q : u$ form an **adjunction** (l, u) if for all $x \in P$ and $y \in Q$, $l(x) \sqsubseteq y$ if and only if $x \sqsubseteq u(y)$. In this case, we write $l \dashv u$, and we call l the **lower adjoint** and u the **upper adjoint**. ◂

Remark 2.2.18. The order of l and u in an adjunction (l, u) varies in the literature. We follow [AJ95], who place the lower adjoint on the left, while [Gie+03] write (u, l) for the same adjunction. We generally prefer the categorical notation $l \dashv u$ to eliminate any ambiguity.

In the literature, adjunctions are also called *Galois connections*. When the posets P and Q are viewed as categories, we recognize the adjoint functions $l \dashv u$ as adjoint functors. Consequently, facts about adjoint functors, e.g., that adjoints uniquely determine each other, also apply to adjoint functions between posets. The following equivalent definitions of adjunctions are well known:

Proposition 2.2.19 ([Gie+03, Theorem O-3.6; AJ95, Propositions 3.1.10 and 3.1.12]). *Let P and Q be posets, and assume that $l : P \leftrightarrows Q : u$ are monotone. The following are equivalent:*

(1) $l \dashv u$ is an adjunction;
(2) $l \circ u \sqsubseteq \mathsf{id}_Q$ and $\mathsf{id}_P \sqsubseteq u \circ l$;
(3) $\forall x \in P,\ l(x) = \min(u^{-1}(\uparrow x))$;
(4) $\forall y \in Q,\ u(y) = \max(l^{-1}(\downarrow y))$.

These conditions imply:

(5) $l = l \circ u \circ l$ and $u = u \circ l \circ u$;
(6) $l \circ u$ and $u \circ l$ are idempotent;
(7) l is injective if and only if $u \circ l = \mathsf{id}_P$ if and only if u is surjective;
(8) l is surjective if and only if $l \circ u = \mathsf{id}_Q$ if and only if u is injective;
(9) l preserves existing suprema, and u preserves existing infima.

The following class of adjunctions is particularly useful in applications to semantics:

Definition 2.2.20. Let P and Q be posets. Monotone functions $e : P \leftrightarrows Q : p$ form an **embedding-projection pair** or **e-p-pair** (e, p) if $p \circ e = \mathsf{id}$ and $e \circ p \sqsubseteq \mathsf{id}$. We say e is an **embedding** and p is a **projection**. As adjoints, the functions e and p uniquely determine each other. Given an e-p-pair (f, g), we may write f^p for g and g^e for f. ◂

We will frequently silently use the following proposition:

Proposition 2.2.21. *Lower adjoints (including embeddings) are strict whenever their domains are pointed. Surjective upper adjoints (including projections) are also strict whenever their domains are pointed.*

Proof. If l is a lower adjoint, then by proposition 2.2.19,

$$l(\bot) = l(\bigsqcup \varnothing) = \bigsqcup \varnothing = \bot.$$

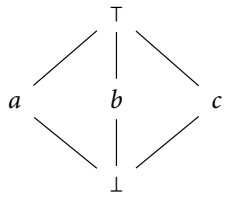


FIGURE 2.4. Elements $a, b, c, \bot$ are compact but not prime

If u is a surjective upper adjoint, then its lower adjoint l is injective by proposition 2.2.19. By definition of adjunction, $(l \circ u)(\bot) \sqsubseteq \bot$. This implies $(l \circ u)(\bot) = \bot$. But $l(\bot) = \bot$ and l is injective, so $u(\bot) = \bot$. □

The ordering in dcpos provides a notion of convergence. We are also interested in a second ordering on dcpos, which specifies which elements can be used to "approximate" other elements.

Definition 2.2.22. We say that x **approximates** y (or that x is **way-below** y) in a dcpo D if for all directed subsets A of D, $y \sqsubseteq \bigsqcup^{\uparrow} A$ implies $x \sqsubseteq a$ for some $a \in A$. In this case, we write $x \ll y$. We say that x is **compact** if $x \ll x$. Write $\mathcal{K}(D) = \{x \in D \mid x \ll x\}$ for the set of compact elements of D. ◂

Remark 2.2.23. The terminology "approximation order" is due Abramsky and Jung [AJ95]. It is traditionally called the "way-below relation".

Remark 2.2.24. Compact elements are often called "finite" elements. This is because the compact elements in the complete lattice induced by a power set are exactly its finite sets.

The approximation and convergence orders are related as follows:

PROPOSITION 2.2.25 ([AJ95, Proposition 2.2.2]). *Let D be a dcpo. Then for all $x, x', y, y' \in D$,*

(1) if $x \ll y$, then $x \sqsubseteq y$;
(2) if $x' \sqsubseteq x \ll y \sqsubseteq y'$, then $x' \ll y'$.

Definition 2.2.26. A **basis** B of a dcpo D is a subset $B \subseteq D$ such that for all $x \in D$, the set $\{b \in B \mid b \ll x\}$ contains a directed subset with supremum x. ◂

A domain is a dcpo equipped with a notion of approximation:

Definition 2.2.27. A dcpo D is an **algebraic domain** if it has a basis of compact elements. It is an **ω-algebraic domain** if $\mathcal{K}(D)$ is a countable basis of D. ◂

Prime elements are a special subclass of compact elements:

Definition 2.2.28. Let D be a dcpo. An element p is **prime** if for all bounded subsets B of D, $p \sqsubseteq \bigsqcup B$ implies $p \sqsubseteq b$ for some $b \in B$. Write $\lfloor D \rfloor$ for the set of prime elements of D. We say that D is **prime-algebraic** if every element of D is the supremum of its prime elements. ◂

Though every prime element is compact, not every compact element is prime. For example, the bottom element is never prime in non-trivial dcpos. More generally, every element in the Hasse diagram of fig. 2.4 is compact, but only the top element is prime.

There are several important classes of domains in semantics.

Definition 2.2.29 ([Gun92, p. 151])**.** A non-empty dcpo D is **bounded-complete** if every bounded subset $M \subseteq D$ has a least upper bound $\bigsqcup M \in D$. We write **BC** for the category of bounded-complete dcpos and continuous morphisms between them. ◂

We say that a pair of elements $x, y \in D$ is **consistent**, written $x \uparrow y$, if they are bounded.

PROPOSITION 2.2.30 ([Gun92, Theorem 5.5]). *A dcpo D is bounded-complete if and only if every consistent pair $x \uparrow y$ in D has a least upper bound.*

An important result about bounded complete domains is:

Proposition 2.2.31 ([Gun92, Lemma 5.10; AJ95, Exercise 4.3.11(2)]). *If D is a bounded-complete domain, then $\sqcap : D \times D \to D$ is continuous.*

Remark 2.2.32. Proposition 2.2.31 is not true in general for bounded-complete dcpos [Gun92, Exercise 5.16].

Definition 2.2.33. A bounded-complete dcpo satisfies the **d-property** if $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ whenever $y \uparrow z$ (equivalently, whenever $x \uparrow y$ and $x \uparrow z$). ◂

The bounded-complete dcpo of fig. 2.4 does not satisfy the d-property.

Definition 2.2.34. An algebraic domain satisfies the **I-property** if every compact element has a finite number of lower bounds. ◂

The domain $\{\bot \sqsubseteq \cdots \sqsubseteq 3 \sqsubseteq 2 \sqsubseteq 1\}$ does not satisfy the I-property [Zha91, p. 140].

Definition 2.2.35. A **dI-domain** is a bounded-complete ω-algebraic domain satisfying properties d and I. ◂

Theorem 2.2.36 ([Zha91, Theorem 6.2]). *If D is a bounded-complete ω-algebraic domain satisfying the I-property, then D is prime algebraic if and only if it is a dI-domain.*

We can characterize the prime elements of dI-domains. Say that x is *immediately below* y if $x \sqsubseteq y$ and for all $x \sqsubseteq z \sqsubseteq y$, either $x = z$ or $z = y$.

Proposition 2.2.37 ([Zha91, Lemma 6.1]). *The prime elements of a dI-domain D are the compact elements with a unique element immediately below them.*

We give Polarized SILL's functional layer a denotational semantics using dI-domains and continuous functions in section 8.3. Defining this semantics requires a cartesian-closed category of dI-domains. To construct such a category, it is well known that we must restrict our attention from continuous functions to *stable* functions.

Definition 2.2.38. A continuous function $f : D \to I$ between dI-domains is **stable** if for all $x \uparrow y$ in D, $f(x \sqcap y) = f(x) \sqcap f(y)$. ◂

Example 2.2.39. All upper adjoints are stable by proposition 2.2.19. ◂

Intuitively, a stable function is one where each finite (compact) piece of output is determined by a unique finite piece of input. Proposition 2.2.40 makes this fact explicit. It adapts [Gir86, Theorem 1.3] from qualitative domains to dI-domains.

Proposition 2.2.40 (Normal Form Theorem). *Let $f : X \to Y$ be a stable function of dI-domains and $a \in X$. If $q \in Y$ is prime and such that $q \sqsubseteq f(a)$, then:*

(1) there exists a $k \in \mathcal{K}(X)$ such that $k \sqsubseteq a$ and $q \sqsubseteq f(k)$, and
(2) if such a k' is chosen to be minimal, then k' is unique.

Proof. To see the first condition, observe first that $a = \bigsqcup^{\uparrow} \mathcal{K}(X)_a$ by algebraicity. By continuity, $f(a) = \bigsqcup^{\uparrow} f(\mathcal{K}(X))_a$. But q is compact, so there exists a $k \in \mathcal{K}(X)_a$ such that $q \sqsubseteq f(k)$.

Assume now that $k' \in \mathcal{K}(X)$ is chosen to be minimal with $k' \sqsubseteq a$ and $q \sqsubseteq f(k')$. Then k and k' are compatible, so $f(k \sqcap k') = f(k) \sqcap f(k')$ by stability. But then $q \sqsubseteq f(k \sqcap k')$, so $k' = k \sqcap k'$ by minimality, whence $k' \sqsubseteq k$. Because k was arbitrary, it follows that k' is the unique minimum. □

Warning 2.2.41. Proposition 2.2.40 *does not* state that the minimal k' is globally minimum such that $q \sqsubseteq f(k')$. It only states that if $k' \sqsubseteq a$ is chosen minimal such that $q \sqsubseteq f(k')$, then it is unique. This k' is sometimes called the **modulus of stability** $M(f, x, q)$ of f, x, and q [Abr07, p. 42].

Definition 2.2.42. Let $f : X \to Y$ be a stable function of dI-domains. Its **skeleton** is the set $\mathrm{sk}(f) = \{(k, q) \in \mathcal{K}(X) \times |Y| \mid k \text{ minimal with } q \sqsubseteq f(k)\}$. ◂

Skeletons are frequently called *traces* in the literature. To avoid confusion with the trace operators of section 2.3, we adopt the terminology of Girard [Giro6, Définition 2.8.2.16].

The skeleton of a stable function is well-defined by proposition 2.2.40. Proposition 2.2.43 adapts [Gir86, Theorem 1.4] from qualitative domains to dI-domains. Its proof can be found in [Zha91, Lemma 6.2].

Proposition 2.2.43 (Representation Theorem). *If $f : X \to Y$ is a stable function between dI-domains, then it is entirely determined by its skeleton: for all $x \in X$,*

$$f(x) = \bigsqcup \{q \mid \exists k \sqsubseteq x.(k,q) \in \mathrm{sk}(f)\}.$$

Definition 2.2.44. If $f, g : X \to Y$ are stable functions between dI-domains, then f is **stably less than** g, written $f \sqsubseteq_s g$, if for all $x \sqsubseteq y$, $f(x) = f(y) \sqcap g(x)$. ◂

Viewing f and g as functors between X and Y, this is exactly the statement that $\sqsubseteq_s$ is a cartesian natural transformation $\sqsubseteq_s : f \Rightarrow g$.

The stable ordering on functions is equivalent to the inclusion ordering on their skeletons:

Proposition 2.2.45. *Let $f, g : X \to Y$ be stable functions between dI-domains. Then $f \sqsubseteq_s g$ if and only if* $\mathrm{sk}(f) \subseteq \mathrm{sk}(g)$.

Proof. Sufficiency is given by [Zha91, Lemma 6.3]. To see necessity, let $x \sqsubseteq y$ be arbitrary. By proposition 2.2.43 and the assumption that $\mathrm{sk}(f) \subseteq \mathrm{sk}(g)$:

$$\begin{aligned} f(y) \sqcap g(x) &= \left(\bigsqcup \{q \mid \exists k \sqsubseteq x.(k,q) \in \mathrm{sk}(f)\}\right) \sqcap \left(\bigsqcup \{q \mid \exists k \sqsubseteq y.(k,q) \in \mathrm{sk}(g)\}\right) \\ &= \bigsqcup \left(\{q \mid \exists k \sqsubseteq x.(k,q) \in \mathrm{sk}(f)\} \sqcap \{q \mid \exists k \sqsubseteq y.(k,q) \in \mathrm{sk}(g)\}\right) \\ &= \bigsqcup \left(\{q \mid \exists k \sqsubseteq x.(k,q) \in \mathrm{sk}(f)\} \sqcap \{q \mid \exists k \sqsubseteq y.(k,q) \in \mathrm{sk}(f)\}\right) \\ &= \bigsqcup \{q \mid \exists k \sqsubseteq x.(k,q) \in \mathrm{sk}(f)\} \\ &= f(x). \end{aligned}$$

□

DI-domains and stable functions form a cartesian-closed category **Stab**. Its product is inherited from **DCPO**. The exponential $\mathbf{Stab}[X \to Y]$ is the dI-domain of stably ordered stable functions from X to Y. We refer the reader to [Gun92, § 5.2] for a proof that **Stab** is cartesian closed. For convenience, we present several useful facts related to its proof.

Proposition 2.2.46 ([Zha91, Lemma 6.4; Gun92, Lemma 5.17]). *If $F \subseteq \mathbf{Stab}[D \to E]$ is bounded, then its supremum is computed point-wise:*

$$\bigsqcup F = \lambda x. \bigsqcup_{f \in F} f(x).$$

Theorem 2.2.47 ([Gun92, Theorem 5.21]). *If D and E are dI-domains, then so is* $\mathbf{Stab}[D \to E]$.

To make a function $f : \prod_{i \in I} A_i \to B$ strict in a component $j \in I$, we use the continuous function $\mathrm{strict}_j : [\prod_{i \in I} A_i \to B] \to [\prod_{i \in I} A_i \to B]$:

$$\mathrm{strict}_j(f)\left((a_i)_{i \in I}\right) = \begin{cases} \bot_B & \text{if } a_j = \bot_{A_j} \\ f\left((a_i)_{i \in I}\right) & \text{otherwise.} \end{cases} \tag{1}$$

2.2.1. Constructions On Partially Ordered Sets. We present various functorial constructions used to form new posets, dcpos, domains, etc., from old.

The **lifting** $P_\bot$ of a poset P is the poset obtained by adjoining a new bottom element to P. Explicitly, its elements are given by the set $\{\bot\} \cup \{[p] \mid p \in P\}$. As usual, $\bot \sqsubseteq p$ for all $p \in P_\bot$, and $[p] \sqsubseteq [q]$ if and only if $p \sqsubseteq q$ in P. Lifting sends morphisms $f : P \to Q$ to strict morphisms $f_\bot : P_\bot \to Q_\bot$ such that $f_\bot([x]) = [f(x)]$. In particular, the diagram

$$\begin{array}{ccc} P & \xrightarrow{\mathsf{up}} & P_\bot \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f_\bot} \\ Q & \xrightarrow{\mathsf{up}} & Q_\bot \end{array} \tag{2}$$

commutes in **Poset** for all morphisms $f : P \to Q$.

The (categorical) **product** $P \times Q$ of posets P and Q is given by the cartesian product of the underlying sets of P and Q, and the ordering is given component-wise. Given pointed posets P_i for $i \in I$ and a $j \in I$, we write $\iota_j : P_j \to \prod_{i \in I} P_i$ for the map that sends $p \in P_j$ to the tuple $(\bot, \ldots, \bot, p, \bot, \ldots, \bot)$ whose j-th component is p, and whose other components are all $\bot$. The **smash product** $P \otimes Q$ of pointed posets P and Q identifies all elements of the form $(\bot, q)$ or $(p, \bot)$ of $P \times Q$. Explicitly,

$$P \otimes Q = \{(p, q) \in P \times Q \mid p \neq \bot \wedge q \neq \bot\} \cup \{(\bot, \bot)\},$$

and the ordering is given component-wise. We use the same notation for projections out of smash products as we do for projections out of products. We warn the reader that smash products do not in general play the role of categorical products in categories of dcpos. This is because the mediating morphism required by universality may not exist.

The **disjoint union** $P_1 \uplus P_2$ of posets P_1 and P_2 is the poset whose underlying set is the disjoint union of P_1 and P_2, and whose ordering is given by $(i, x) \sqsubseteq (j, y)$ if and only if $i = j$ and $x \sqsubseteq y$ in P_i. The **coalesced sum** $P_1 \oplus P_2$ of pointed posets P_1 and P_2 identifies all labelled bottom elements of the disjoint union $P_1 \uplus P_2$. Explicitly,

$$P_1 \oplus P_2 = \{\bot\} \cup \{(i, p) \mid 1 \leq i \leq 2 \wedge p \in P_i \wedge p \neq \bot\}$$

The ordering is given by $x \sqsubseteq y$ if $x = \bot$, or if $x = (i, x')$ and $y = (i, y')$ and $x' \sqsubseteq y'$ in P_i.

Proposition 2.2.48. *Where defined, the lifting, product, smash product, disjoint union, or coalesced sum of a poset, dcpo, or (bc-,dI-)domain is again a poset, dcpo, or (bc-,dI-)domain.*

Products play the categorical role of product in the various categories. Coalesced sums play the role of coproducts in the various subcategories of pointed posets with strict morphisms. We refer the reader to [AJ95, § 3.2] for further properties of the aforementioned constructions. Given a functor F into a category of posets closed under lifting, we often abbreviate $(-)_\bot \circ F$ as $F_\bot$.

Lemmas 2.2.49 and 2.2.50 describe embedding-projection pairs involving the above constructions. They will frequently be used in the denotational semantics of Polarized SILL.

Lemma 2.2.49. *If $l_i : A_i \to L_i$ and $r_i : A_i \to R_i$ are such that $\langle l_i, r_i \rangle : A_i \to L_i \times R_i$ is an embedding for $1 \leq i \leq n$, then*

$$\left\langle \prod_{i=1}^n l_i, \prod_{i=1}^n r_i \right\rangle : \prod_{i=1}^n A_i \to \left(\prod_{i=1}^n L_i \right) \times \left(\prod_{i=1}^n R_i \right)$$

is an embedding with associated projection

$$\left(\prod_{i=1}^n L_i \right) \times \left(\prod_{i=1}^n R_i \right) \xrightarrow{\cong} \prod_{i=1}^n L_i \times R_i \xrightarrow{\prod_{i=1}^n \langle l_i, r_i \rangle^p} \prod_{i=1}^n A_i.$$

Lemma 2.2.50. *Let $F, G : \mathbf{C} \to \mathbf{Poset}_{\bot!}$ be functors, where $\mathbf{Poset}_{\bot!}$ is the category of pointed posets and strict monotone maps. The natural transformation $\delta : ((-)_\bot F) \times G \Rightarrow (-)_\bot (F \times G)$ given by*

$$\delta_C(x, y) = \begin{cases} \bot & \text{if } x = \bot \\ [(z, y)] & \text{if } x = [z]. \end{cases}$$

is a natural family of projection. The associated family of embeddings

$$(\delta_C)^e (x) = \begin{cases} (\bot, \bot) & \text{if } x = \bot \\ ([a], b) & \text{if } x = [(a, b)] \end{cases}$$

is natural, and each component is stable.

If $\langle \phi, \gamma \rangle : H \Rightarrow F \times G : \mathbf{C} \to \mathbf{Poset}_{\bot!}$ is natural, then

$$\langle (-)_\bot \phi, \mathsf{down} * \gamma \rangle = \delta^e \circ (-)_\bot \langle \phi, \gamma \rangle : (-)_\bot H \Rightarrow (-)_\bot F \times G.$$

Proof. Straightforward computation shows that both families are natural, and that they form an e-p-pair. It is also clear that each component $(\delta_C)^e$ is stable. A calculation shows the equality of natural transformations. □

2.2.1.1. *ω-Colimits.* The category **DCPO** is closed under colimits of a class of diagrams called "expanding sequences" [AJ95, Definition 3.3.6 and Theorem 3.3.7]. We specialize these results to diagrams of shape ω.

Definition 2.2.51. Let ω be the category induced by the poset $\mathbb{N}$ under the usual ordering. Explicitly, its objects are natural numbers, and there exists a unique map $m \to m + k$ for all $m, k \geq 0$. ◂

Definition 2.2.52. An **ω-chain** is a diagram of shape ω. ◂

Colimits of expanding sequences in **DCPO** follow the usual pattern of colimits in categories of sets with structure (cf. [Rie16, §3.2, § 3.5]. Theorem 2.2.53 is a special case of [AJ95, Theorem 3.3.7].

Theorem 2.2.53 (Limit-Colimit Coincidence). *Let $(e_{mn} : D_n \to D_m)_{n \leq m}$ be an ω-chain in* **DCPO** *such that each e_{mn} is an embedding, and write p_{nm} for $e^p_{mn} : D_m \to D_n$. Define:*

$$D = \{(x_n)_{n\in\mathbb{N}} \in \prod_{n\in\mathbb{N}} D_n \mid \forall n \leq m. x_n = p_{nm}(x_m)\},$$
$$p_m((x_n)_{n\in\mathbb{N}}) = x_m : D \to D_m \text{ for } m \in \mathbb{N},$$
$$e_m(x) = \left(\bigsqcup\nolimits^{\uparrow}_{k\geq n,m} (p_{nk} \circ e_{km})(x) \right)_{n\in\mathbb{N}} : D_m \to D \text{ for } m \in \mathbb{N}.$$

Then

1. *The maps (e_m, p_m) form e-p-pairs, and $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} e_n \circ p_n = \mathsf{id}_D$.*
2. *The cone $(p_m : D \to D_m)_{m\in\mathbb{N}}$ is limiting. Given any other cone $(g_m : C \to D_m)_{m\in\mathbb{N}}$, the unique mediating morphism of cones is $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} e_n \circ g_n$.*
3. *The cocone $(e_m : D_m \to D)_{m\in\mathbb{N}}$ is colimiting. Given any other cocone $(f_m : D_m \to E)_{m\in\mathbb{N}}$, the unique mediating morphism of cocones is $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} f_n \circ p_n$.*

Let the canonical representatives of colimits of ω-chains of embeddings be given by theorem 2.2.53.

Though the category **Stab** of dI-domains and stable maps is not closed under ω-colimits of embeddings, it is closed under ω-colimits of "rigid" embeddings:

Definition 2.2.54 ([KP93, Definition 9.3])**.** An embedding-projection pair $e : D \leftrightarrows E : p$ is **rigid** if it additionally satisfies

$$\forall x \in D. \forall y \in E. y \sqsubseteq e(x) \supset y = (e \circ p)(y).$$ ◂

Intuitively, an embedding is **rigid** if its image is downward-closed. Rigid embeddings are exactly the embeddings given when homsets are ordered stably instead of point-wise:

Proposition 2.2.55 ([CGW88, Lemma 2; Zha92, p. 168]). *Consider an embedding-projection pair $e : D \leftrightarrows E : p$ between algebraic domains D and E, with e and p continuous. Then (e, p) is rigid if and only if $e \circ p \sqsubseteq_s \mathsf{id}$.*

Proposition 2.2.56 ([Ber94, p. 34; CGW88, p. 350]). *The category* $\mathbf{Stab}^{re}$ *of dI-domains and rigid embeddings is closed under ω-colimits.*

Corollary 2.2.64, below, shows that the ω-colimits of proposition 2.2.56 coincide with those given by theorem 2.2.53.

2.2.2. Order-Enriched Category Theory. We refer the reader to [Gun92, Chapter 10; SP82; Fio94, § 2.3] for additional background on order-enriched categories and **O**-categories.

Definition 2.2.57. An **O-category** [SP82, Definition 5] (or **DCPO**-enriched category) is a category **K** where every hom-set $\mathbf{K}(C, D)$ is a dcpo, and where composition of morphisms is continuous with respect to the partial ordering on morphisms. ◂

Example 2.2.58. The category **DCPO** is an **O**-category. Functor categories $\mathbf{Cat}[\mathbf{C} \to \mathbf{D}]$ are **O**-categories whenever **D** is an **O**-category. ◂

Example 2.2.59. Full subcategories of **O**-category are again **O**-categories. ◂

Remark 2.2.60. Any given subcategory of **DCPO** may induce distinct **O**-categories. For example, the category **Stab** of dI-domains and stable functions induces the **O**-category whose homsets are stably ordered, as well as the **O**-category whose homsets are pointwise ordered. This subtlety has critical implications when solving domain equations. Indeed, the category of dI-domains whose morphisms are embeddings under the stable ordering is closed under ω-colimits, while the category of dI-domains whose morphisms are embeddings under the pointwise ordering is not.

Definition 2.2.61. A functor $F : \mathbf{D} \to \mathbf{E}$ between **O**-categories is **locally continuous** if the maps $f \mapsto F(f) : \mathbf{D}(D_1, D_2) \to \mathbf{E}(F(D_1), F(D_2))$ are continuous for all objects D_1, D_2 of **D**. ◂

Small **O**-categories form a 2-cartesian closed category **O**, where horizontal morphisms are locally continuous functors and vertical morphisms are natural transformations.

Example 2.2.62. The functors defining lifting, products, smash products, disjoint unions, and coalesced sums are all locally continuous relative to the pointwise and stable ordering. ◂

When **K** is an **O**-category, we write $\mathbf{K}^e$ for the subcategory of **K** whose morphisms are embeddings. The category $\mathbf{K}^e$ is not in general an **O**-category under the induced ordering [SP82, p. 768].

A cocone $\kappa : J \Rightarrow A$ in $\mathbf{K}^e$ is an **O-colimit** [SP82, Definition 7] if $(\kappa_n \circ \kappa_n^p)_n$ is an ascending chain in $\mathbf{K}(A, A)$ and $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} \kappa_n \circ \kappa_n^p = \mathsf{id}_A$. **K** is **O-cocomplete** if every ω-chain in $\mathbf{K}^e$ has an **O**-colimit in **K**. Our interest in **O**-colimits is due to proposition 2.2.63, which appears as [SP82, Propositions A and D] and as part of the proof of [SP82, Proposition A]. It characterizes the colimits that will be used to construct the denotations of recursive session types. Parts of proposition 2.2.63 can also be found in the proof of [Gun92, Theorem 10.4] or specialized to **DCPO** as [AJ95, Theorem 3.3.7]. This result gives us an explicit characterization of colimits in **O**-categories. We invite the reader to compare it to theorem 2.2.53.

Proposition 2.2.63 ([SP82, Propositions A and D]). *Let* **K** *be an* **O***-category,* Φ *an* ω*-chain in* $\mathbf{K}^e$*, and* $\alpha : \Phi \Rightarrow A$ *a cocone in* **K**.

1. *If* $\beta : \Phi \Rightarrow B$ *is a cocone in* $\mathbf{K}^e$*, then* $(\alpha_n \circ \beta_n^p)_{n\in\mathbb{N}}$ *is an ascending chain in* $\mathbf{K}(B, A)$ *and the morphism* $\theta = \bigsqcup^{\uparrow}_{n\in\mathbb{N}} \alpha_n \circ \beta_n^p$ *is mediating from* β *to* α.
2. *If* $\beta : \Phi \Rightarrow B$ *is an* **O***-colimit and* α *lies in* $\mathbf{K}^e$*, then* θ *is an embedding.*
3. *If* α *is an* **O***-colimit, then* α *is colimiting in both* **K** *and* $\mathbf{K}^e$.
4. *If* α *is colimiting in* **K***, then* α *lies in* $\mathbf{K}^e$ *and is an* **O***-colimit.*

Corollary 2.2.64. *Let* **Stab** *be the* **O***-category of dI-domains and stable maps, where homsets are stably ordered. Then* $\mathbf{Stab}^e$ *is the category of dI-domains and rigid embeddings. Let* Φ *be an* ω*-chain in* $\mathbf{Stab}^e$*, and let* $\kappa : \Phi \Rightarrow A$ *be colimiting in* **Stab***. Then* $\kappa : \Phi \Rightarrow A$ *is also colimiting in* **DCPO**.

Proof. The first part is exactly proposition 2.2.55. If $\kappa : \Phi \Rightarrow A$ is colimiting in **Stab**, then it is colimiting in $\mathbf{Stab}^e$. It follows that κ is an **O**-colimit relative to the stable ordering. The stable ordering implies the pointwise ordering, and directed suprema in $\mathbf{Stab}[A \to A]$ are computed point-wise by proposition 2.2.46. It follows that $\bigsqcup^{\uparrow}_{n\in\mathbb{N}} \kappa_n \circ \kappa_n^p = \mathsf{id}_A$ in $\mathbf{DCPO}[A \to A]$, so κ is an **O**-colimit in **DCPO**, where homsets are ordered pointwise. We conclude that κ is colimiting in **DCPO**. □

2.3. Properties of Parametrized Fixed-Point and Trace Operators

We saw in section 2.2 that the category **DCPO** is equipped with a fixed-point operator. It and many other categories of interest in semantics are also equipped with a parametrized fixed-point operator. Particularly nice parametrized fixed-point operators are called *Conway operators* [BÉ96].

Definition 2.3.1 ([SP00, Definitions 2.2 and 2.4])**.** A **parametrized fixed-point operator** on a cartesian category $\mathbf{M}$ is a family of morphisms $(\cdot)^\dagger : \mathbf{M}[X \times A \to A] \Rightarrow \mathbf{M}[X \to A]$ satisfying:

(1) Naturality: for any $g : X \to Y$ and $f : Y \times A \to A$,
$$f^\dagger \circ g = (f \circ (g \times \mathsf{id}_A))^\dagger : X \to A.$$
(2) The parametrized fixed-point property: for any $f : X \times A \to A$,
$$f \circ \langle \mathsf{id}_X, f^\dagger \rangle = f^\dagger : X \to A.$$

It is a **Conway operator** if it additionally satisfies:

(3) Parameterized dinaturality: for any $f : X \times B \to A$ and $g : X \times A \to B$,
$$f \circ \langle \mathsf{id}_X, (g \circ \langle \pi_X, f \rangle)^\dagger \rangle = (f \circ \langle \pi_1, g \rangle)^\dagger : X \to A.$$
(4) The diagonal property: for any $f : X \times A \times A \to A$,
$$(f \circ (\mathsf{id}_X \times \Delta))^\dagger = \left(f^\dagger\right)^\dagger : X \to A,$$
where $\Delta : A \to A \times A$ is the diagonal map. ◀

Proposition 2.3.2. *The operator* $(\cdot)^\dagger : \mathbf{DCPO}_\perp[X \times A \to A] \Rightarrow \mathbf{DCPO}_\perp[X \to A]$ *given by*
$$f^\dagger(x) = \mathrm{lfp}(\lambda a \in A. f(x, a))$$
is a Conway operator.

Using proposition 2.2.16, we can characterize this Conway operator in two ways:

Corollary 2.3.3. *Let X be a dcpo, A a pointed dcpo, and $f : X \times A \to A$ a continuous function. The parametrized fixed-point $f^\dagger$ is equivalently constructed:*

(1) using the Kleene fixed-point theorem, with $f^\dagger(x) = \bigsqcup^\uparrow_{n \in \mathbb{N}} (\lambda a \in A. f(x, a))^n(\perp_A)$;
(2) using a variant of the Knaster-Tarski theorem, with $f^\dagger(x) = \bigsqcap \{a \in A \mid f(x, a) \sqsubseteq a\}$.

Traces are a third kind of fixed-point operator. Traces were first discovered by Căzănescu and Ştefănescu [CŞ90, § 4.3] and then independently rediscovered by Joyal, Street, and Verity [JSV96] in the setting of balanced monoidal categories.

Definition 2.3.4 ([BH03, Definition 2.4])**.** Let $(\mathbf{M}, \otimes, I, \lambda, \rho, \alpha, \sigma)$ be a symmetric monoidal category. A **trace** on $\mathbf{M}$ is a family of functions
$$\mathrm{Tr}^U_{A,B} : \mathbf{M}(A \otimes U, B \otimes U) \to \mathbf{M}(A, B)$$
satisfying the following conditions:

(1) Naturality in A (left tightening): if $f : A' \otimes U \to B \otimes U$ and $g : A \to A'$, then
$$\mathrm{Tr}^U_{A,B}(f \circ (g \otimes \mathsf{id}_U)) = \mathrm{Tr}^U_{A',B}(f) \circ g : A \to B.$$
(2) Naturality in B (right tightening): if $f : A \otimes U \to B' \otimes U$ and $g : B' \to B$, then
$$\mathrm{Tr}^U_{A,B}((g \otimes \mathsf{id}_U) \circ f) = g \circ \mathrm{Tr}^U_{A,B'}(f) : A \to B.$$
(3) Dinaturality (sliding): if $f : A \otimes U \to B \otimes V$ and $g : V \to U$, then
$$\mathrm{Tr}^U_{A,B}((\mathsf{id}_B \otimes g) \circ f) = \mathrm{Tr}^V_{A,B}(f \circ (\mathsf{id}_A \otimes g)) : A \to B.$$

(4) Action (vanishing): if $f : A \to B$, then

$$\mathrm{Tr}^I_{A,B}\left(\rho^{-1} \circ f \circ \rho\right) = f : A \to B,$$

and if $f : A \otimes (U \otimes V) \to B \otimes (U \otimes V)$, then

$$\mathrm{Tr}^{U\otimes V}_{A,B}(f) = \mathrm{Tr}^U_{A,B}\left(\mathrm{Tr}^V_{A\otimes U, B\otimes U}\left(\alpha^{-1} \circ f \circ \alpha\right)\right).$$

(5) Superposing:[5] if $f : A \otimes U \to B \otimes U$, then

$$\mathrm{Tr}^U_{C\otimes A, C\otimes B}\left(\alpha^{-1} \circ (\mathsf{id}_C \otimes f) \circ \alpha\right) = \mathsf{id}_C \otimes \mathrm{Tr}^U_{A,B}(f) : C \otimes A \to C \otimes B.$$

(6) Yanking: for all U,

$$\mathrm{Tr}^U_{U,U}\left(\sigma_{U,U}\right) = \mathsf{id}_U : U \to U.$$

In this case, we call **M** a **symmetric traced category**.[6] ◀

Example 2.3.5. The following defines a trace on the category $\mathbf{DCPO}_\perp$ [AHS02, § 5.6]:

$$\mathrm{Tr}^X_{A,B}(f) = \pi^{B\times X}_B \circ f \circ \left\langle \mathsf{id}_A, \left(\pi^{B\times X}_X \circ f\right)^\dagger \right\rangle.$$ ◀

For conciseness and clarity, we will often elide the subscripts on the trace operator when they are clear from context.

We can reason about traces using string diagrams, where the trace operator is captured by looping the output of a function back into the corresponding input. We adopt the graphical notation of [Sel11; Mal10, § 2.8]. Figure 2.5 presents the trace axioms in this graphical notation.

Theorem 2.3.6 ([Mal10, Theorem 2.8.1]). *A well-formed equation between morphism terms in the language of traced symmetric monoidal categories follows from the axioms of traced symmetric monoidal categories if and only if it holds, up to isomorphism of diagrams, in the graphical language.*

Hasegawa [Has99, Theorem 7.1] and Hyland [BH03, p. 281] independently discovered that a cartesian category has a trace if and only if it has a Conway operator. The following is a special case of the proof of the Hasegawa-Hyland theorem.

Proposition 2.3.7. *For all $f : A \times X \to B \times X$, we have $f \circ \langle \mathsf{id}_A, (\pi^{B\times X}_X \circ f)^\dagger \rangle = (f \circ \pi^{B\times X\times A}_{A\times X})^\dagger$. Consequently, $\mathrm{Tr}^X_{A,B}(f) = \pi^{B\times X}_B \circ (f \circ \pi^{B\times X\times A}_{A\times X})^\dagger$ when* Tr *is defined as in example 2.3.5.*

Corollary 2.3.8. *Let A be a dcpo, let B and X be pointed dcpos, and let $f : A \times X \to B \times X$ be a continuous function. Then the trace $\mathrm{Tr}^X_{A,B}(f)$ is equivalently constructed:*

(1) using the Kleene fixed-point theorem, with

$$\mathrm{Tr}^X(f)(a) = \pi^{B\times X}_B\left(\bigsqcup\nolimits^{\uparrow}_{n\in\mathbb{N}} \left(\lambda\,(b,x)\,.f\,(a,x)\right)^n (\perp_B, \perp_X)\right);$$

(2) using a variant of the Knaster-Tarski theorem, with

$$\mathrm{Tr}^X(f)(a) = \pi^{B\times X}_B\left(\bigsqcap \{(b,x) \in B \times X \mid f(a,x) \sqsubseteq (b,x)\}\right).$$

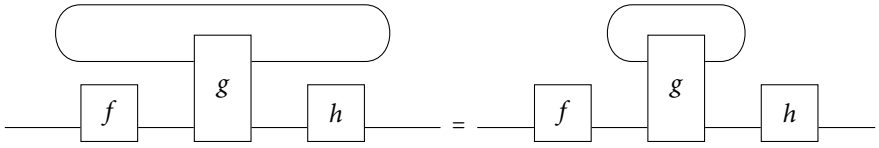
The following corollary gives a collection of identities that will be useful for reasoning about traces.

Corollary 2.3.9. *Let A be a dcpo, let B and X be pointed dcpos, and let $f : A \times X \to B \times X$ be a continuous function. Let $a \in A$ be arbitrary, and set*

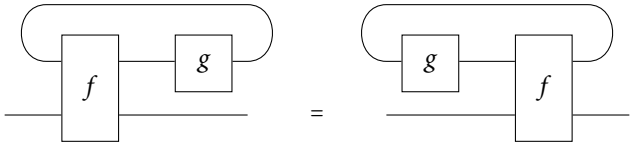
$$P = \{(b,x) \in B \times X \mid f(a,x) \sqsubseteq (b,x)\},$$
$$(\beta,\chi) = \bigsqcap P.$$

[5] This axiom is sometimes replaced by the "strength" axiom [Sel11; Mal10]: if $f : A \otimes X \to B \otimes X$ and $g : C \to D$, then $\mathrm{Tr}^X_{C\otimes A, D\otimes B}\left(\alpha^{-1} \otimes (g \otimes f) \circ \alpha\right) = g \otimes \mathrm{Tr}^X_{A,B}(f)$. Both collections of axioms give equivalent definitions.

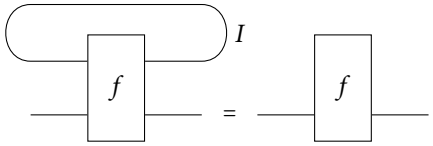
[6] This categorical nomenclature is due to Selinger [Sel11].



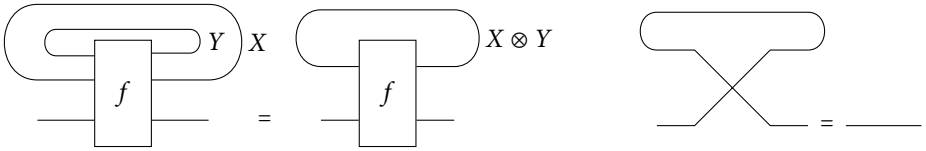
(A) Naturality (left and right tightening)



(B) Dinaturality (sliding)

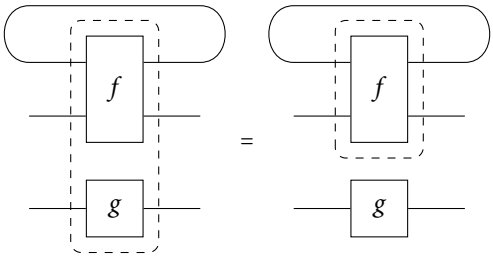


(C) Vanishing I



(D) Vanishing II

(E) Yanking



(F) Strength

FIGURE 2.5. String diagram notation for traced categories

Then $(\beta, \chi) \in P$*, and the following identities hold:*

$$\left(f \circ \pi^{A\times B\times X}_{A\times X}\right)^{\dagger}(a) = (\beta,\chi), \qquad \mathrm{Tr}^X(f)(a) = \beta,$$
$$f(a,\chi) = (\beta,\chi), \qquad \min\{b \mid \exists x \in X. f(a,x) \sqsubseteq (b,x)\} = \beta.$$

Proof. We begin by showing that $\left(f \circ \pi^{A\times B\times X}_{A\times X}\right)^{\dagger}(a) = (\beta,\chi)$. By corollary 2.3.3, we have

$$\begin{aligned}
&\left(f \circ \pi^{A\times B\times X}_{A\times X}\right)^{\dagger}(a)\\
&= \bigsqcap \left\{(b,x) \in B\times X \mid \left(f \circ \pi^{A\times B\times X}_{A\times X}\right)(b,a,x) \sqsubseteq (b,x)\right\}\\
&= \bigsqcap \{(b,x) \in B\times X \mid f(a,x) \sqsubseteq (b,x)\}\\
&= \bigsqcap P\\
&= (\beta,\chi).
\end{aligned}$$

By the parametrized fixed-point property (definition 2.3.1),

$$\begin{aligned} f(a,\chi) &= \left(f \circ \pi^{A\times B\times X}_{A\times X}\right)(a,\beta,\chi) \\ &= \left(\left(f \circ \pi^{A\times B\times X}_{A\times X}\right) \circ \left\langle \mathsf{id}_A, \left(f \circ \pi^{A\times B\times X}_{A\times X}\right)^\dagger \right\rangle\right)(a) \\ &= \left(f \circ \pi^{A\times B\times X}_{A\times X}\right)^\dagger (a) \\ &= (\beta,\chi). \end{aligned}$$

So $(\beta,\chi) \in P$. By corollary 2.3.8, $\mathrm{Tr}^X(f)(a) = \pi_B(\bigsqcap P) = \beta$. Finally, the infimimum $\bigsqcap P$ is the least element of P because $\bigsqcap P \in P$. Infima of products are computed component-wise, so:

$$\begin{aligned} &\min\{b \mid \exists x \in X. f(a,x) \sqsubseteq (b,x)\} \\ &= \pi_B\left(\min\{(b,x) \mid f(a,x) \sqsubseteq (b,x)\}\right) \\ &= \pi_B\left(\bigsqcap P\right) \\ &= \beta. \end{aligned}$$

□

2.4. Generalized Abstract Binding Trees

Processes in Polarized SILL feature two kinds of place-holders. The first kind is **variables**, which stand for unknown values, and whose meaning is given by substitution. The second kind is **symbols** or names, which do not stand for anything, and whose meaning is given by their structural role as identifiers. Both variables and symbols can appear in free and bound positions in processes. To make the important distinction between variables and symbols precise, we generalize abstract binding trees [Har16, § 1.2] to allow for bound symbols. We also generalize abstract binding trees to allow for infinite trees. Abstract binding trees find their roots in the work of Church, and their theory was developed by Harper. Our exposition closely follows Harper's.

2.4.1. Abstract Binding Trees. Abstract binding trees, or *abt*s, generalize abstract syntax trees to account for binding and scoping. Fix a finite set $\mathcal{S}$ of **sorts**. A **sort** $s \in \mathcal{S}$ is an identifier for a sort, type, kind, or variety of tree. Abstract binding trees are constructed from operators, which combine zero or more arguments of given sorts to form new trees of a given sort. Importantly, each argument can use zero or more bound variables.

Valencies specify the number of bound variables in each argument. A *valency* is a syntactic expression $s_1,\ldots,s_n.s$ (abbreviated $\vec{s}.s$) with $s_1,\ldots,s_n,s \in S$ and $n \geq 0$. It specifies that an argument has sort s, and that it has n bound variables representing trees of sort s_i. An *arity* is a syntactic expression $(\nu_1,\ldots,\nu_n)s$ that specifies that an operator of sort s takes $n \geq 0$ arguments of valency $\nu_1,\ldots,\nu_n$. *Operators* are elements of an arity-indexed family of sets $\mathcal{O} = \{\mathcal{O}_\alpha\}$.

To specify the sorts of variables that appear in abstract binding trees, we assume a sort-indexed family of variables $\mathcal{X} = \{\mathcal{X}_s\}_{s\in\mathcal{S}}$. We say that x is a *variable of sort* s if $x \in \mathcal{X}_s$. We say that x is *fresh* for $\mathcal{X}$ if $x \notin \mathcal{X}_s$ for all s. Given a family $\mathcal{X}$ of variables, a variable x fresh for $\mathcal{X}$, and a sort s, we write $\mathcal{X}, x$ for the family of variables obtained by extending $\mathcal{X}_s$ with x; context will disambiguate the sort of x in $\mathcal{X}, x$. More generally, given a family $\mathcal{X}'$ of fresh variables for $\mathcal{X}$, we write $\mathcal{X}, \mathcal{X}'$ for the family obtained by extending $\mathcal{X}_s$ with $\mathcal{X}'_s$.

Abstract binding trees form sort-indexed families $\mathcal{B}[\mathcal{X}] = \{\mathcal{B}[\mathcal{X}]_s\}_{s\in\mathcal{S}}$ over sort-indexed families of variables $\mathcal{X}$. *Fresh renamings* account for the free renaming of bound variables in abts. Given variables $\mathcal{Y}$ and fresh variables $\mathcal{Y}'$ (relative to $\mathcal{X}$), a *fresh renaming* is a bijection $\rho\colon \mathcal{Y} \leftrightarrow \mathcal{Y}'$. Given an abt $a \in \mathcal{B}[\mathcal{X},\mathcal{Y}]_s$, replacing each $y \in \mathcal{Y}$ in a by its counterpart in $\mathcal{Y}'$ gives a new abt $[\rho]a \in \mathcal{B}[\mathcal{X},\mathcal{Y}']_s$.

Definition 2.4.1 ([Har16, p. 8])**.** The family of **abstract binding trees** is the least sort-indexed family $\mathcal{B}[\mathcal{X}] = \{\mathcal{B}[\mathcal{X}]_s\}_{s\in\mathcal{S}}$ of sets closed under the following conditions:

(1) if $x \in \mathcal{X}_s$, then $x \in \mathcal{B}[\mathcal{X}]_s$;
(2) for each operator o of arity $(\vec{s}_1.s_1,\ldots,\vec{s}_n.s_n)s$, if for each $1 \leq i \leq n$ and each fresh renaming $\rho_i\colon \vec{x}_i \leftrightarrow \vec{x}'_i$, we have $[\rho_i]a_i \in \mathcal{B}[\mathcal{X},\vec{x}'_i]$, then $o(\vec{x}_1.a_1,\ldots,\vec{x}_n.a_n) \in \mathcal{B}[\mathcal{X}]_s$. ◂

Remark 2.4.2. The family $\mathcal{X}$ of variables in the above definition is not fixed, but varies according to the variables introduced by operators' abstraction operations.

We always identify abstract binding trees up to α-equivalence:

Definition 2.4.3 ([Har16, p. 9])**.** **α-Equivalence** on abstract binding trees is the strongest congruence $\equiv_\alpha$ such that

(1) $x \equiv_\alpha x$ for all variables x;
(2) $o(\vec{x}_1.a_1, \ldots, \vec{x}_n.a_n) \equiv_\alpha o(\vec{x}'_1.a'_1, \ldots, \vec{x}'_n.a'_n)$ if for every $1 \leq i \leq n$, $[\rho_i]a_i \equiv_\alpha [\rho'_i]a'_i$ for all fresh renamings $\rho_i : \vec{x}_i \leftrightarrow \vec{z}_i$ and $\rho'_i : \vec{x}'_i \leftrightarrow \vec{z}_i$. ◂

Example 2.4.4. The family of abstract binding trees for the unityped[7] λ-calculus is generated by the following data:

- the family $\mathcal{S} = \{t\}$ of sorts;
- the sort-indexed family $\mathcal{O}$ where all sets are empty except for

$$\mathcal{O}_{(t.t)t} = \{\lambda\},$$
$$\mathcal{O}_{(t,t)t} = \{\mathsf{app}\}.$$

The set of abstract binding trees over $\mathcal{X}_t = \{x, y\}$ is:

$$\mathcal{B}[\mathcal{X}]_t = \{x, y, \mathsf{app}(x, y), \mathsf{app}(y, x), \lambda(x.x), \lambda(x.y), \mathsf{app}(x, \lambda(x.x)), \ldots\}.$$

The result of substituting $\mathsf{app}(x, y)$ for x in $\mathsf{app}(x, \lambda(x.x))$ is

$$[\mathsf{app}(x, y)/x]\,(\mathsf{app}(x, \lambda(x.x))) = \mathsf{app}(\mathsf{app}(x, y), \lambda(x.x)).$$ ◂

As described above, the meaning of variables is given by capture-avoiding substitution, where a variable x of sort s stands for an unknown abstract binding tree of sort s. Capture-avoiding substitution is formally defined as follows:

Definition 2.4.5. A **morphism of abts** $\sigma : \mathcal{B}[\mathcal{X}] \rightsquigarrow \mathcal{B}[\mathcal{Y}]$ is a choice $\sigma(x) \in \mathcal{B}[\mathcal{Y}]$ for each variable $x \in \mathcal{X}$ such that if $x \in \mathcal{X}_s$, then $\sigma(x) \in \mathcal{B}[\mathcal{X}]_s$ is of the same sort. ◂

Definition 2.4.6. Given a morphism $\sigma : \mathcal{B}[\mathcal{X}] \rightsquigarrow \mathcal{B}[\mathcal{Y}]$ and an abt $a \in \mathcal{B}[\mathcal{X}]$, the abt $[\sigma]a \in \mathcal{B}[\mathcal{Y}]$ is given by **simultaneous capture-avoiding substitution** of $\sigma(x)$ for x in a, a procedure recursively defined on the structure of a by:

(1) $[\sigma]x = \sigma(x)$;
(2) $[\sigma](o(\vec{x}_1.a_1, \ldots, \vec{x}_n.a_n)) = o\,(\vec{x}_1.[\sigma]a_1, \ldots, \vec{x}_n.[\sigma]a_n)$ where we assume without loss of generality that $\mathcal{X} \cap \vec{x}_i = \varnothing$ for all $1 \leq i \leq n$. ◂

As a special case, we write $[b_1, \ldots, b_n/x_1, \ldots, x_n]$ for the morphism that substitutes b_i for x_i and fixes all other variables.

2.4.2. General Binding Trees. Just as abts extended abstract syntax trees to account for bound variables, general binding trees, or *gbt*s, extend abstract binding trees to account for bound symbols. The development of gbts closely the development of abts, except that we also take symbols into account. A **sort** $s \in \mathcal{S}$ is an identifier for a sort, type, kind, or variety of tree. A *valency* is a syntactic expression $s_1, \ldots, s_m.s'_1, \ldots, s'_n.s$, where $s_i, s'_i, s \in S$ are sorts. The sorts $s_1, \ldots, s_m$ describe the sorts of bound symbols; the sorts $s'_1, \ldots, s'_n$ describe the sorts of bound variables. *Arities* are again syntactic expressions $(\nu_1, \ldots, \nu_n)s$, where the ν_i are valencies. Operators are again given by arity-indexed families of sets.

In addition to the sort-indexed family of variables $\mathcal{X}$, we assume a sort-indexed family $\mathcal{U} = \{\mathcal{U}_s\}_{s \in \mathcal{S}}$ of disjoint sets of symbols. The terminology and notation for variables from above carries over to symbols. We assume that the sets of variables and of symbols are disjoint from each other.

We extend fresh renamings to account for the renaming of bound symbols. Given symbols $\mathcal{V}$ and variables $\mathcal{Y}$, and fresh symbols $\mathcal{V}'$ and fresh variables $\mathcal{Y}'$ (relative to $\mathcal{U}$ and $\mathcal{X}$, respectively),

[7] The "untyped" λ-calculus is really "unityped" or "monotyped".

a *fresh renaming* is a bijection $\rho : \mathcal{V}; \mathcal{Y} \leftrightarrow \mathcal{V}'; \mathcal{Y}'$ that sends symbols to symbols and variables to variables. Replacing symbols by symbols and variables by variables according to ρ in a tree $a \in \mathcal{B}[\mathcal{U}, \mathcal{V}; \mathcal{X}, \mathcal{Y}]_s$ gives a new tree $[\rho]a \in \mathcal{B}[\mathcal{U}, \mathcal{V}'; \mathcal{X}, \mathcal{Y}']_s$.

Definition 2.4.7. The family of **general binding trees** is the largest sort-indexed family $\mathcal{B}[\mathcal{U}; \mathcal{X}] = \{\mathcal{B}[\mathcal{U}; \mathcal{X}]_s\}_{s \in \mathcal{S}}$ of sets closed under the following conditions:

(1) if $x \in \mathcal{X}_s$, then $x \in \mathcal{B}[\mathcal{U}; \mathcal{X}]_s$;
(2) if $u \in \mathcal{U}_s$, then $u \in \mathcal{B}[\mathcal{U}; \mathcal{X}]_s$;
(3) if $o(\vec{u}_1.\vec{x}_1.a_1, \ldots, \vec{u}_n.\vec{x}_n.a_n) \in \mathcal{B}[\mathcal{U}; \mathcal{X}]_s$ and $\rho_i : \vec{u}_i; \vec{x}_i \leftrightarrow \vec{u}'_i; \vec{x}'_i$ is a fresh renaming for each $1 \leq i \leq n$, then $[\rho_i]a_i \in \mathcal{B}[\mathcal{U}, \vec{u}'_i; \mathcal{X}, \vec{x}'_i]$. ◂

Remark 2.4.8. The families $\mathcal{U}$ of symbols and $\mathcal{X}$ of variables in the above definition are not fixed, but vary according to the symbols and variables introduced by operators' abstraction operations.

Remark 2.4.9. The coinductive (greatest fixed point) definition of general binding tree diverges from the inductive (least fixed point) definition of abstract binding trees to allow for *infinite* trees. This change lets us form judgments about infinite objects below. We can recover an inductive definition by using least families and by replacing condition 3 of definition 2.4.7 with

(3') for each operator o of arity $(\vec{s}_1.\vec{s}'_1.s_1, \ldots, \vec{s}_n.\vec{s}'_n.s_n)s$, if $[\rho_i]a_i \in \mathcal{B}[\mathcal{U}, \vec{u}'_i; \mathcal{X}, \vec{x}'_i]$ for all $1 \leq i \leq n$ and fresh renamings $\rho_i : \vec{u}_i; \vec{x}_i \leftrightarrow \vec{u}'_i; \vec{x}'_i$, then $o(\vec{u}_1.\vec{x}_1.a_1, \ldots, \vec{u}_n.\vec{x}_n.a_n) \in \mathcal{B}[\mathcal{U}; \mathcal{X}]_s$.

Though we do not pursue it here, it would be interesting to see how to combine sorts whose trees can be inductively or coinductively defined.

Every abstract binding tree can be seen as a general binding tree with no symbols. We always identify general binding trees up to α-equivalence:

Definition 2.4.10. **α-Equivalence** on general binding trees is the strongest congruence $\equiv_\alpha$ such that

(1) $x \equiv_\alpha x$ for all variables x;
(2) $u \equiv_\alpha u$ for all symbols u;
(3) $o(\vec{u}_1.\vec{x}_1.a_1, \ldots, \vec{u}_n.\vec{x}_n.a_n) \equiv_\alpha o(\vec{u}'_1.\vec{x}'_1.a'_1, \ldots, \vec{u}'_n.\vec{x}'_n.a'_n)$ if for every $1 \leq i \leq n$, $[\rho_i]a_i \equiv_\alpha [\rho'_i]a'_i$ for all fresh renamings $\rho_i : \vec{u}_i, \vec{x}_i \leftrightarrow \vec{w}_i, \vec{z}_i$ and $\rho'_i : \vec{u}'_i, \vec{x}'_i \leftrightarrow \vec{w}_i, \vec{z}_i$. ◂

Definition 2.4.11. A **morphism of gbts** $\sigma : \mathcal{B}[\mathcal{U}; \mathcal{X}] \rightsquigarrow \mathcal{B}[\mathcal{V}; \mathcal{Y}]$ is:

(1) a choice $\sigma(x) \in \mathcal{B}[\mathcal{V}; \mathcal{Y}]$ for each variable $x \in \mathcal{X}$ such that if $x \in \mathcal{X}_s$, then $\sigma(x) \in \mathcal{B}[\mathcal{V}; \mathcal{X}]_s$ is of the same sort; and
(2) a choice $\sigma(u) \in \mathcal{V}$ for each symbol $u \in \mathcal{U}$ such that if $u \in \mathcal{U}_s$, then $\sigma(u) \in \mathcal{V}_s$ is of the same sort. ◂

Definition 2.4.12. Given a morphism $\sigma : \mathcal{B}[\mathcal{U}; \mathcal{X}] \rightsquigarrow \mathcal{B}[\mathcal{V}; \mathcal{Y}]$ and a gbt $a \in \mathcal{B}[\mathcal{U}; \mathcal{X}]$, the gbt $[\sigma]a \in \mathcal{B}[\mathcal{V}; \mathcal{Y}]$ is given by **simultaneous capture-avoiding substitution** of $\sigma(x)$ for x in a, a procedure recursively defined on the structure of a by:

(1) $[\sigma]x = \sigma(x)$ for variables x;
(2) $[\sigma]u = \sigma(u)$ for symbols u;
(3) $[\sigma](o(\vec{u}_1.\vec{x}_1.a_1, \ldots, \vec{u}_n.\vec{x}_n.a_n)) = o(\vec{u}_1.\vec{x}_1.[\sigma]a_1, \ldots, \vec{u}_n.\vec{x}_n.[\sigma]a_n)$ where we assume without loss of generality that $\mathcal{X} \cap \vec{x}_i = \varnothing$ and $\mathcal{U} \cap \vec{u}_i = \varnothing$ for all $1 \leq i \leq n$. ◂

2.5. Inductively and Coinductively Defined Judgments

We introduce substructural parametric generic judgments on general binding trees. This work closely follows and synthesizes Harper [Har16, chap. 3] and Sangiorgi [San12, chap. 2] with two important differences. First, we take a substructural approach to derivation instead of a structural approach. This allows for clearer explanations of various constructions that appear later in this dissertation. Moreover, we lose nothing by taking a substructural approach: the structural approach is a special case. Second, instead of forming judgments about abstract binding trees, we form judgments about general binding trees. Generic judgments are families of hypothetical judgments that are closed under substitutions, while parametric judgments are families closed under renamings of symbols.

2.5.1. Judgments, Generally. A **judgment** is an object of knowledge [Mar96], and we say that a judgment **holds** if we deem it to be true. We range over judgments using the metavariable J. In this thesis, we will only concern ourselves with knowledge about general binding trees. For example, if we consider general binding trees for λ-terms given by example 2.4.4, then judgments we might want to consider include $M : \tau$ (term M has type τ), $M\ \mathsf{val}$ (term M is a value), or $M \to M'$ (term M steps to term M'). We call these kinds of judgments—judgments that are not composed of other judgments—basic judgments, and we let K and L range over them in this section. Formally:

Definition 2.5.1. Fix some family of gbts $\mathcal{B}[\mathcal{U};\mathcal{X}]$. **Basic judgments** over $\mathcal{B}[\mathcal{U};\mathcal{X}]$ are gbts of sort b in a family of gbts obtained by extending $\mathcal{B}[\mathcal{U};\mathcal{X}]$ with a new sort b such that whenever o is an operator of arity $(\nu_1,\dots,\nu_n)b$, the valencies ν_i do not mention b and have no bound variables or symbols. ◀

Example 2.5.2. We encode the basic judgments $M\ \mathsf{val}$ and $M \to M'$ by modifying $\mathcal{S}$ and $\mathcal{O}$ of example 2.4.4 as follows. We first extend the family $\mathcal{S}$ with a new sort b of basic judgments: $\mathcal{S} = \{t, b\}$. Next, we extend the sort-indexed family $\mathcal{O}$ to have the following unary and binary judgment formers:

$$\mathcal{O}_{(t)b} = \{\mathsf{val}\},$$
$$\mathcal{O}_{(t,t)b} = \{\mathsf{step}\},$$

where we treat $M\ \mathsf{val}$ and $M \to M'$ as concrete syntax for the gdts $\mathsf{val}(M)$ and $\mathsf{step}(M, M')$. The basic judgments about the terms of example 2.4.4 are general binding trees of sort b, i.e., elements of $\mathcal{B}[\mathcal{U};\mathcal{X}]_b$. ◀

2.5.2. Inductive and Coinductive Definitions. In our setting, we are interested in judgments whose truth is inductively or coinductively defined by a collection of inference rules. Inference rules specify sufficient conditions for a judgment to hold, and they are often schematically depicted as follows:

$$\frac{J_1 \quad \cdots \quad J_k}{J}$$

The (ordered) judgments $J_1,\dots,J_k$ are called the **premisses** of the rule, while J is its **conclusion**. Intuitively, the premisses of a rule are sufficient conditions for its conclusion: if $J_1,\dots,J_k$ hold, then we can infer J also holds. A rule with no premisses is called an **axiom**. For ease of reference, we often give rules a name, which we put to the right of the rule in brackets, e.g.,

$$\frac{J_1 \quad \cdots \quad J_k}{J}\ (\textsc{ExampleRule})$$

Formally, an **inference rule** over some set of judgments $\mathcal{J}$ is the set of all ground rules satisfying its schemata. A **ground rule** is a pair (P, C) where $P \subseteq \mathcal{J}$ is a set of basic judgments that are the premisses of the ground rule, and $C \in \mathcal{J}$ is its conclusion. Given a collection $\mathcal{R}$ of inference rules, we pun and also write $\mathcal{R}$ for the union of its inference rules viewed as sets of ground rules.

We say that a set T of judgments is **closed under** a set of rules $\mathcal{R}$ if for all $(P, C) \in \mathcal{R}$, if $P \subseteq T$, then $C \in T$. We say that a set of judgments is **inductively defined** by a set of rules $\mathcal{R}$ if it is the least collection of judgments closed under $\mathcal{R}$. To make this precise, we observe that each set $\mathcal{R}$ of ground rules defines a monotone **rule functional** $\Phi_{\mathcal{R}}$ on the complete lattice $\wp(\mathcal{J})$ of sets of judgments:

$$\Phi_{\mathcal{R}}(T) = \{C \mid \exists (P, C) \in \mathcal{R}.P \subseteq T\}.$$

We say that a family of judgments is inductively defined by the set $\mathcal{R}$ of rules if it is the least fixed point $\mathrm{lfp}(\Phi_{\mathcal{R}})$ of the functional $\mathcal{R}$, and that it is **coinductively defined** if it is the greatest fixed point $\mathrm{gfp}(\Phi_{\mathcal{R}})$. These least and greatest fixed points exist by the Knaster-Tarski theorem (theorem 2.2.7). The functional $\Phi_{\mathcal{R}}$ is ω-continuous by [San12, Exercise 2.9.2], so the least fixed point can be constructed by theorem 2.2.9. If the set $\{P \mid (P, C) \in \mathcal{R}\}$ is finite for all C, then $\Phi_{\mathcal{R}}$ is ω-cocontinuous by [San12, Theorem 2.9.4] and the greatest fixed point can also be constructed by theorem 2.2.9.

Example 2.5.3. Consider the following reduction rule for the λ-calculus:

$$\frac{M \to M'}{MN \to M'N}$$

Formally, this reduction rule stands for the following inference rule schemata, where the terms M, M', and N are implicitly universally quantified meta-variables:

$$\frac{\mathsf{step}(M, M')}{\mathsf{step}(\mathsf{app}(M, N), \mathsf{app}(M', N))}$$

The corresponding inference rule is:

$$\{(\{\mathsf{step}(M, M')\}, \mathsf{step}(\mathsf{app}(M, N), \mathsf{app}(M', N))) \mid M, M', N \in \mathcal{B}[\mathcal{U}; \mathcal{X}]_t\}\,.$$

Alone, this inference rule inductively defines an empty set of judgments (consider the action of the induced functional on the bottom element $\varnothing$ of $\wp(\mathcal{J})$ and apply theorem 2.2.9). ◂

2.5.3. Derivations. A **derivation** of a judgment J using a collection of rules $\mathcal{R}$ is a well-founded tree (usually assumed to be finite) constructed by composing rules from $\mathcal{R}$, with J at the bottom and axioms at its leaves. For example, if $\mathcal{D}_1, \ldots, \mathcal{D}_k$ are derivations of $J_1, \ldots, J_k$, then the following tree is a derivation of J using (ExampleRule):

$$\frac{\mathcal{D}_1 \quad \cdots \quad \mathcal{D}_k}{J}$$

We say that a judgment is **derivable** if it has a derivation. We write $\blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ to mean that J is derivable using $\mathcal{R}$, and that it is a gbt using symbols $\mathcal{U}$ and variables $\mathcal{X}$. By [San12, Theorem 2.11.2], a collection of judgments is inductively defined by $\mathcal{R}$ if and only if each judgment has a finite derivation built from rules in $\mathcal{R}$.

2.5.3.1. *Hypothetical Derivations.* In practice, we would like to study derivability while assuming certain hypotheses, instead of requiring that all leaves in a derivation be axioms. A **hypothetical derivation** is a derivation where leaves are also allowed to be judgments. Assuming some ordered collection of judgments $J_1, \ldots, J_k$, we write $J_1, \ldots, J_k \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ if there exists a derivation of J where all leaves are axioms, except for $J_1, \ldots, J_k$, which appear as leaves (from left-to-right) in the derivation. We often abbreviate collections of hypotheses, called **contexts**, using Γ or other capital Greek letters. The judgment $\Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ is called **hypothetical derivability**.

Hypothetical derivability satisfies the following structural properties by definition:

(1) **reflexivity**: each judgment is a consequence of itself: $J \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ for each judgment J;
(2) **transitivity**: we can compose derivations: if $\Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J'$ and $\Delta, J', \Sigma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, then $\Delta, \Gamma, \Sigma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$.

Example 2.5.4. Consider judgments a, b, c, d governed by the rules

$$\frac{a \quad c}{b} \qquad \frac{d \quad c}{a} \qquad \frac{c \quad b}{d} \qquad \frac{}{c}$$

Then the following hypothetical derivation justifies $c, b, c \blacktriangleright b$:

$$\cfrac{\cfrac{c \quad b}{d} \quad \cfrac{}{c}}{\cfrac{a \quad c}{b}}$$

We remark that it *does not* justify $c, c, b \blacktriangleright b$, or $c, b \blacktriangleright b$, or anything else. ◂

This treatment of derivation is extremely restrictive, and often we would like to let hypotheses go unused or let them be used repeatedly. Conditions on the usage of hypotheses are called **structural properties** and they include:

(1) **exchange**: hypotheses can be used in any order: if $\Gamma, J_1, J_2, \Gamma' \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, then $\Gamma, J_2, J_1, \Gamma' \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$;

(2) **weakening**: hypotheses can go unused: if $\Gamma, \Gamma' \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, then $\Gamma, J', \Gamma' \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$;
(3) **contraction**: hypotheses can be used arbitrarily often: if $\Gamma, J', J', \Gamma' \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, then $\Gamma, J', \Gamma' \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$.

We say that hypothetical derivability is **structural** if it satisfies exchange, weakening, and contraction, and that it is **substructural** otherwise. We say that it is **linear** if it allows exchange, but not weakening or contraction. Linearity ensures that each hypotheses is used exactly once.

2.5.3.2. *Generic Derivations.* The meaning of variables is given by substitution, and we would like derivations to respect this. In particular, if we designate certain variables $\mathcal{Y}$ as *generic*, then a **generic derivation** is a derivation that is invariant under renaming or substitution for these variables. Explicitly, generic derivability $\mathcal{Y} \mid \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ is defined to hold if and only if $\Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}\mathcal{Y}} J$, and the rules $\mathcal{R}$ are interpreted such that generic derivability enjoys the following structural properties:

(1) **proliferation**: closure under the expansion of the universe: if $\mathcal{Y} \mid \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, and $\mathcal{Y}'$ is fresh for $\mathcal{X}, \mathcal{Y}$, then $\mathcal{Y}, \mathcal{Y}' \mid \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$;
(2) **renaming**: closure under renaming: if $\mathcal{Y} \mid \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, and $\rho : \mathcal{Y} \leftrightarrow \mathcal{Y}'$ is a fresh renaming, then $\mathcal{Y}' \mid [\rho]\Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} [\rho]J$;
(3) **substitution**: closure under substitution: if $\mathcal{Y} \mid \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ and $\sigma : \mathcal{U}; \mathcal{X}, \mathcal{Y} \rightsquigarrow \mathcal{U}; \mathcal{Y}'$ is a morphism fixing $\mathcal{U}$ and $\mathcal{X}$, then $\mathcal{Y}' \mid \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} [\sigma]J$.

We remark that renaming is a special case of substitution.

2.5.3.3. *Parametric Derivations.* Just as generic derivations are stable under renaming or substitution of generic variables, **parametric derivations** are stable under renaming of symbols deemed *parametric*. Explicitly, parametric derivability $\mathcal{V} \parallel \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$ is defined to hold if and only if $\Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U},\mathcal{V};\mathcal{X}} J$, and the rules $\mathcal{R}$ are interpreted such that parametric derivability enjoys the following properties:

(1) **proliferation**: closure under the expansion of the universe: if $\mathcal{U} \parallel \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, and $\mathcal{V}''$ is fresh for $\mathcal{U}, \mathcal{V}$, then $\mathcal{V}, \mathcal{V}' \parallel \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$;
(2) **renaming**: closure under renaming: if $\mathcal{V} \parallel \Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J$, and $\rho : \mathcal{V} \leftrightarrow \mathcal{V}'$ is a fresh renaming, then $\mathcal{V}' \parallel [\rho]\Gamma \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} [\rho]J$.

Remark 2.5.5. Judgments can be both generic and parametric. These parametric generic judgments are notated by "stacking" the syntax for generic and parametric judgments: $\mathcal{V} \parallel \mathcal{Y} \mid J$. We will revisit these parametric generic judgments in section 2.5.7 when we discuss type systems for programs that use both variables and symbols.

2.5.4. Hypothetical Judgments. Hypothetical judgments internalize the notion of hypothetical derivability, and they let us inductively define judgments that are subject to hypotheses. A **hypothetical judgment** $K_1, \ldots, K_k \vdash L$ means that we can derive the basic judgment L assuming the ordered collection of basic judgments $K_1, \ldots, K_k$. It is the formal analog of the hypothetical derivability judgment $K_1, \ldots, K_k \blacktriangleright_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} L$. As with hypothetical derivability, **structural properties** govern the use of hypotheses in contexts in hypothetical judgments:

(1) **exchange**: hypotheses can be used in any order: if $\Gamma, K, K', \Gamma' \vdash L$, then $\Gamma, K', K, \Gamma' \vdash L$;
(2) **weakening**: hypotheses can go unused: if $\Gamma, \Gamma' \vdash L$, then $\Gamma, K, \Gamma' \vdash L$;
(3) **contraction**: hypotheses can be used arbitrarily often: if $\Gamma, K, K, \Gamma' \vdash L$, then $\Gamma, K, \Gamma' \vdash L$.

We say that a context in a hypothetical judgment is **structural** if it satisfies exchange, weakening, and contraction, and that it is **substructural** otherwise. We say that it is **linear** if it allows exchange, but not weakening or contraction.

Sometimes, we would like to have multiple contexts of hypotheses, each governed by its own structural properties. For example, in part 2 we will encounter a hypothetical judgment $\Psi \,;\, \Delta \vdash P :: c : A$ for typing processes, where the context Ψ of "functional variables" is structural and the context Δ of "channel names" is linear. The reader is referred to [Walo5] for more details on structural properties and on substructural type systems.

As with basic judgments, we often want to inductively define a collection of hypothetical judgments. We do so by a collection $\mathcal{R}$ of **hypothetical rules** schematically depicted as

$$\frac{\Gamma\Gamma_1 \vdash K_1 \quad \cdots \quad \Gamma\Gamma_k \vdash K_k}{\Gamma \vdash L}$$

where Γ contains *global* hypotheses and the Γ_i contain *local* hypotheses. Hypothetical rules refine inference rules to ensure that the hypothetical judgments they define faithfully capture the notion of derivation under hypotheses. In particular, hypothetical rules are **uniform**, i.e., their contexts of global hypotheses Γ are implicitly universally quantified. We also require that collections of inductively defined hypothetical judgments satisfy the following structural properties:

(1) **reflexivity**: each judgment is a consequence of itself: $K \vdash K$ for each judgment K;
(2) **transitivity**: we can compose derivations: if $\Gamma \vdash K$ and $\Delta, K, \Sigma \vdash L$, then $\Delta, \Gamma, \Sigma \vdash L$.

These two structural properties and the three properties governing the use of contexts respectively correspond to the following uniform inference rules:

$$\frac{}{K \vdash K} \qquad \frac{\Gamma \vdash K \quad \Delta, K, \Sigma \vdash L}{\Delta, \Gamma, \Sigma \vdash L} \qquad \frac{\Gamma, K', K, \Gamma' \vdash L}{\Gamma, K, K', \Gamma' \vdash L} \qquad \frac{\Gamma, \Gamma' \vdash L}{\Gamma, K, \Gamma' \vdash L} \qquad \frac{\Gamma, K, K, \Gamma' \vdash L}{\Gamma, K, \Gamma' \vdash L}$$

We say that a collection of hypothetical judgments $\Gamma \vdash L$ is **inductively defined** by $\mathcal{R}$ if it is the least collection of hypothetical judgments closed under $\mathcal{R}$, the rules for the two structural properties, and the rules for the desired structural properties governing Γ. The above definitions extends straightforwardly to hypothetical judgments $\Gamma_1; \ldots; \Gamma_n \vdash L$ with multiple contexts.

2.5.5. Generic Judgments. Just as hypothetical judgments internalize hypothetical derivability, generic judgments internalize generic derivability. A formal **generic judgment** $\mathcal{Y} \mid J$ specifies a family of judgments closed under substitution for the variables $\mathcal{Y}$. Generic judgments $\mathcal{Y} \mid J$ are identified up-to-renaming of the variables $\mathcal{Y}$, so that we identify $\mathcal{Y} \mid J$ and $\mathcal{Y}' \mid [\rho]J$ for any renaming $\rho\colon \mathcal{Y} \leftrightarrow \mathcal{Y}'$. For a generic judgment $\mathcal{Y} \mid J$ to be well formed, we assume that the variables in $\mathcal{Y}$ are pairwise distinct. The judgment J can refer both to variables in $\mathcal{Y}$ and others.

Generic judgments can be inductively defined using generic rules. **Generic rules** are of the form:

$$\frac{\mathcal{Y}\mathcal{Y}_1 \mid J_1 \quad \cdots \quad \mathcal{Y}\mathcal{Y}_k \mid J_k}{\mathcal{Y} \mid J}$$

The variables $\mathcal{Y}$ are the *global* variables, while the $\mathcal{Y}_i$ are the *local* variables. A collection of generic judgments is **inductively defined** by a collection $\mathcal{R}$ of rules if it is the least collection of judgments closed under $\mathcal{R}$ and the following structural rules capturing proliferation and substitution:

$$\frac{\mathcal{Y} \mid \Gamma \vdash J}{\mathcal{Y}, \mathcal{Y}' \mid \Gamma \vdash J} \tag{3}$$

$$\frac{\mathcal{Y} \mid J}{\mathcal{Y}' \mid [\sigma]J} \tag{4}$$

The substitution rule (4) encodes both the renaming and substitution structural properties. It is universally quantified over all gbt morphisms σ fixing all symbols and fixing all variables save those in $\mathcal{Y}$.

2.5.6. Parametric Judgments. We repeat the same development for parametric judgments, which internalize parametric derivations. A formal **parametric judgment** $\mathcal{V} \parallel J$ specifies a family of judgments parametrized by the variables $\mathcal{V}$. Parametric judgments $\mathcal{V} \parallel J$ are identified up-to-renaming of the symbols $\mathcal{V}$, so that we identify $\mathcal{V} \parallel J$ and $\mathcal{V}' \parallel [\rho]J$ for any renaming $\rho\colon \mathcal{V} \leftrightarrow \mathcal{V}'$. For a parametric judgment $\mathcal{V} \parallel J$ to be well formed, we assume that the symbols in $\mathcal{V}$ are pairwise distinct. The judgment J can refer both to symbols in $\mathcal{V}$ and others. Unlike generic judgments, which are closed under substitution, parametric judgments are only closed under renamings of symbols.

Parametric judgments can be inductively defined using parametric rules. **Parametric rules** are of the form:

$$\frac{\mathcal{V}\mathcal{V}_1 \parallel J_1 \quad \cdots \quad \mathcal{V}\mathcal{V}_k \parallel J_k}{\mathcal{V} \parallel J}$$

The symbols $\mathcal{V}$ are called *global* symbols, while the $\mathcal{V}_i$ are called *local* symbols. A collection of parametric judgments is **inductively defined** by a collection $\mathcal{R}$ of rules if it is the least collection of judgments closed under $\mathcal{R}$ and the following structural rules capturing proliferation and renaming:

$$\frac{\mathcal{V} \parallel \Gamma \vdash J}{\mathcal{V}, \mathcal{V}' \parallel \Gamma \vdash J} \tag{5}$$

$$\frac{\mathcal{V} \parallel J}{\mathcal{V}' \parallel [\rho]J} \tag{6}$$

The renaming rule (6) is universally quantified over all renamings $\rho : \mathcal{V} \leftrightarrow \mathcal{V}'$.

Remark 2.5.6. The interactions between coinductively defined derivations, and hypothetical, generic, or parametric judgments are far from clear. Though we will make extensive use of coinductively defined judgments in chapter 6, at no point will we use coinductively defined hypothetical, generic, or parametric judgments.

2.5.7. Encoding Type Systems. Typing judgments $e : \tau$ specify that an expression e has type τ. Type systems are often inductively defined using hypothetical judgments $\Gamma \vdash e : \tau$, where the context Γ is a list of hypotheses $x_1 : \tau_1, \ldots, x_n : \tau_n$ for variables x_i and types τ_i. For a type system to be well-behaved, these hypothetical judgments must in fact be interpreted as *generic hypothetical judgments* $x_1, \ldots, x_n \mid \Gamma \vdash e : \tau$ in $\mathcal{B}[x_1, \ldots, x_n]$. This ensures both that the particular choice of variable names does not matter, and that type systems are closed under substitution.

Later in part 2, we will see type systems that are encoded using *parametric generic hypothetical judgments* $c_0, \ldots, c_n \parallel x_1, \ldots, x_m \mid x_1 : \tau_1, \ldots, x_m : \tau_m \; ; \; c_1 : A_1, \ldots, c_n : A_n \vdash P :: c_0 : A_0$. There, the symbols $c_0, \ldots, c_n$ correspond to channel names, and we interpret the judgment as a parametric judgment to ensure that these channel names can be freely varied. The variabes $x_1, \ldots, x_m$ correspond to usual variables in a functional language, and we interpret the judgment as generic in these to ensure that the judgments respect substitution. Again, we assume that the only free symbols and variables appearing in these parametric typing judgments are those in which the judgment is parametric or generic, respectively.

In practice, we elide the sequences of parametric symbols and generic variables in typing judgments. Despite this elision, typing judgments should always be interpreted as generic in their free variables and parametric in their free symbols.

Type systems of this form enjoy a typed notion of substitution, called a "context morphism", which refines morphisms of abts or gbts:

Definition 2.5.7. Fix typing contexts $\Gamma = y_1 : \tau_1, \ldots, y_m : \tau_m$ and $\Gamma' = x_1 : \tau'_1, \ldots, x_n : \tau'_n$. A **context morphism** $\sigma : \Gamma \rightsquigarrow \Gamma'$ is a morphism of (g)abts $\sigma : \mathcal{B}[\mathcal{U}; x_1, \ldots, x_n] \rightsquigarrow \mathcal{B}[\mathcal{U}; y_1, \ldots, y_m]$ such that $y_1, \ldots, y_m \mid \Gamma \vdash \sigma(x_i) : \tau_i$ for all $1 \leq i \leq n$.

A context morphism $\sigma : \Gamma \rightsquigarrow \Gamma'$ acts on typing judgments over Γ' by substitution: if $\vec{x} \mid \Gamma' \vdash e : \tau$, then $\vec{y} \mid \Gamma \vdash [\sigma]e : \tau$. ◂

2.5.8. Modes of Use. Judgments can be thought of as n-ary relations over gbts. It is often useful to think as some components as inputs and others as outputs of the judgment. We indicate these **modes** of use using colours, where we designate inputs in blue and outputs in red.

Example 2.5.8. Bidirectional type-checking [PT00; DK19] is given by two judgments: $\Gamma \vdash e \Leftarrow \tau$ means that e *checks* agains the type τ under the context Γ, while $\Gamma \vdash e \Rightarrow \tau$ means that e *synthesizes* the type τ under Γ. These two judgments are inductively defined, and they describe mutually recursive functions whose modes correspond to the inputs and outputs of the functions. ◂

CHAPTER 3

Fairness for Multiset Rewriting Systems

Session-typed languages are often defined using multiset rewriting systems (MRS). In chapter 6, we give an observed communications semantics for Polarized SILL. To ensure that it is well-defined in the presence of non-termination, we require that process executions be *fair*. Intuitively, fairness ensures that if a process can make progress, then it eventually does so. To this end, we introduce and study *fairness* for multiset rewriting systems.

We begin by reviewing multiset rewriting systems and their relation to linear logic in section 3.1. In section 3.2 we introduce three different varieties of fairness, each of which subdivides along the axis of weak and strong fairness. We study properties of fair traces in section 3.3. In particular, we construct a scheduler, we give sufficient conditions for traces to be fair, and we study the effects of permutations on traces. We also introduce a novel notion of trace equivalence, called *union equivalence*, that will be essential to the development of the observed communication semantics of chapter 6.

This chapter builds on work presented at EXPRESS/SOS 2020 [Kav20a].

3.1. Multiset Rewriting Systems

In this section, we review (first-order) multiset rewriting systems. For expository reasons, we start with the simpler formalism MSR_1 of Cervesato and Scedrov [CS09] in section 3.1.1. We extend it in section 3.1.2 to handle the *persistency* features of the multiset rewriting system MSR of [Cer+05]. These first sections are expository, and they serve solely to give a uniform presentation to pre-existing work. Our contribution comes in section 3.1.4, where we extend parallel multiset rewriting [Cer01, §§ 5.3–5.4] to support persistency.

Definition 3.1.1. A **multiset** M is a pair (S, m) where S is a set (the *underlying set*) and $m : S \to \mathbb{N}$ is a function. It is finite if $\sum_{s \in S} m(s)$ is finite. We say s is an **element** of M, $s \in M$, if $m(s) > 0$. The **support** of M is the set $\mathsf{supp}(M) = \{s \in S \mid s \in M\}$. ◂

Remark 3.1.2. A multiset with a finite underlying set is always finite. The converse is false: the multiset $(\mathbb{N}, \lambda x \in \mathbb{N}.0)$ is finite, but the underlying set $\mathbb{N}$ is infinite.

When considering several multisets at once, we assume without loss of generality that they have equal underlying sets.

Definition 3.1.3. Multisets $M_1 = (S, m_1)$ and $M_2 = (S, m_2)$ are equipped with the following operations and relations:

(1) the **sum** of M_1 and M_2 is the multiset $M_1, M_2 = (S, \lambda s \in S. m_1(s) + m_2(s))$;
(2) the **union** of M_1 and M_2 is the multiset $M_1 \cup M_2 = (S, \lambda s \in S. \max(m_1(s), m_2(s)))$;
(3) the **intersection** of M_1 and M_2 is the multiset $M_1 \cap M_2 = (S, \lambda s \in S. \min(m_1(s), m_2(s)))$;
(4) the **difference** of M_1 and M_2 is the multiset $M_1 \setminus M_2 = (S, \lambda s \in S. \max(0, m_1(s) - m_2(s)))$;
(5) M_1 is **included** in M_2, written $M_1 \subseteq M_2$, if $m_1(s) \leq m_2(s)$ for all $s \in S$. ◂

We abuse terminology and call multisets (S, m) *sets* if $m(s) \leq 1$ for all $s \in S$. We write $\varnothing$ for **empty** multisets, i.e., for multisets (S, m) such that $m(s) = 0$ for all $s \in S$.

Example 3.1.4. A finite string s over an alphabet Σ describes a multiset (Σ, m), where $m(\sigma)$ is the multiplicity of σ in s. ◂

3.1.1. First-Order Multiset Rewriting. Consider finite multisets of first-order atomic formulas over some fixed initial signature Σ_i. We call formulas **facts**. We write $M(\vec{x})$ to mean that the facts in the multiset M draw their variables from $\vec{x}$, where $\vec{x} = x_1, \ldots, x_m$ for some m. Given $M(\vec{x})$ and some choice of terms $\vec{t}$ for $\vec{x}$, we write $M(\vec{t})$ for the simultaneous substitution $[\vec{t}/\vec{x}]M$.

Multiset rewrite rules describe localized changes to multisets of facts. A *multiset rewrite rule* r is a pair of multisets $F(\vec{x})$ and $G(\vec{x}, \vec{n})$, and it is schematically represented by:

$$r : \forall \vec{x}.F(\vec{x}) \to \exists \vec{n}.G(\vec{x}, \vec{n}).$$

Informally, we interpret the variables $\vec{x}$ as being universally quantified in F and G, and the variables $\vec{n}$ as being existentially quantified in G. In particular, we treat $\vec{x}$ and $\vec{n}$ as bound variables, and assume that they can be freely α-varied. We will make these intuitions precise below when we relate multiset rewriting systems to linear logic. A *multiset rewriting system* (MRS) is a set $\mathcal{R}$ of multiset rewrite rules.

Given a rule $r : \forall \vec{x}.F(\vec{x}) \to \exists \vec{n}.G(\vec{x}, \vec{n})$ in $\mathcal{R}$ and some choice of constants $\vec{t}$ for $\vec{x}$, we say that the *instantiation* $r(\vec{t}) : F(\vec{t}) \to \exists \vec{n}.G(\vec{t}, \vec{n})$ is *applicable* to a multiset M if there exists a multiset M' such that $M = F(\vec{t}), M'$. The rule r is applicable to M if $r(\vec{t})$ is applicable to M for some $\vec{t}$. In these cases, the *result* of applying $r(\vec{t})$ to M is the multiset $G(\vec{t}, \vec{d}), M'$, where $\vec{d}$ is a choice of pairwise-distinct fresh constants. In particular, we assume that the constants $\vec{d}$ do not appear in M or in $\mathcal{R}$. We call $\theta = [\vec{t}/\vec{x}]$ the **matching substitution** and $\xi = [\vec{d}/\vec{n}]$ the **fresh-constant substitution**. Intuitively, the matching substitution specifies to which portion of M the rule r is applied. The **instantiating substitution** for r relative to M is the composite substitution $\delta = (\theta, \xi)$. We capture this relation using the syntax

$$F(\vec{t}), M' \xrightarrow{(r;\delta)} G(\vec{t}, \vec{d}), M'.$$

For conciseness, we often abuse notation and write $r(\theta)$, $F(\theta)$, and $G(\theta, \xi)$ for $r(\vec{t})$, $F(\vec{t})$, and $G(\vec{t}, \vec{d})$. We call $F(\vec{t})$ the *active* multiset and M' the *stationary* multiset.

Definition 3.1.5. Given an MRS $\mathcal{R}$ and a multiset M_0, a **trace** from M_0 is a countable sequence of steps

$$M_0 \xrightarrow{(r_1;\delta_1)} M_1 \xrightarrow{(r_2;\delta_2)} M_2 \xrightarrow{(r_3;\delta_3)} \cdots \tag{7}$$

such that, where $\delta_i = (\theta_i, \xi_i)$, the constants in M_i and ξ_j are disjoint for all $i < j$.

The notation $(M_0, (r_i; \delta_i)_{i \in I})$ abbreviates the trace (7), where I always ranges over $\mathbb{N}$ or $\mathbf{n} = \{1, \ldots, n\}$ for some $n \in \mathbb{N}$. An **execution** is a maximally long trace. ◂

Let the **support** $\mathsf{supp}(T)$ of a trace $T = (M_0; (r_i, \delta_i)_I)$ be the set $\mathsf{supp}(T) = \bigcup_{i \geq 0} \mathsf{supp}(M_i)$. Write $M \to M'$ if $M \xrightarrow{(r,\theta)} M'$ for some (r, θ), and let $\to^*$ be the reflexive, transitive closure of $\to$.

Example 3.1.6. We model computations with queues. Let the fact $\mathsf{queue}(q, \$)$ mean that q is the empty queue, and let $\mathsf{queue}(q, v \rightsquigarrow q')$ mean that the queue q has value v at its head and that its tail is the queue q'. Then the multiset $Q = \mathsf{queue}(q, \mathsf{o} \rightsquigarrow q'), \mathsf{queue}(q', \$)$ describes a one-element queue containing o. The following two rules capture enqueuing values on empty and non-empty queues, respectively, where the fact $\mathsf{enq}(q, v)$ is used to enqueue v onto the queue q:

$$e_1 : \forall x, y.\mathsf{enq}(x, y), \mathsf{queue}(x, \$) \to \exists z.\mathsf{queue}(x, y \rightsquigarrow z), \mathsf{queue}(z, \$),$$
$$e_2 : \forall x, y, z, w.\mathsf{enq}(x, y), \mathsf{queue}(x, z \rightsquigarrow w) \to \mathsf{queue}(x, z \rightsquigarrow w), \mathsf{enq}(w, y).$$

The following execution from $Q, \mathsf{enq}(q, 1)$ captures enqueuing 1 on the queue q:

$$Q, \mathsf{enq}(q, 1) \xrightarrow{(e_2;([q,1,\mathsf{o},q'/x,y,z,w],\varnothing))} Q, \mathsf{enq}(q', 1)$$
$$\xrightarrow{(e_1;([q',1/x,y],[a/z]))} \mathsf{queue}(q, \mathsf{o} \rightsquigarrow q'), \mathsf{queue}(q', 1 \rightsquigarrow a), \mathsf{queue}(a, \$). \quad ◂$$

The substructural operational semantics of Polarized SILL is given by a multiset rewriting system, and it uses queues of messages to ensure that messages sent by processes arrive in order. These message queues are captured by a variant of example 3.1.6.

Example 3.1.7. We define addition on unary natural numbers. The MRS uses the facts $\mathsf{add}(m, n, l)$ and $\mathsf{val}(l, v)$, where $\mathsf{val}(l, v)$ represents a memory cell l with value v, and $\mathsf{add}(m, n, l)$ causes the sum of m and n to be stored in cell l. It is given by the following rules:

$$a_{\mathsf{z}} : \forall n, l.\mathsf{add}(\mathsf{z}, n, l) \to \mathsf{val}(l, n) \tag{8}$$

$$a_{\mathsf{s}} : \forall m, n, l.\mathsf{add}(\mathsf{s}(m), n, l) \to \mathsf{add}(m, \mathsf{s}(n), l) \tag{9}$$

Write 3 for the unary representation $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{z})))$ of three. The following execution stores the sum of two and three in l:

$$\mathsf{add}(\mathsf{s}(\mathsf{s}(\mathsf{z})), 3, l) \to \mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{s}(3), l) \to \mathsf{add}(\mathsf{z}, \mathsf{s}(\mathsf{s}(3)), l) \to \mathsf{val}(l, \mathsf{s}(\mathsf{s}(3))).$$

The first two rules in this execution are instances of a_{s}, while the last rule is an instance of a_{z}. ◀

Example 3.1.8. We build on example 3.1.7 to recursively compute the n-th Fibonacci number. The fact $\mathsf{fib}(n, l)$ causes the n-th Fibonacci number to be stored in cell l. The MRS is given by a_{z}, a_{s}, and the following new rules:

$$f_0 : \forall l.\mathsf{fib}(\mathsf{z}, l) \to \mathsf{val}(l, \mathsf{s}(\mathsf{z})) \tag{10}$$

$$f_1 : \forall l.\mathsf{fib}(\mathsf{s}(\mathsf{z}), l) \to \mathsf{val}(l, \mathsf{s}(\mathsf{z})) \tag{11}$$

$$f : \forall l, n.\mathsf{fib}(\mathsf{s}(\mathsf{s}(n)), l) \to \exists l', l''.\mathsf{cont}(l, l', l''), \mathsf{fib}(\mathsf{s}(n), l'), \mathsf{fib}(n, l'') \tag{12}$$

$$c : \forall l, l', l'', m, n.\mathsf{cont}(l, l', l''), \mathsf{val}(l', m), \mathsf{val}(l'', n) \to \mathsf{add}(m, n, l) \tag{13}$$

Rules f_0 and f_1 directly calculate the zeroth and first Fibonacci numbers. The rule f makes two "recursive calls" that will store their results in fresh locations l' and l''. It uses the continuation fact $\mathsf{cont}(l, l', l'')$ to signal that the values in locations l' and l'' need to be added and stored in location l. Once values m and n are available in locations l' and l'', the rule c causes m and n to be added and stored in location l.

We remark that this implementation builds in garbage collection. Indeed, because we represent memory locations $\mathsf{val}(l, n)$ using ephemeral facts, these locations are discarded as soon as they are no longer needed by future computation. However, this implementation is not very efficient: it repeatedly recomputes the same n-th Fibonacci number and requires exponential time. In example 3.1.11, we will use *persistent* facts to implement a memoized version of this algorithm. ◀

Remark 3.1.9. The order in which rewrite rules are applied is non-deterministic and is outside of the control of a multiset rewriting system. For example, when computing $\mathsf{fib}(2, l)$, a scheduler could non-deterministically choose to apply rule f_1 or f_0 after applying f_2. Moreover, multiset rewriting systems need not satisfy any confluence properties. This means that, in general, finite executions from a given multiset need not result in the same final multiset. In example 3.1.11, we illustrate design considerations for multiset rewrite systems that force a scheduler to order certain rule applications. In section 3.3, we present a condition on MRSs called "interference-freedom". Intuitively, it states that the order in which rules are applied does not matter, because rules do not interfere with or disable each other.

Though the above presentation of multiset rewriting due to Cervesato et al. [Cer+05] is relatively concise, its implicit treatment of eigenvariables and signatures introduces some ambiguity. For greater clarity and to more easily relate multiset rewriting to linear logic, we sometimes adopt an equivalent presentation due to Cervesato and Scedrov [CS09]. In this presentation, multiset rewriting systems rewrite multisets-in-context. A *multiset-in-context* is a pair $\Sigma\ ;\ M$ where the term-level symbols used by M appear in the signature Σ, and Σ contains the initial signature Σ_i. We write $\Sigma \vdash t$ to mean that the term t is valid over the signature Σ, and $\Sigma \vdash \vec{t}$ for the obvious extension to collections of terms $\vec{t}$.

Consider a rule $r : \forall \vec{x}.F(\vec{x}) \to \exists \vec{n}.G(\vec{x}, \vec{n})$. Given a signature Σ, an *instantiation* $r(\vec{t})$ of r is a rule of the form $r(\vec{t}) : F(\vec{t}) \to \exists \vec{n}.G(\vec{t}, \vec{n})$ for some $\vec{t}$ with $\Sigma \vdash \vec{t}$. This instantiation is *applicable* to a multiset-in-context $\Sigma\ ;\ M$ if $M = F(\vec{t}), M'$ for some M'. The *result* of applying the instantiation

$$\frac{}{\Gamma\,;\,\Delta \longrightarrow_{\Sigma,\Sigma'} \exists\Sigma'.\Delta}\ (\textsc{Obs}) \qquad \frac{\Gamma, A\,;\,\Delta, A \longrightarrow_\Sigma C}{\Gamma, A\,;\,\Delta \longrightarrow_\Sigma C}\ (\textsc{Clone})$$

$$\frac{\Gamma\,;\,\Delta_1 \longrightarrow_\Sigma A \quad \Gamma\,;\,\Delta_2, A \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta_1, \Delta_2 \longrightarrow_\Sigma C}\ (\textsc{Cut}) \qquad \frac{\Gamma\,;\,\cdot \longrightarrow_\Sigma A \quad \Gamma, A\,;\,\Delta \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta \longrightarrow_\Sigma C}\ (\textsc{Cut}!)$$

$$\frac{\Gamma\,;\,\Delta \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta, \mathbf{1} \longrightarrow_\Sigma C}\ (\mathbf{1}\textsc{l}) \qquad \frac{\Gamma\,;\,\Delta, A_1, A_2 \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta, A_1 \otimes A_2 \longrightarrow_\Sigma C}\ (\otimes\textsc{l})$$

$$\frac{\Gamma\,;\,\Delta, B \longrightarrow_{\Sigma,\Sigma'} C}{\Gamma\,;\,\Delta, \Delta', (\exists\Sigma'.\Delta') \multimap B \longrightarrow_{\Sigma,\Sigma'} C}\ (\multimap'\textsc{l}) \qquad \frac{\Gamma\,;\,\Delta, A_i \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta, A_1 \,\&\, A_2 \longrightarrow_\Sigma C}\ (\&\textsc{l}_i)$$

$$\frac{\Gamma, A\,;\,\Delta \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta, !\,A \longrightarrow_\Sigma C}\ (!\textsc{l}) \qquad \frac{\Sigma \vdash t \quad \Gamma\,;\,\Delta, [t/x]A \longrightarrow_\Sigma C}{\Gamma\,;\,\Delta, \forall x.A \longrightarrow_\Sigma C}\ (\forall\textsc{l}) \qquad \frac{\Gamma\,;\,\Delta, A \longrightarrow_{\Sigma,x} C}{\Gamma\,;\,\Delta, \exists x.A \longrightarrow_\Sigma C}\ (\exists\textsc{l})$$

FIGURE 3.1. $\mathsf{LV}^{\mathsf{obs}}$ sequent presentation of intuitionistic linear logic [CS09, Fig. 3]

$r(\vec{t})$ to $\Sigma\,;\,M$ is the multiset-in-context $\Sigma, \vec{n}\,;\,G(\vec{t}), M'$, where we extend the signature Σ with the globally fresh symbols[1] $\vec{n}$ (modulo α-renaming). We can represent this transition schematically as:

$$\Sigma\,;\,F(\vec{t}), M' \longrightarrow_{\mathcal{R}, r:\forall\vec{x}.F(\vec{x})\to\exists\vec{n}.G(\vec{x},\vec{n})} \Sigma, \vec{n}\,;\,G(\vec{t}), M' \quad \text{if } \Sigma \vdash \vec{t}.$$

Here, $\mathcal{R}$ lists the other rules in the multiset rewriting system, and the substitution $[\vec{t}/\vec{x}]$ applied to F and G corresponds to the matching substitution. Extending the signature Σ with $\vec{n}$ captures the fresh-constant substitution $[\vec{n}/\vec{n}]$.

Given an MRS $\mathcal{R}$ and a multiset-in-context $\Sigma_0\,;\,M_0$, a **trace** from $\Sigma_0\,;\,M_0$ is a countable sequence of applications $\Sigma_0\,;\,M_0 \longrightarrow_{\mathcal{R}} \Sigma_0, \Sigma_1\,;\,M_1 \longrightarrow_{\mathcal{R}} \Sigma_0, \Sigma_1, \Sigma_2\,;\,M_2 \longrightarrow \cdots$. Again, an **execution** is a maximally long trace.

3.1.1.1. *Relation to Linear Logic.* We relate first-order multiset rewriting to the $\mathsf{LV}^{\mathsf{obs}}$ fragment of intuitionistic linear logic of Cervesato and Scedrov [CS09]. This fragment is equivalent to a fragment of the intuitionistic version of Pfenning's [Pfe95] sequent calculus presentation LV of linear logic. The fragment $\mathsf{LV}^{\mathsf{obs}}$ uses sequents of the form $\Gamma\,;\,\Delta \longrightarrow_\Sigma C$. Here Γ is a structural context of reusable assumptions, Δ is a linear context of assumptions, and C is the goal formula. The term-level symbols appearing in Γ, Δ, and C are listed in the signature Σ. The rules are reproduced in fig. 3.1. Where Δ is a context, we use the syntax $\exists\Sigma.\Delta$ to mean $\exists\Sigma.\bigotimes\Delta$, where

$$\bigotimes(\cdot) = \mathbf{1}, \qquad \bigotimes(A, \Delta) = A \otimes (\bigotimes\Delta),$$
$$\exists(\cdot).C = C, \qquad \exists(x, \Sigma).C = \exists x.\exists\Sigma.C.$$

This sequent calculus enjoys cut-elimination, i.e., the rules (Cut) and (Cut!) are both admissible [CS09, Lemmas 2.13 and 2.14] and every derivable sequent in $\mathsf{LV}^{\mathsf{obs}}$ has a cut-free derivation in $\mathsf{LV}^{\mathsf{obs}}$ [CS09, Theorem 2.15].

We interpret multiset rewriting into $\mathsf{LV}^{\mathsf{obs}}$ using the following homomorphic mapping $\ulcorner\cdot\urcorner$ that takes multisets and rules to logical formulas:

$$\ulcorner\varnothing\urcorner = \mathbf{1}$$
$$\ulcorner M, s\urcorner = \ulcorner M\urcorner \otimes s$$
$$\ulcorner\forall\vec{x}.F(\vec{x}) \to \exists\vec{n}.G(\vec{x},\vec{n})\urcorner = \forall\vec{x}.\ulcorner F\urcorner \multimap \exists\vec{n}.\ulcorner G\urcorner$$

We extend the homomorphism to map multiset rewriting systems to contexts of formulas, where $\ulcorner\varnothing\urcorner = \cdot$ and $\ulcorner\mathcal{R}, r\urcorner = \ulcorner\mathcal{R}\urcorner, \ulcorner r\urcorner$.

The following result specializes [CS09, Properties 3.3 and 3.4] and the surrounding discussion to our setting:

PROPOSITION 3.1.10. *For all signatures Σ, Σ', multisets M, M', and multiset rewriting system $\mathcal{R}$, the sequent $\ulcorner\mathcal{R}\urcorner\,;\,\ulcorner M\urcorner \longrightarrow_\Sigma \exists\Sigma'.\ulcorner M'\urcorner$ is derivable in $\mathsf{LV}^{\mathsf{obs}}$ if and only if $\Sigma\,;\,M \longrightarrow^*_{\mathcal{R}} (\Sigma, \Sigma')\,;\,M'$.*

[1] In particular, we expect the constants to be disjoint from M and from those appearing in any other rule.

3.1.2. First-Order Multiset Rewriting with Persistence. Facts in multisets represent pieces of knowledge. In section 3.1.1, these facts were ephemeral: they could be consumed or destroyed by applying multiset rewrite rules. Often times, we would like some facts to be *persistent*, i.e., for some facts to be reusable and preserved by all rules. To this end, we partition facts as **persistent** (indicated by bold face, $\mathbf{p}$) and **ephemeral** (indicated by sans serif face, p). We then extend the multiset rewriting system of the previous section to support persistence. In doing so, we diverge slightly from Cervesato et al. [Cer+05] to allow for the set of persistent facts to grow across time.

Persistent facts are reusable, so we do not care about their multiplicities in multisets. For simplicity, we assume throughout that they form a set. We also separate persistent facts from ephemeral facts and write a generic multiset as $\mathbf{\Pi}, M$, where the set $\mathbf{\Pi}$ contains the persistent facts and the multiset M contains the ephemeral facts. A **multiset-in-context** is now a triple $\Sigma \,;\, \mathbf{\Pi}, M$. As before, all term-level symbols appearing in $\mathbf{\Pi}$ and M are contained in the signature Σ.

A **multiset rewrite rule** is now schematically represented by

$$r : \forall \vec{x}.\boldsymbol{\pi}(\vec{x}), F(\vec{x}) \to \exists \vec{n}.\boldsymbol{\pi}'(\vec{x}, \vec{n}), G(\vec{x}, \vec{n}),$$

where F and G are as before, and $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ are sets of persistent facts.

As before, multiset rewrite rules describe localized changes to multisets. Fix a multiset rewriting system $\mathcal{R}$. Given a rule $r : \forall \vec{x}.\boldsymbol{\pi}(\vec{x}), F(\vec{x}) \to \exists \vec{n}.\boldsymbol{\pi}'(\vec{x}, \vec{n}), G(\vec{x}, \vec{n})$ in $\mathcal{R}$ and some choice of constants $\vec{t}$ for $\vec{x}$, we say that the **instantiation** $r(\vec{t}) : \boldsymbol{\pi}(\vec{t}), F(\vec{t}) \to \exists \vec{n}.\boldsymbol{\pi}'(\vec{t}, \vec{n}), G(\vec{t}, \vec{n})$ is **applicable** to a multiset $\mathbf{\Pi}, M$ if there exists a multiset M' such that $M = F(\vec{t}), M'$ and if $\boldsymbol{\pi}(\vec{t}) \subseteq \mathbf{\Pi}$. The rule r is applicable to M if $r(\vec{t})$ is applicable to M for some $\vec{t}$. In these cases, the **result** of applying $r(\vec{t})$ to $\mathbf{\Pi}, M$ is the multiset $\left(\mathbf{\Pi} \cup \boldsymbol{\pi}'(\vec{t}, \vec{d})\right), G(\vec{t}, \vec{d}), M'$, where $\vec{d}$ is a choice of fresh constants. Again, we assume that the constants $\vec{d}$ do not appear in M or in $\mathcal{R}$. Because $\mathbf{\Pi}$ and $\boldsymbol{\pi}'(\vec{t}, \vec{d})$ were both assumed to be sets, the multiset $\mathbf{\Pi} \cup \boldsymbol{\pi}'(\vec{t}, \vec{d})$ in the result is again a set.

Multiset rewrite rules again equivalently describe localized changes to multisets-in-context. Consider a rule $r : \forall \vec{x}.\boldsymbol{\pi}(\vec{x}), F(\vec{x}) \to \exists \vec{n}.\boldsymbol{\pi}'(\vec{x}, \vec{n}), G(\vec{x}, \vec{n})$ in a multiset rewriting system $\mathcal{R}$. Given a signature Σ, an **instantiation** $r(\vec{t})$ of r is a rule of the form $r(\vec{t}) : \boldsymbol{\pi}(\vec{t}), F(\vec{t}) \to \exists \vec{n}.\boldsymbol{\pi}'(\vec{t}, \vec{n}), G(\vec{t}, \vec{n})$ for some $\vec{t}$ with $\Sigma \vdash \vec{t}$. This instantiation is **applicable** to a multiset-in-context $\Sigma \,;\, \mathbf{\Pi}, M$ if there exists a multiset M' such that $M = F(\vec{t}), M'$ and if $\boldsymbol{\pi}(\vec{t}) \subseteq \mathbf{\Pi}$. The **result** of applying the instantiation $\Sigma \,;\, \mathbf{\Pi}, M$ is the multiset-in-context $\Sigma, \vec{n} \,;\, \left(\mathbf{\Pi} \cup \boldsymbol{\pi}'(\vec{t})\right), G(\vec{t}), M'$, where we extend the signature Σ with the globally fresh symbols $\vec{n}$ (module α-renaming). We can represent this transition schematically as:

$$\begin{aligned}\Sigma \,;\, \boldsymbol{\pi}(\vec{t}), \mathbf{\Pi}', F(\vec{t}), M' \longrightarrow_{\mathcal{R}, (r:\forall \vec{x}.\boldsymbol{\pi}(\vec{x}), F(\vec{x}) \to \exists \vec{n}.\boldsymbol{\pi}'(\vec{x}, \vec{n}), G(\vec{x}, \vec{n}))} \\ \Sigma, \vec{n} \,;\, \left(\boldsymbol{\pi}(\vec{t}) \cup \mathbf{\Pi}' \cup \boldsymbol{\pi}'(\vec{t})\right), G(\vec{t}), M' \quad \text{if } \Sigma \vdash \vec{t},\end{aligned}$$

where $\mathcal{R}$ lists the other rules in the multiset rewriting system. The **active multiset** is $(\boldsymbol{\pi}, F)(\vec{t})$, while the **stationary multiset** is $\mathbf{\Pi}', M'$.

The definitions of trace and execution are as before.

Example 3.1.11. We use memoization to improve the time complexity of example 3.1.8, which computed the n-th Fibonacci number. Memoization uses a persistent fact $\mathbf{memo}(n, m)$ that means "m is the n-th Fibonacci number". Care is often needed when designing multiset rewriting systems to counter the effects of non-deterministic rule application. Indeed, in the case of memoization, the scheduler could chose to ignore available memoized values. To see how, consider a naïve implementation of memoization, where extend example 3.1.8 with the rule

$$f_{memo} : \forall l, n, m.\mathsf{fib}(n, l), \mathbf{memo}(n, m) \to \mathsf{val}(l, m).$$

The scheduler could apply the rule f to the multiset $\mathsf{fib}(\mathtt{s}(\mathtt{s}(\mathtt{z})), l), \mathbf{memo}(\mathtt{s}(\mathtt{s}(\mathtt{z})), 2)$ instead of f_{memo}, even though a memoized value is available.

We can force the scheduler's hand by disabling rules after one use. We do so by making them depend on an ephemeral fact that is never replaced. Concretely, we use an ephemeral fact $\mathsf{notyet}(n)$

that is consumed on the first invocation of $\mathsf{fib}(n, l)$:

$$f_0 : \forall l.\mathsf{fib}(\mathtt{z}, l), \mathsf{notyet}(\mathtt{z}) \to \mathbf{memo}(\mathtt{z}, \mathtt{s}(\mathtt{z})), \mathsf{val}(l, \mathtt{s}(\mathtt{z})) \tag{14}$$

$$f_1 : \forall l.\mathsf{fib}(\mathtt{s}(\mathtt{z}), l), \mathsf{notyet}(\mathtt{s}(\mathtt{z})) \to \mathbf{memo}(\mathtt{s}(\mathtt{z}), \mathtt{s}(\mathtt{z})), \mathsf{val}(l, \mathtt{s}(\mathtt{z})) \tag{15}$$

$$\begin{aligned} f : \forall l, n.\mathsf{fib}(\mathtt{s}(\mathtt{s}(n)), l), \mathsf{notyet}(\mathtt{s}(\mathtt{s}(n))) \to \\ \to \exists l', l''.\mathsf{fibcont}(\mathtt{s}(\mathtt{s}(n)), l, l', l''), \mathsf{fib}(\mathtt{s}(n), l'), \mathsf{fib}(n, l'') \end{aligned} \tag{16}$$

The rule f uses the modified continuation fact $\mathsf{fibcont}(k, l, l', l'')$. As with the continuation fact $\mathsf{cont}(l, l', l'')$ of example 3.1.8, it means that the values in locations l' and l'' should be added and stored in l. We also use it to mean that this sum should be memoized as the value of the k-th Fibonacci number. Because the fact $\mathsf{notyet}(n)$ gets consumed, subsequent attempts to compute $\mathsf{fib}(n, l')$ are forced to used the memoized value. Using the memoized value is captured by:

$$r : \forall l, n, m.\mathbf{memo}(n, m), \mathsf{fib}(n, l) \to \mathsf{val}(l, m) \tag{17}$$

We split the rule c of example 3.1.8 in two. The first rule c_f waits until values n and m are available in the locations l' and l''. It then causes them to be added and stored in location l. It also creates a continuation fact $\mathsf{addcont}(k, l)$. This fact is used by the rule c_a to memoize the value in l as the value of the k-th Fibonacci number:

$$\begin{aligned} c_f : \forall l, l', l'', k, m, n.\mathsf{fibcont}(k, l, l', l''), \mathsf{val}(l', n), \mathsf{val}(l'', m) \to \\ \to \mathsf{addcont}(k, l), \mathsf{add}(l, n + m) \end{aligned} \tag{18}$$

$$c_a : \forall l, k, m.\mathsf{addcont}(k, l), \mathsf{val}(l, m) \to \mathbf{memo}(k, m), \mathsf{val}(l, m) \tag{19}$$

To compute the n-th Fibonacci number, take an arbitrary execution from the multiset

$$\mathsf{notyet}(\mathtt{z}), \mathsf{notyet}(\mathtt{s}(\mathtt{z})), \ldots, \mathsf{notyet}(n), \mathsf{fib}(n, l).$$

We can show that this execution is finite. Its final multiset will contain a fact $\mathsf{val}(l, m)$, where m is the desired value. ◀

Sometimes matching substitutions can make a pair of rules indistinguishable:

Definition 3.1.12. Two rule instantiations $r_1(\theta_1)$ and $r_2(\theta_2)$ are **equivalent**, $r_1(\theta_1) \equiv r_2(\theta_2)$, if they are applicable to the same multisets, and if whenever they are applicable to some multiset M, then the result of applying either to M is the same (up to choice of fresh constants). Otherwise, they are **distinct**. ◀

We will use instantiation equivalence in section 3.3 to study the relationship between various forms of fairness and properties of fair traces. We can characterize it as follows:

Proposition 3.1.13. *Consider rules $r_i : \forall \vec{x}_i.\boldsymbol{\pi}_\mathbf{i}(\vec{x}_i), F_i(\vec{x}_i) \to \exists \vec{n}_i.\boldsymbol{\pi}'_\mathbf{i}(\vec{x}_i, \vec{n}_i), G_i(\vec{x}_i, \vec{n}_i)$ and matching substitutions θ_i for $i = 1, 2$. The instantiations $r_1(\theta_1)$ and $r_2(\theta_2)$ are equivalent if and only if*

(1) $\boldsymbol{\pi}_\mathbf{1}(\theta_1), F_1(\theta_1) = \boldsymbol{\pi}_\mathbf{2}(\theta_2), F_2(\theta_2)$;
(2) $\exists \vec{n}_1.G_1(\theta_1, \vec{n}_1) = \exists \vec{n}_2.G_2(\theta_2, \vec{n}_2)$ (up to renaming of bound variables); and
(3) $\exists \vec{n}_1.\boldsymbol{\pi}_\mathbf{1}(\theta_1) \cup \boldsymbol{\pi}'_\mathbf{1}(\theta_1, \vec{n}_1) = \exists \vec{n}_2.\boldsymbol{\pi}_\mathbf{2}(\theta_2) \cup \boldsymbol{\pi}'_\mathbf{2}(\theta_2, \vec{n}_2)$ (up to renaming of bound variables).

Proof. We start with sufficiency. Assume that $r_1(\theta_1)$ and $r_2(\theta_2)$ are equivalent. Then they are both applicable to the multiset $\boldsymbol{\pi}_\mathbf{1}(\theta_1), F_1(\theta_1)$, so we deduce $(\boldsymbol{\pi}_\mathbf{2}(\theta_2), F_2(\theta_2)) \subseteq (\boldsymbol{\pi}_\mathbf{1}(\theta_1), F_1(\theta_1))$. A symmetric observation gives the opposite inclusion, so we deduce that the multisets are equal. Next, the result of applying either to $M = \boldsymbol{\pi}_\mathbf{1}(\theta_1), F_1(\theta_1)$ is the same, so

$$\exists \vec{n}_1.\left(\boldsymbol{\pi}_\mathbf{1}(\theta_1) \cup \boldsymbol{\pi}'_\mathbf{1}(\theta_1, \vec{n}_1)\right), G_1(\theta_1, \vec{n}_1) = \exists \vec{n}_2.\left(\boldsymbol{\pi}_\mathbf{2}(\theta_2) \cup \boldsymbol{\pi}'_\mathbf{2}(\theta_2, \vec{n}_2)\right), G_2(\theta_2, \vec{n}_2)$$

up to renaming of bound variables. Recall that the collections of ephemeral and persistent facts are disjoint, so this implies the remaining two conditions.

Next, we show necessity. The first condition implies that $r_1(\theta_1)$ and $r_2(\theta_2)$ are applicable to the same multisets. The last two conditions imply that the result of applying either rule to a given multiset is the same. We deduce that the two rule instantiations are equivalent. □

Example 3.1.14. Consider the rules

$$r_1 : \forall x, y.A(x, y) \to \exists n.B(x, n),$$
$$r_2 : \forall x.A(x, x) \to \exists n.B(x, n)$$

and matching substitutions $\theta_1 = [a, a/x, y]$ and $\theta_2 = [a/x]$. Then $r_1(\theta_1) \equiv r_2(\theta_2)$:

$$r_1(\theta_1) : A(a, a) \to \exists n.B(a, n),$$
$$r_2(\theta_2) : A(a, a) \to \exists n.B(a, n).$$

Moreover, applying either $r_1(\theta_1)$ or $r_2(\theta_2)$ to the multiset $A(a, a), C(b, b)$ gives $B(a, a'), C(b, b)$ for some fresh constant a'. ◂

3.1.2.1. *Relation to Linear Logic.* We conjecturally relate first-order multiset rewriting with persistence to the sequent calculus $\mathsf{LV}^{\mathsf{obs}}_{\mathbf{1}\otimes\exists!}$ of Cervesato and Scedrov [CS09, § 2.6]. It is given by the rules of fig. 3.1, except that the rule (Obs) is replaced by the observation rule

$$\frac{}{\Gamma, \Gamma' \,;\, A \longrightarrow_{\Sigma,\Sigma'} \exists\Sigma'.(!\,\Gamma, \Delta)} \ (\textsc{Obs'})$$

where we write $!\,\Gamma$ for the linear context obtained by prefixing each fact in Γ with the bang operator !. Cut elimination does not hold in $\mathsf{LV}^{\mathsf{obs}}_{\mathbf{1}\otimes\exists!}$: the rule (Cut) is admissible, but (Cut!) is not [CS09, p. 1056].

We adapt the homomorphic mapping of section 3.1.1.1 to handle persistent facts. In positive positions, we prefix persistent facts with the bang operator; in negative positions, we translate them as though they were ephemeral. Where M ranges over arbitrary multisets potentially including both ephemeral and persistent facts:

$$\ulcorner\varnothing\urcorner = \mathbf{1} \quad \ulcorner M, \mathsf{s}\urcorner = \ulcorner M\urcorner \otimes s \quad \ulcorner M, \mathbf{p}\urcorner = \ulcorner M\urcorner \otimes p$$
$$\llcorner\varnothing\lrcorner = \mathbf{1} \quad \llcorner M, \mathsf{s}\lrcorner = \llcorner M\lrcorner \otimes s \quad \llcorner M, \mathbf{p}\lrcorner = \llcorner M\lrcorner \otimes !\,p$$
$$\ulcorner\forall\vec{x}.\boldsymbol{\pi}(\vec{x}), F(\vec{x}) \to \exists\vec{n}.\boldsymbol{\pi}'(\vec{x}, \vec{n}), G(\vec{x}, \vec{n})\urcorner = \forall\vec{x}.\ulcorner\boldsymbol{\pi}, F\urcorner \multimap \exists\vec{n}.\llcorner\boldsymbol{\pi}', G\lrcorner$$

We extend the homomorphism to multiset rewriting systems, where $\ulcorner\varnothing\urcorner = \cdot$ and $\ulcorner\mathcal{R}, r\urcorner = \ulcorner\mathcal{R}\urcorner, \ulcorner r\urcorner$.

The following conjecture generalizes proposition 3.1.10 to the setting with persistence. We will not use this conjecture, but state it for the sake of analogy with first-order multiset rewriting.

Conjecture 3.1.15. *For all signatures Σ, Σ', multisets $\mathbf{\Pi}, M$ and $\mathbf{\Pi}', M'$, and multiset rewriting system $\mathcal{R}$, the sequent $\ulcorner\mathcal{R}\urcorner \,;\, \ulcorner\mathbf{\Pi}, M\urcorner \longrightarrow_\Sigma \exists\Sigma'.\llcorner\mathbf{\Pi}, M'\lrcorner$ is derivable in $\mathsf{LV}^{\mathsf{obs}}_{\mathbf{1}\otimes\exists!}$ if and only if $\Sigma \,;\, \mathbf{\Pi}, M \longrightarrow^*_{\mathcal{R}} (\Sigma, \Sigma') \,;\, \mathbf{\Pi}', M'$.*

3.1.3. Semantic Irrelevance of Fresh Constants. The constants in fresh-constant substitutions are, by construction, not semantically meaningful. Indeed, they are arbitrarily chosen globally fresh constants.

In the absence of persistence, this semantic irrelevance is made precise by appealing to proposition 3.1.10. Indeed, consider some sequence of rewriting steps $\Sigma \,;\, M \longrightarrow^*_{\mathcal{R}} (\Sigma, \Sigma') \,;\, M'$. By proposition 3.1.10 and an induction on the rules, the fresh constants Σ' must have been obtained by extending the signature Σ using the rule ($\exists$L). In each case, the signature is extended using some symbol x that was previously a bound variable in the goal formula. In particular, the new symbol x could freely be α-varied prior to being used to extend the signature Σ. If conjecture 3.1.15 holds, then an analogous observation can be made for multiset rewriting with persistence.

As a result of these observations, we identify traces up to refreshing substitutions. A **refreshing substitution** for a trace $T = (M_0, (r_i; (\theta_i, \xi_i))_i)$ is a collection of fresh-constant substitutions $\eta = (\eta_i)_i$ such that $[\eta]T = (M_0, (r_i; (\theta_i, \eta_i))_i)$ is also a trace. Explicitly, we identify traces T and T' if there exists a refreshing substitution η such that $T' = [\eta]T$.

3.1.4. Parallel Rule Applications. We have so far only considered sequential rule application. However, we are interested in modelling parallel computation, and to do so, we would expect parallel (or simultaneous) rule application to be required. To this end, we briefly discuss parallel multiset rewriting systems. We show that we can emulate parallel rule application using sequential rule application and vice-versa. As a result, it will be sufficient to consider only sequential rule applications.

We define an operator $*$ on rules that combines them for parallel application. Given rules $r_i : \forall \vec{x}_i.\boldsymbol{\pi_i}(\vec{x}_i), F_i(\vec{x}_i) \to \exists \vec{n}_i.\boldsymbol{\pi'_i}(\vec{x}_i, \vec{n}_i), G_i(\vec{x}_i, \vec{n}_i)$ for $i = 1, 2$, let the rule $r_1 * r_2$ be given by:

$$r_1 * r_2 : \forall \vec{x}_1, \vec{x}_2.(\boldsymbol{\pi_1}(\vec{x}_1) \cup \boldsymbol{\pi_2}(\vec{x}_2)), F_1(\vec{x}_1), F_2(\vec{x}_2) \to \\ \to \exists \vec{n}_1, \vec{n}_2.(\boldsymbol{\pi'_1}(\vec{x}_1, \vec{n}_1) \cup \boldsymbol{\pi'_2}(\vec{x}_2, \vec{n}_2)), G_1(\vec{x}_1, \vec{n}_1), G_2(\vec{x}_2, \vec{n}_2).$$

The multiset rewrite rule $r_1 * r_2$ captures applying r_1 and r_2 in parallel. Intuitively, every application of this rule splits the ephemeral portion of a multiset in two, applies each of the rules separately, and then recombines the results at the end. It is clear that $*$ is an associative and commutative operator with identity $1_* : \varnothing \to \varnothing$.

Given a multiset rewriting system $\mathcal{R}$, let the *parallel multiset rewriting system* $\mathcal{R}^*$ be the multiset rewriting system given by:

$$\mathcal{R}^* = \{r_1 * \cdots * r_n \mid n \in \mathbb{N}, r_i \in \mathcal{R}\}.$$

The following proposition shows that parallel rewriting can be emulated by sequential rewriting. At a high level, its proof replaces each rule $r_1 * \cdots * r_n$ appearing in a trace $M \to^*_{\mathcal{R}^*} M'$ by the sequence of rules $r_1, \ldots, r_n$. We can make this replacement because, by definition of $*$, the rule $r_1 * \cdots * r_n$ describes making the localized changes described by each r_i to a disjoint portion (modulo persistence) of the multiset. Because each r_i rewrites a disjoint portion of the multiset, the rules $r_1, \ldots, r_n$ do not disable or otherwise interfere with each other, so applying each r_i sequentially gives the same result.

Proposition 3.1.16. *For all multiset rewriting systems $\mathcal{R}$ and multisets M and M', $M \to^*_{\mathcal{R}} M'$ if and only if $M \to^*_{\mathcal{R}^*} M'$.*

Proof. Sufficiency is clear: every trace over $\mathcal{R}$ is a trace over $\mathcal{R}^*$. Conversely, assume that $M \to^*_{\mathcal{R}^*} M'$. We proceed by induction on the number of steps taken. If no steps were taken, i.e., if $M \to^*_{\mathcal{R}^*} M'$ by reflexivity, then we are done. Now assume that $m + 1$ steps were taken, i.e., that $M \to_{\mathcal{R}^*} M'' \to^*_{\mathcal{R}^*} M'$ for some M''. Assume that the first step is given by the rule

$$r_1 * \cdots * r_k : \forall \vec{x}_1, \ldots, \vec{x}_k.\left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\vec{x}_i)\right), F_1(\vec{x}_1), \ldots, F_k(\vec{x}_k) \to \\ \to \exists \vec{n}_1, \ldots, \vec{n}_k.\left(\bigcup_{i=1}^{k} \boldsymbol{\pi}'_i(\vec{x}_i, \vec{n}_i)\right), G_1(\vec{x}_1, \vec{n}_1), \ldots, G_k(\vec{x}_k, \vec{n}_k)$$

for some rules $r_i \in \mathcal{R}$ and $k \geq 1$, with matching substitution θ and fresh-constant substitution ξ. Let θ_i and ξ_i be the obvious restrictions of θ and ξ to $\vec{x}_i$ and $\vec{n}_i$, respectively. By definition of rule application, it follows that

$$\begin{aligned} M &= \left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\theta)\right), \boldsymbol{\Pi}', F_1(\theta), \ldots, F_k(\theta), N \\ &= \left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\theta_i)\right), \boldsymbol{\Pi}', F_1(\theta_1), \ldots, F_k(\theta_k), N, \qquad (20) \\ M'' &= \left(\left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\theta)\right) \cup \boldsymbol{\Pi}' \cup \left(\bigcup_{i=1}^{k} \boldsymbol{\pi}'_i(\theta, \xi)\right)\right), G_1(\theta, \xi), \ldots, G_k(\theta, \xi), N \\ &= \left(\left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\theta_i)\right) \cup \boldsymbol{\Pi}' \cup \left(\bigcup_{i=1}^{k} \boldsymbol{\pi}'_i(\theta_i, \xi_i)\right)\right), G_1(\theta_1, \xi_1), \ldots, G_k(\theta_k, \xi_k), N. \qquad (21) \end{aligned}$$

We claim that, where $M_0 = M$, the following is a trace for some multisets $M_1, \ldots, M_{k+1}$, and that $M_{k+1} = M''$:

$$M_0 \xrightarrow{(r_1;(\theta_1,\xi_1))} M_1 \to \cdots \to M_k \xrightarrow{(r_k;(\theta_k,\xi_k))} M_{k+1} \tag{22}$$

This claim implies that $M \to_{\mathcal{R}}^* M''$. By the induction hypothesis, $M'' \to_{\mathcal{R}}^* M'$. Because $\to_{\mathcal{R}}^*$ is transitive, we can then conclude $M \to_{\mathcal{R}}^* M'$.

We show that (22) is a trace. To do so, we proceed by induction on j, $0 \le j \le k+1$, to show that

$$M_j = \left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\theta_i)\right) \cup \mathbf{\Pi}' \cup \left(\bigcup_{i=1}^{j} \boldsymbol{\pi}'_i(\theta_i, \xi_i)\right), \\ G_1(\theta_1, \xi_1), \ldots, G_j(\theta_j, \xi_j), F_{j+1}(\theta_{j+1}), \ldots, F_k(\theta_k), N.$$

It will then follow that r_j is applicable to M_j for all $0 \le j \le k$, i.e., that (22) is a trace. It will also be immediate by (21) that $M_{k+1} = M''$.

The case $k = 0$ is immediate by eq. (20). Assume the result for some j, then by definition of rule application, we have

$$M_{j+1} = \left(\bigcup_{i=1}^{k} \boldsymbol{\pi}_i(\theta_i)\right) \cup \mathbf{\Pi}' \cup \left(\bigcup_{i=1}^{j+1} \boldsymbol{\pi}'_i(\theta_i, \xi_i)\right), \\ G_1(\theta_1, \xi_1), \ldots, G_{j+1}(\theta_{j+1}, \xi_{j+1}), F_{j+2}(\theta_{j+2}), \ldots, F_k(\theta_k), N.$$

This is exactly what we wanted to show. We conclude the result. □

Parallel rule applications for multiset rewriting systems are discussed by Cervesato [Cer01, §§ 5.3–5.4]. Our approach is inspired by Cervesato's: we both decompose a multiset into disjoint pieces, apply rules in parallel, and recombining multisets. However, we diverge in the details from Cervesato's approach by defining a new MRS $\mathcal{R}^*$ that captures parallel execution, instead of using an inductively defined parallel rewriting judgment. Our approach also allows for persistent facts.

3.2. Three Varieties of Fairness

Intuitively, fairness properties provide progress guarantees for components in computational systems. Countless varieties of fairness were introduced in the 1980s, and they were classified according to various taxonomies by Francez [Fra86] and Kwiatkowska [Kwi89]. A common classification is along the axis of *strength*. **Weak fairness** ensures that components that are almost always able to make progress do make progress infinitely often. In contrast, **strong fairness** ensures that components that are able to make progress infinitely often do make progress infinitely often.

We introduce three varieties of fairness for multiset rewriting systems—*rule fairness*, *fact fairness*, and *instantiation fairness*—and we give a weak and a strong formulation for each. We motivate each variety of fairness by an example. In section 3.2.4, we show that these three varieties are independent.

3.2.1. Rule Fairness. We motivate rule fairness using an encoding of Petri nets as multiset rewriting systems. Petri nets [Pet80; Pet77] are structures used to model concurrency. A *Petri net* is given by two sets—a set P of *places* and a set T of *transitions*—and a pair of functions $I, O : T \to \wp(P)$ specifying the *inputs* and *outputs* of the transitions in T.

Informally, one executes a Petri net by placing tokens in places and observing how the tokens move through the net. An assignment $\mu : P \to \mathbb{N}$ of tokens to places is called a *marking*. A *marked Petri net* (P, T, I, O, μ) is conveniently depicted as a directed graph, where circles represent places, solid dots represent tokens, thick lines represent transitions, and arrows represent input/output. For example, fig. 3.2(a) depicts the marked Petri net given by $P = \{p_1, p_2, p_3\}$, $T = \{t_1\}$, $I(t_1) = \{p_1, p_2\}$, $O(t_1) = \{t_3\}$, and $\mu_0 = (p_1 \mapsto 2, p_2 \mapsto 1, p_3 \mapsto 0)$.

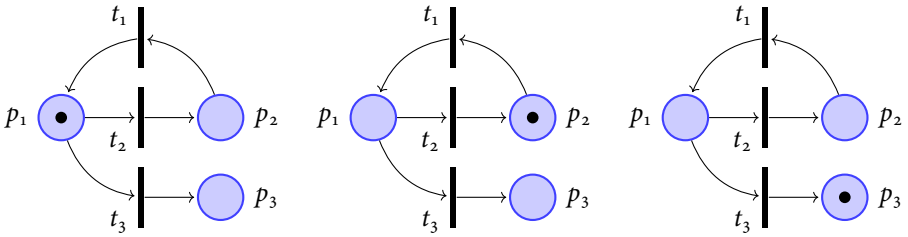
A transition is *enabled* if all of its inputs have at least one token. In this case, it *fires* by taking one token from each of its input places and adding one token to each of its output places; this changes the marking of the Petri net. A Petri net executes by repeatedly firing enabled transitions.

(A) Marking μ_0 (B) Marking μ_1

FIGURE 3.2. Two markings of the same Petri net

An *execution* is then a sequence $\mu_0, \mu_1, \ldots$ of markings, each obtained from its predecessor by firing an enabled transition. For example, in the Petri net of fig. 3.2(a), the transition t_1 is enabled, and the result of firing t_1 is the Petri net of fig. 3.2(b). When considering executions from a given marked Petri net, it is often more convenient to notate executions by their *firing sequence*, i.e., by the sequence of transitions that fired, than by the sequence of markings. The firing sequence for the execution μ_0, μ_1 is t_1.

Transitions fire non-deterministically. Consider, for example, the marked Petri net in fig. 3.3(a). It could fire t_2 first to obtain the marking μ_1 of fig. 3.3(b). From here, it can fire t_1 to return to the marking μ_0. Alternatively, it could have fired the transition t_3 to get the marking μ_2 of fig. 3.3(c). No transitions are enabled in this marking, so an execution ends as soon as it reaches this marking. It follows that all executions from μ_0 are given by the firing sequences[2] t_2, $(t_2t_1)^\infty$, or $(t_2t_1)^*t_3$.



(A) Marking μ_0 (B) Marking μ_1 (C) Marking μ_2

FIGURE 3.3. Markings reachable from fig. 3.3(a), illustrating non-deterministic firings.

In applications, it is often desirable to rule out so-called "unfair" executions. For example, we could deem the execution $(t_2t_1)^\omega$ to be unfair because, though the transition t_3 is enabled infinitely often, it never fires. To this end, we recall the definitions of weak and strong fairness for Petri nets [Leu+88]. We say that an execution $(\mu_i)_{i\in I}$ of a Petri net is

- *weakly fair* if it is finite, or if it is infinite and for all transitions $t \in T$, if t is enabled on all but finitely many markings μ_i, then there exist infinitely many $i \in I$ such that μ_{i+1} was obtained from μ_i by firing t;
- *strongly fair* if it is finite, or if it is infinite and for all transitions $t \in T$, if t is enabled on infinitely many markings μ_i, then there exist infinitely many $i \in I$ such that μ_{i+1} was obtained from μ_i by firing t.

Weak and strong fairness rule out different executions. For example, the firing sequence $(t_2t_1)^\omega$ describes a weakly fair execution from μ_0: the execution alternates between the markings μ_0 and μ_1, so no transitions are enabled on all but finitely many markings. The execution is not strongly fair: the transition t_3 is enabled infinitely often, but it never fires. The only strongly fair executions from μ_0 are given by the finite firing sequences t_2 and $(t_2t_1)^*t_3$.

[2]We adopt notation from ω-regular languages, where Σ^* and Σ^ω respectively denote finite and infinite words over the alphabet Σ, and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$.

We can encode a Petri net (P, T, I, O) as a multiset rewriting system.[3] Each transition $t \in T$ induces a rule

$$t : \mathsf{i}_1, \ldots, \mathsf{i}_m \to \mathsf{o}_1, \ldots, \mathsf{o}_n$$

where $I(t) = \{i_1, \ldots, i_m\}$ and $O(t) = \{o_1, \ldots, o_n\}$. A marking specifies a multiset containing $\mu(p)$ many facts p for each place $p \in P$. Firing a transition t corresponds to applying the rule t. Observe that a transition is enabled if and only if the rule is applicable.

Example 3.2.1. The Petri net of fig. 3.2 induces the single rule $t_1 : \mathsf{p}_1, \mathsf{p}_2 \to \mathsf{p}_3$. The marking μ_0 corresponds to the multiset $\mathsf{p}_1, \mathsf{p}_1, \mathsf{p}_2$. Firing the transition t_1, i.e., applying the rule t_1, results in the multiset $\mathsf{p}_1, \mathsf{p}_3$. ◀

Weak and strong fairness for Petri nets exactly correspond to the concepts of weak and strong rule-fairness for multiset rewriting systems. Consider an MRS $\mathcal{R}$. A trace $T = (M_0, (r_i; \delta_i)_{i \in I})$ is:

- **weakly rule-fair** if it is finite, or if it is infinite and for all rules $r \in \mathcal{R}$, if r is applicable to all but finitely many M_i, then there exist infinitely many $i \in I$ such that $r_i = r$;
- **strongly rule-fair** if it is finite, or if it is infinite and for all $r \in \mathcal{R}$, if r is applicable to infinitely many M_i, then there exist infinitely many $i \in I$ such that $r_i = r$.

3.2.2. Fact Fairness. Rule fairness alone is insufficient for an intuitively reasonable notion of fairness. We illustrate this fact by means of an MRS that grows and shrinks trees of finite height. Consider a collection of formulas $\mathsf{B}_n(a, s)$ capturing branches in a tree. Here, $n \in \mathbb{N}$ denotes some amount of "growth potential", a is the branch's ancestor, and s is a globally unique symbol identifying the branch. The root of a tree is given by a formula $\mathsf{B}_n(a, a)$. We can depict trees-as-multisets graphically, using dots to represent growth potential. For example, fig. 3.4 depicts multiset $\mathsf{B}_2(a, a), \mathsf{B}_1(a, b), \mathsf{B}_0(a, c), \mathsf{B}_3(c, d)$.

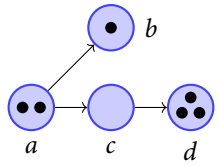


FIGURE 3.4. Visualization of the multiset $\mathsf{B}_2(a, a), \mathsf{B}_1(a, b), \mathsf{B}_0(a, c), \mathsf{B}_3(c, d)$ as a tree

A branch can sprout a new branch if it has positive growth potential. Branching is given by a family of rules, where we have a "branching rule" $b_{j,k}$ for all $n > 0$ and $j, k \geq 0$ such that $j + k = n$:

$$b_{j,k} : \forall x y.\mathsf{B}_n(x, y) \to \exists z.\mathsf{B}_j(x, y), \mathsf{B}_k(y, z).$$

It takes a branch y with potential n, and creates a new descendent z with potential k, leaving y with potential j.

Consider the following execution starting from a root a_0 with two units of growth potential:

$$\begin{aligned}
&\mathsf{B}_2(a_0, a_0) \\
&\xrightarrow{b_{1,1}} \mathsf{B}_1(a_0, a_0), \mathsf{B}_1(a_0, a_1) \\
&\xrightarrow{b_{0,1}} \mathsf{B}_1(a_0, a_0), \mathsf{B}_0(a_0, a_1), \mathsf{B}_1(a_1, a_2) \\
&\quad\vdots \\
&\xrightarrow{b_{0,1}} \mathsf{B}_1(a_0, a_0), \mathsf{B}_0(a_0, a_1), \mathsf{B}_0(a_1, a_2), \ldots, \mathsf{B}_1(a_n, a_{n+1}) \\
&\quad\vdots
\end{aligned} \tag{23}$$

[3] This encoding is very similar to the one used to encode Petri nets in Concurrent LF [Cer+03, § 5.3.1].

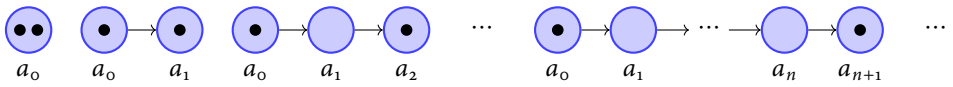


FIGURE 3.5. Graphical depiction of the multisets in execution (23)

It is graphically depicted by the sequence of trees in fig. 3.5. This execution grows the tree forever by applying $b_{0,1}$ to the right-most branch in the tree. It is both weakly and strongly rule-fair. Indeed, the only rule that is applicable infinitely often is $b_{0,1}$, and it is applied infinitely often. However, the execution could be deemed "unfair" to the branch a_0. Indeed, though a_0 has a unit of potential left, it never gets to use it to grow a second branch. This motivates the notion of fact fairness, which ensures that facts that could be used to take a step are not ignored.

Consider an MRS $\mathcal{R}$. We say that a fact $J \in M$ is **enabled** in M if there exists an instantiation $r(\theta) : F(\theta) \to \exists \vec{n}.G(\theta, \vec{n})$ of a rule $r \in \mathcal{R}$ such that $J \in F(\theta)$. A trace $T = (M_0, (r_i; \delta_i)_{i \in I})$ is:

- **weakly fact-fair** if it is finite, or if it is infinite and for all facts $J \in \mathsf{supp}(T)$, if J is enabled in all but finitely many M_i, then there exist infinitely many $i \in I$ such that J is in the active multiset of $r_i(\theta_i)$;
- **strongly fact-fair** if it is finite, or if it is infinite and for all facts $J \in \mathsf{supp}(T)$, if J is enabled in infinitely many M_i, then there exist infinitely many $i \in I$ such that J is in the active multiset of $r_i(\theta_i)$.

3.2.3. Instantiation Fairness. To illustrate a final variety of fairness, we suspend disbelief and assume that we can water individual branches. Watering a branch gives it a unit of potential:

$$w_n : \forall xy.B_n(x, y) \to B_{n+1}(x, y).$$

Intuitively, fairness should ensure that no branch in the tree is left unwatered forever. Rule-fairness and fact-fairness are insufficient to ensure this in general. To illustrate this, we make the simplifying assumption that we only have the rule w_0 plus a family of rules that redistribute potential:

$$r_{m,n} : \forall xyz.B_{m+1}(x, y), B_n(y, z) \to B_m(x, y), B_{n+1}(y, z).$$

Consider the execution starting from the multiset $B_0(a, a), B_0(a, b), B_0(b, c)$ given by the sequence of rule instantiations

$$w_0(a), r_{0,0}(a, a, b), r_{0,0}(a, b, c), \quad w_0(a), r_{0,0}(a, a, b), r_{0,1}(a, b, c), \quad \ldots, w_0(a), r_{0,0}(a, a, b), r_{0,n}(a, b, c), \ldots .$$

After the $3n$-th rule, the multiset is $B_0(a, a), B_0(a, b), B_n(b, c)$. This execution is strongly rule-fair: the only rules applicable infinitely often are w_0 and $r_{0,0}$, and they are both applied infinitely often. It is strongly fact-fair: the only facts enabled infinitely often are $B_0(a, a)$, $B_1(a, a)$, $B_0(a, b)$, and $B_1(a, b)$. Each of these appears in the active multisets of infinitely many rule applications. However, the instantiation is still intuitively unfair because though the branch b could be watered infinitely often, it never gets watered. Explicitly, the rule instantiation $w_0(a, b)$ is applicable infinitely often, but it is never applied. To address this, we introduce *instantiation fairness*.

Recall the definition of equivalent instantiations $r_1(\theta_1) \equiv r_2(\theta_2)$ from definition 3.1.12. Consider an MRS $\mathcal{R}$. A trace $T = (M_0, (r_i; \delta_i)_{i \in I})$ is:

- **weakly instantiation-fair** if it is finite, or if it is infinite and for all rules $r \in \mathcal{R}$ and θ, if $r(\theta)$ is applicable to all but finitely many M_i, then there exist infinitely many $i \in I$ such that $r_i(\theta_i) \equiv r(\theta)$;
- **strongly instantiation-fair** if it is finite, or if it is infinite and for all $r \in \mathcal{R}$ and θ, if $r(\theta)$ is applicable to infinitely many M_i, then there exist infinitely many $i \in I$ such that $r_i(\theta) = r(\theta)$.

3.2.4. Comparing Varieties of Fairness. We have a proliferation of varieties of fairness, and it raises the question: are they all useful and independent from each other? The first question is normative and implicitly presupposes an answer to the question: *useful to what end*? We motivated each kind of fairness by an application, but it is up to the practitioner to decide whether or not a particular kind of fairness is useful or desirable in a particular setting. We can, however, answer the second question. We show that rule, fact, and instantiation fairness are independent notions, no two of which imply the other. To do so, we construct traces that satisfy two of the forms of fairness, but not the third.

Consider the MRS given by the following rules:

$$a : \forall x.A(x) \to A(x)$$
$$b : \forall x.A(x), B \to A(x), B.$$

Consider the multiset $M_0 = A(c), A(d), B$. The trace given by alternating applications of $a(c)$ and $b(d)$ is strongly fact-fair and strongly-rule fair, but it is not weakly or strongly instantiation fair. Indeed, though the instantiation $b(c)$ is always applicable, it never gets applied.

Now consider the MRS given by the following rule:

$$r : \forall xy.A(x), B(y) \to \exists z.A(x), B(z).$$

Consider the multiset $M_0 = A(a), A(a'), B(b_0)$. Assume that we generate the fresh constants $b_1, b_2, \ldots$. Then the trace given by $r(a, b_0), r(a, b_1), r(a, b_2), \ldots$ is strongly instantiation-fair and strongly rule-fair, but it is not weakly or strongly fact-fair. Indeed, the only rule r is applied infinitely often, so the trace is strongly rule-fair, and all instantiations of r are globally unique, so the trace is instantiation-fair. It is not fact fair because, though the fact $A(a')$ is always enabled (it is in the active multiset of $r(a', b_n)$ for each n), it is never in the active multiset of a rule in the trace.

Finally, consider the MRS given by the following rules:

$$a : \forall x.A(x) \to \exists y.A(y)$$
$$b : \forall x.B(x) \to \exists y.B(y)$$
$$i : \forall x, y.A(x), B(y) \to A(x), B(y).$$

Consider the multiset $A(a_0), B(b_0)$. Consider the trace given by alternating applications of a and b:

$$A(a_0), B(b_0) \to A(a_1)B(b_0) \to A(a_1), B(b_1) \to A(a_2), B(b_1) \to A(a_2), B(b_2) \to \cdots$$

It is strongly fact-fair: each fact appears at most twice in the trace. It is strongly instantiation-fair: each rule instantiation is applicable at most twice in the trace. However, it is not weakly or strongly rule-fair: the rule i never gets applied.

3.2.5. Weak, Strong, and Über Fairness. We say that a trace is **weakly fair** if it is weakly rule-, fact-, and instantiation-fair, and **strongly fair** if it is strongly rule-, fact-, and instantiation-fair. Surprisingly, a much stronger notion of fairness arises naturally in applications of multiset rewriting systems. We say that a trace $(M_0, (r_i; \delta_i)_{i \in I})$ is **über fair** if it is finite, or if for all $i \in I$, $r \in \mathcal{R}$, and θ, whenever $r(\theta)$ is applicable to M_i, there exists a $j > i$ such that $r_j(\theta_j) \equiv r(\theta)$. Given an über fair trace T, we write $v_T(i, r, \theta)$ for the least such j if it exists. Every über fair trace is also strongly fair, and every strongly fair is also weakly fair.

3.3. Properties of Fair Traces

We study sufficient conditions for multiset rewriting systems to have fair traces. One of these, "interference-freedom", implies that all fair traces are permutations of each other. We also study the effects of permuting steps in traces.

The fair tail property is immediate from the definitions of fairness:

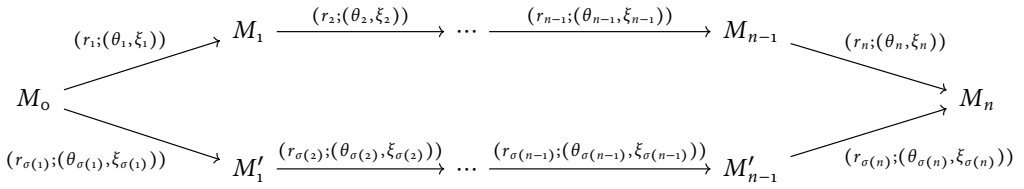
Proposition 3.3.1 (Fair Tail Property). *Let "fair" range over the nine notions of fairness considered in section 3.2.5. If $(M_0, (t_i; \delta_i)_{i \in I})$ is fair, then so is $(M_n, (t_i; \delta_i)_{n < i, i \in I})$ for all $n \in I$.*

Most notions of fairness are closed under concatenation with finite prefixes:

Proposition 3.3.2 (Fair Concatenation Property). *Let "fair" range over weak and strong rule, fact, and instantiation fairness. If $M_0 \to^* M_n$, and T is a fair trace from M_n, then $M_0 \to^* M_n$ followed by T is a fair trace from M_0. If T is finite, then the result also holds for über fairness.*

Interference-freedom roughly means that at any given point, the order in which we apply applicable rules does not matter. Write S_I for the group of bijections on I; its elements are called *permutations*. A permutation $\sigma \in S_I$ acts on a trace $T = (M_0, (t_i; \delta_i)_{i\in I})$ to produce a sequence $\sigma \cdot T = (M_0, (t_{\sigma(i)}; \delta_{\sigma(i)})_{i\in I})$. This sequence $\sigma \cdot T$ is a **permutation of** T whenever it is also a trace. We adopt group-theoretic notation for cyclic permutations and write $(x, \sigma(x), \sigma(\sigma(x)), \dots)$ for a cyclic permutation $\sigma : I \to I$; implicit is that all elements not in the orbit of x are fixed by σ. For example, let T be given by the sequence of rule instances t_1, t_2, t_3, t_4, t_5. Then $(4, 3, 2) \cdot T$ is the sequence t_1, t_4, t_2, t_3, t_5. Cyclic permutations of length two are called *transpositions*.

Consider an MRS $\mathcal{R}$ and let $r_1(\theta_1), \dots, r_n(\theta_n)$ enumerate all distinct instantiations of rules in $\mathcal{R}$ applicable to M_0. We say that $\mathcal{R}$ **commutes** on M_0 or is **interference-free** on M_0 if for all corresponding pairwise-disjoint fresh-constant substitutions ξ_i, the following diagram commutes for all permutations $\sigma \in S_{\mathbf{n}}$, and both paths around it are traces:



We note that interference-freedom is only defined if the enumeration of distinct applicable instantiations is finite. The following proposition is an immediate consequence of the definition of commuting rules:

Proposition 3.3.3. *Let $\mathcal{R}$ commute on M_0, and let $r_i(\theta_i)$ with $1 \le i \le n$ be the distinct instantiations applicable on M_0. If $M_0 \xrightarrow{(r_1;(\theta_1,\xi_1))} M_1$, then $r_2(\theta_2), \dots, r_n(\theta_n)$ are applicable to and commute on M_1.*

Interference-freedom implies the existence of über fair executions. To prove this, we construct a scheduler that enqueues all applicable rules and applies them one by one. Interference freedom ensures that a rule is still applicable once it reaches the front of the queue.

Given an MRS $\mathcal{R}$ and a property P of multisets, we say P holds **from** M_0 if for all traces $(M_0, (r_i; \delta_i)_{i\in I})$, P holds for M_0 and for M_i for all $i \in I$.

Proposition 3.3.4 (Fair Scheduler). *If $\mathcal{R}$ is interference-free from M_0, then there exists an über fair execution from M_0.*

Proof. We define an über fair execution $(r_i; (\theta_i, \xi_i)_i)$ from M_0 by induction on n.

Let $Q_0 = r_{01}(\theta_{01}), \dots, r_{0n}(\theta_{0n})$ be an enumeration of the distinct instantiations of rules from $\mathcal{R}$ applicable to M_0. The enumeration is finite because $\mathcal{R}$ commutes on M_0. We maintain for all n the following invariants:

- all distinct rules instantiations applicable to M_n appear in Q_n;
- all rules in Q_n are applicable to M_n and are distinct instantiations of rules in $\mathcal{R}$.

Let Q_n be given; we construct r_{n+1}, θ_{n+1}, ξ_{n+1}, M_{n+1} and Q_{n+1}. If Q_n is empty, then no rules are applicable to M_n and $(M_0, (r_i; (\theta_i, \xi_i)_i))$ is maximal. Otherwise, let $r_{n+1}(\theta_{n+1})$ be first element in Q_n so that $Q_n = r_{n+1}(\theta_{n+1}), Q'_n$ for some Q'_n. Pick a suitably fresh ξ_{n+1}. Take

$$M_n \xrightarrow{(r_{n+1};(\theta_{n+1},\xi_{n+1}))} M_{n+1}.$$

By assumption, all rules in Q_n are applicable to M_n, and these are all instantiations of rules in $\mathcal{R}$. By interference-freedom, $\mathcal{R}$ commutes on M_n, so all rules in Q'_n are applicable to M_{n+1}

by proposition 3.3.3. Let $Q_{n+1} = Q'_n, N_{n+1}$, where N_{n+1} is an enumeration of the distinct rule instantiations applicable to M_{n+1} not already in Q'_n. It is finite because $\mathcal{R}$ commutes on M_{n+1}. The invariants hold by construction.

The resulting trace is clearly über fair: for all j, if r is applicable to M_j, then it appears at some finite depth d in Q_{j+1} and will appear in the trace after d steps. □

Though interference-freedom simplifies fair scheduling, it is primarily of interest for reasoning about executions. For example, it is useful for showing confluence properties. It also lets us safely permute certain steps in a trace without affecting observations for session-typed processes (see chapter 6). This can simplify process equivalence proofs, because it lets us assume that related steps in an execution happen one after another.

Interference-freedom is a strong property, but it holds for many multiset rewriting systems of interest. This is because many systems can be captured using rules whose active multisets do not intersect, and rules with disjoint active multisets commute. In fact, even if their active multisets intersect, rules do not disable each other so long as they preserve these intersections. Because persistent facts are always preserved, it is sufficient to consider only ephemeral facts.

To make this intuition explicit, consider multisets $M_i \subseteq M$ for $1 \le i \le n$. Their **overlap** in M is $\Omega_M(M_1, \ldots, M_n) = M_1, \ldots, M_n \setminus M$. Consider an MRS $\mathcal{R}$ and let $r_i(\theta_i) : \boldsymbol{\pi}_{\mathbf{i}}(\theta_i), F_i(\theta_i) \to \exists \vec{n}_i.\boldsymbol{\pi}'_{\mathbf{i}}(\theta_i), G_i(\theta_i, \vec{n}_i)$, $1 \le i \le n$, enumerate all distinct instantiations of rules in $\mathcal{R}$ applicable to M. We say that $\mathcal{R}$ is **non-overlapping** on M if for all $1 \le i \le n$ and fresh-constant substitutions ξ_i, $F_i(\theta_i) \cap \Omega_M(F_1(\theta_1), \ldots, F_n(\theta_n)) \subseteq G_i(\theta_i, \xi_i)$, i.e., if each rule instantiation preserves its portion of the overlap.

Example 3.3.5. The overlap of A, B and B, C in A, B, C is $\Omega_{A,B,C}((A, B), (B, C)) = B$. The overlap of A, B and B, C in A, B, B, C is the empty multiset $\Omega_{A,B,B,C}((A, B), (B, C)) = \varnothing$. The overlap of A, B and B, C, and C, A in A, B, C is $\Omega_{A,B,C}((A, B), (B, C), (C, A)) = A, B, C$. ◂

Example 3.3.6. The MRS given by example 3.1.6 is non-overlapping from any multiset of the form Q, E where Q is a queue rooted at q, and E contains at most one fact of the form $\mathsf{enq}(q, v)$. ◂

Proposition 3.3.7 characterizes the application of non-overlapping rules, while proposition 3.3.9 characterizes the relationship between commuting and non-overlapping rules. Because persistent facts pose no difficulty (multiset union is a commutative operation), we elide them from these results for clarity.

Proposition 3.3.7. *Let $\mathcal{R}$ be non-overlapping on M_0 and let $r_i(\theta_i) : F_i(\theta_i) \to \exists \vec{n}_i.G(\theta_i, \vec{n}_i)$ with $1 \le i \le n$ be the distinct instantiations applicable to M_0. If $M_0 \xrightarrow{(r_1;(\theta_1,\xi_1))} M_1$ and $r_1, \ldots, r_n$ are non-overlapping on M_0, then $r_2(\theta_2), \ldots, r_n(\theta_n)$ are applicable to and non-overlapping on M_1.*

In particular, abbreviate $F_i(\theta_i)$ and $G_i(\theta_i, \xi_i)$ by F_i and G_i, respectively, and let O be the overlap $O = \Omega_{M_0}(F_1, \ldots, F_n) \cap F_1$. There exist F'_1 and G'_1 be such that $F_1 = O, F'_1$ and $G_1 = O, G'_1$, and there exists an M such that $M_0 = O, F'_1, M$ and $M_1 = O, G'_1, M$. The instantiations $r_2(\theta_2), \ldots, r_n(\theta_n)$ are all applicable to $O, M \subseteq M_1$.

Proof. Where $M = (S, m)$ is a multiset and $s \in S$, we abuse notation and write $M(s)$ for $m(s)$.

Let $O = \Omega_{M_0}(F_1, \ldots, F_n) \cap F_1$. By assumption, $O \subseteq G_1$, so $F_1 = O, F'_1$ and $G_1 = O, G'_1$ for some F'_1 and G'_1. It follows that $M_0 = O, F'_1, M$ and $M_1 = O, G'_1, M$ for some M. We show that $r_2(\theta_2), \ldots, r_n(\theta_n)$ are all applicable to O, M. Without loss of generality, we show that $r_2(\theta_2)$ is applicable to O, M. This requires that we show that $F_2 \subseteq O, M$, i.e., that $F_2(s) \le O(s) + M(s)$ for

all s. We compute:

$$(\Omega_{M_0}(F_1,\ldots,F_n))(s) = \max\left(0, \left(\sum_{i=1}^{n} F_i(s)\right) - M_0(s)\right)$$
$$= \max\left(0, \left(\sum_{i=2}^{n} F_i(s)\right) - M(s)\right), \tag{24}$$
$$O(s) = \min\left(F_1(s), \max\left(0, \left(\sum_{i=2}^{n} F_i(s)\right) - M(s)\right)\right). \tag{25}$$

Because $r_2(\theta_2)$ is applicable to M_0, we have $F_2 \subseteq M_0$, i.e., $F_2(s) \leq M_0(s) = O(s) + F_1'(s) + M(s)$ for all s. If $F_1'(s) = 0$, then we are done. Assume now that $F_1'(s) > 0$. From this it follows that $F_1(s) = F_1'(s) + O(s) > O(s)$. We consider three cases for the value of $O(s)$, based on the three possibilities in eq. (25):

Case $O(s) = F_1(s)$**:** Impossible, for it contradicts $F_1(s) > O(s)$.

Case $O(s) = 0$**:** Then $F_1(s) = 0$ or $\max(0, (\sum_{i=2}^{n} F_i(s)) - M(s)) = 0$. The case $F_1(s) = 0$ is impossible because $F_1(s) > O(s) = 0$. So $0 \geq (\sum_{i=2}^{n} F_i(s)) - M(s)$, so $M(s) \geq \sum_{i=2}^{n} F_i(s)$. Because the F_i are all non-negative, $M(s) \geq F_2(s)$, so we are done.

Case $O(s) = (\sum_{i=2}^{n} F_i(s)) - M(s)$**:** Then $O(s) + M(s) = \sum_{i=2}^{n} F_i(s) \geq F_2(s)$, so we are done.

We conclude that $r_2(\theta_2)$ is applicable to M_1.

Next, we show that $r_2(\theta_2), \ldots, r_n(\theta_n)$ is non-overlapping on M_1. In particular, we must show that for all i, $\Omega_{M_1}(F_2,\ldots,F_n) \cap F_i \subseteq G_i$. By assumption,

$$\Omega_{M_0}(F_1,\ldots,F_n) \cap F_i \subseteq G_i$$

for all i, so it is sufficient to show that

$$\Omega_{M_1}(F_2,\ldots,F_n) \subseteq \Omega_{M_0}(F_1,\ldots,F_n).$$

To do so, we show that for all s,

$$(\Omega_{M_1}(F_2,\ldots,F_n))(s) \leq (\Omega_{M_0}(F_1,\ldots,F_n))(s).$$

Let $M_1 = O, G_1', M$ as before. We compute:

$$(\Omega_{M_1}(F_2,\ldots,F_n))(s) = \max\left(0, \left(\sum_{i=2}^{n} F_i(s)\right) - M_1(s)\right)$$
$$= \max\left(0, \left(\sum_{i=2}^{n} F_i(s)\right) - O(s) - G_1'(s) - M(s)\right).$$

Because all values involved are non-negative,

$$\left(\sum_{i=2}^{n} F_i(s)\right) - O(s) - G_1'(s) - M(s) \leq \left(\sum_{i=2}^{n} F_i(s)\right) - M(s).$$

Recall that max is monotone and recall eq. (24). It follows that

$$(\Omega_{M_1}(F_2,\ldots,F_n))(s) \leq (\Omega_{M_0}(F_1,\ldots,F_n))(s)$$

as desired. We conclude that $r_2(\theta_2), \ldots, r_n(\theta_n)$ are non-overlapping on M_1. □

Lemma 3.3.8. *Consider distinct instantiations $r_i(\theta_i) : F_i(\theta_i) \to \exists \vec{n}_i.G_i(\theta_i, \vec{n}_i)$ that are applicable to M_0 for $i = 1, 2$. If they are non-overlapping on M_0, then they commute on M_0.*

Proof. We must show that for all disjoint fresh-constant substitutions ξ_1 and ξ_2, the following diagram commutes and both paths around it are traces:

$$\begin{array}{ccc} M_0 & \xrightarrow{(r_1;(\theta_1,\xi_1))} & M_1 \\ {\scriptstyle (r_2;(\theta_2,\xi_2))}\downarrow & & \downarrow{\scriptstyle (r_2;(\theta_2,\xi_2))} \\ M_1' & \xrightarrow{(r_1;(\theta_1,\xi_1))} & M_2. \end{array}$$

Fix some such substitutions and abbreviate $F_i(\theta_i)$ and $G_i(\theta_i, \xi_i)$ by F_i and G_i, respectively. Both paths around the square are traces by proposition 3.3.7; we show that it commutes.

Let $O_i' = F_i \cap \Omega_{M_0}(F_1, F_2)$, $O_{12} = O_1' \cap O_2'$, and $O_i = O_i' \setminus O_{12}$. By assumption, $O_i' \subseteq G_i$, so for some F_i' and G_i' we have $F_i(\theta_i) = O_i', F_i' = O_{12}, O_i', F_i'$ and $G_i = O_i', G_i' = O_{12}, O_i', G_i'$. Because $F_i \subseteq M_0$ for $i = 1, 2$, it follows that $M_0 = O_{12}, O_1, O_2, F_1', F_2', M$. Then the two paths are

$$M_0 \xrightarrow{(r_1;(\theta_1,\xi_1))} O_{12}, O_1, O_2, G_1', F_2', M \xrightarrow{(r_2;(\theta_2,\xi_2))} O_{12}, O_1, O_2, G_1', G_2', M$$
$$M_0 \xrightarrow{(r_2;(\theta_2,\xi_2))} O_{12}, O_1, O_2, F_1', G_2', M \xrightarrow{(r_1;(\theta_1,\xi_1))} O_{12}, O_1, O_2, G_1', G_2', M.$$

We conclude that the diagram commutes. □

In [Kav20a, Proposition 5], we claimed that the converse of proposition 3.3.9 was false. The counter-example used does not support this claim.

Proposition 3.3.9. *An MRS commutes on M_0 if it is non-overlapping on M_0.*

Proof. Assume that the rules are non-overlapping on M_0. Let $r_i(\theta_i) : F_i(\theta_i) \to \exists \vec{n}_i.G(\theta_i, \vec{n}_i)$ enumerate the distinct instantiations that are applicable to M_0 with $1 \le i \le n$. Consider pairwise-disjoint fresh-constant substitutions ξ_i for $1 \le i \le n$. We must show that $(M_0, (r_i; (\theta_i, \xi_i))_{1 \le i \le n})$ is a trace and that permuting its steps does not change the last multiset.

We proceed by induction on n to show that it is a trace. Informally, proposition 3.3.7 will ensure that no matter which instance we apply to get to the next multiset, the remaining instances will be applicable to and non-overlapping on that multiset. The result is immediate when $n = 0$. Assume that the result holds for some k, and assume that $n = k + 1$. By the induction hypothesis, the following sequence of k steps is a trace:

$$M_0 \xrightarrow{(r_1;(\theta_1,\xi_1))} M_1 \xrightarrow{(r_2;(\theta_2,\xi_2))} \cdots \xrightarrow{(r_{k-1};(\theta_{k-1},\xi_{k-1}))} M_{k-1} \xrightarrow{(r_k;(\theta_k,\xi_k))} M_k.$$

We proceed by induction on $0 \le j \le k$ to show that $r_{j+1}(\theta_{j+1}), \ldots, r_{k+1}(\theta_{k+1})$ are all applicable and non-overlapping on M_j.

Case $j = 0$**:** Immediate from the assumption that the rules are non-overlapping on M_0.

Case $j = j' + 1$ with $j' < k$**:** The instances $r_j(\theta_j), \ldots, r_{k+1}(\theta_{k+1})$ are all applicable to M_j by the induction hypothesis on j'. By proposition 3.3.7, the instances $r_{j+1}(\theta_{j+1}), \ldots, r_{k+1}(\theta_{k+1})$ are all applicable and non-overlapping on M_{j+1}.

This completes the nested induction on j, and we conclude that $r_{k+1}(\theta_{k+1})$ is applicable to M_k. Pairwise-disjointness of the ξ_i guarantees that the resulting sequence of $k + 1$ steps is a trace. This completes the induction on n.

Next, we observe that any permutation of the above trace is a trace. Indeed, the numbering of the $r_i(\theta_i) : F_i(\theta_i) \to \exists \vec{n}_i.G(\theta_i, \vec{n}_i)$ was arbitrary, and the proof that $(M_0, (r_i; (\theta_i, \xi_i))_{1 \le i \le n})$ was a trace did not depend in any way on the numbering of the rules or on the individual rules themselves.

Finally, we show that all permutations of the trace have the same final multiset. Concretely, fix some σ and let $(N_0, (r_{\sigma(i)}; (\theta_{\sigma(i)}, \xi_{\sigma(i)}))_{i \in \mathbf{n}}) = \sigma \cdot (M_0, (r_i; (\theta_i, \xi_i))_{i \in \mathbf{n}})$. We show that $M_n = N_n$. The permutation σ factors as a product of transpositions of adjacent steps by the proof of [Hun74, Corollary I.6.5]. By the previous paragraph, each of these transpositions preserves the property of being a trace. By lemma 3.3.8, each transposition preserves the multisets at its endpoints. In particular, each transposition preserves M_n. An induction on the number of transpositions in the factorization of σ gives that $M_n = N_n$. □

Weak, strong, and über fairness coincide in the presence of interference-freedom:

Proposition 3.3.10. *If $\mathcal{R}$ is interference-free from M_0 and T is a trace from M_0, then the following are equivalent: T is weakly instantiation-fair, T is strongly instantiation-fair, T is über fair.*

Proof. It is clear that über fairness implies both forms of instantiation-fairness, and that strong instantiation-fairness implies weak instantiation-fairness. It is sufficient to show that if T is weakly instantiation-fair, then it is über fair.

Assume that T is weakly instantiation-fair. If T is finite, then we are done, so assume that $T = (M_0; (r_i, \delta_i)_i)$ is infinite. Consider some arbitrary M_i, and assume that $r(\theta)$ is applicable to M_i. We must show that there exists some $k \geq 1$ such that $r_{i+k}(\theta_{i+k}) \equiv r(\theta)$. By induction on $k \geq 1$ with proposition 3.3.7, we know that if $r_{i+k}(\theta_{i+k}) \not\equiv r(\theta)$, then $r(\theta)$ is applicable to M_{i+k+1}. This implies that $r_{i+k}(\theta_{i+k}) \equiv r(\theta)$ for some $k \geq 1$ or that $r(\theta)$ is applicable to all but finitely many M_j (i.e., at most the first i multisets). In the first case, we are done. In the latter case, we know that $r(\theta)$ is applied infinitely often by weak instantiation-fairness. We conclude that T is über fair. □

Corollary 3.3.11 (Fair Concatenation for Über Fairness). *If $\mathcal{R}$ is interference-free from M_0, $M_0 \to^* M_n$, and T is an über fair trace from M_n, then $M_0 \to^* M_n$ followed by T is an über fair trace from M_0.*

Proof. Immediate by propositions 3.3.2 and 3.3.10. □

In light of proposition 3.3.10, we hereinafter use the words "fair trace" to equivalently mean "weakly fair trace", "strongly fair trace", or "über fair trace" when assuming interference-freedom.

Assumption 3.3.12. For the remainder of this section, we assume that if $(M_0, (r_i; \delta_i)_i)$ is a fair trace, then its MRS is interference-free from M_0.

In the remainder of this section, we study the effects of permutations on fair traces. We first show that fairness is invariant under permutation. Then, we show that all fair executions are permutations of each other.

Interference-freedom implies the ability to safely permute finitely many steps that do not depend on each other. However, it is not obvious that finite permutations, let alone infinite permutations, preserve fairness. We begin by showing that finite permutations preserve fairness. Our proof relies on the fact that finite permutations factor as products of cycles, which themselves factor as products of transpositions. We begin by showing that transpositions of adjacent steps preserve fairness.

Lemma 3.3.13. *Let $\mathcal{R}$ be interference-free from M_0 and let $T = (M_0, (r_i; (\theta_i, \xi_i))_{i \in I})$ be a trace, an execution, or a fair trace. For all transpositions $(m+1, m) \in S_I$, if $r_{m+1}(\theta_{m+1})$ is applicable to M_{m-1}, then $(m+1, m) \cdot T$ is respectively a trace, an execution, or a fair trace.*

Proof. Consider some transpositions $(m+1, m) \in S_I$ such that $r_{m+1}(\theta_{m+1})$ is applicable to M_{m-1}. By non-interference, it follows that $(m+1, m) \cdot T = (M_0, (r'_i; (\theta'_i, \xi'_i))_i)$ is also a trace. Observe that for all j, if $j \neq m$, then $M'_j = M_j$, and if $j < m$ or $j > m+1$, then $(r'_j; (\theta'_j, \xi'_j)) = (r_j; (\theta_j, \xi_j))$.

Assume that T is an execution. We must show that $(m+1, m) \cdot T$ is also maximal. If it is infinite, then we are done. If it ends at some M'_n, then $M'_n = M_n$ by the above observation because $m < m+1 \leq n$. Because T is maximal, no rules are applicable to M_n, so no rules are applicable to M'_n. We conclude that $(m+1, m) \cdot T$ is maximal.

Assume now that T is fair. We show that $(m+1, m) \cdot T$ is also fair. Consider some $r(\theta)$ applicable to M'_j. We proceed by case analysis on $j < m$, $j = m$, and $j > m$ to show that $r(\theta)$ appears as some $r'_k(\theta'_k)$ with $k > j$.

Case $j < m$**:** By the above observations, $M'_j = M_j$. Because T is fair, there exists a $k' > j$ such that $r_{k'}(\theta_{k'}) \equiv r(\theta)$. Because $j < m$ and $j < k$, it follows that $k = (m+1, m)(k') > j$. So $r(\theta)$ appears as $r'_k(\theta'_k)$ after M'_j as desired.

Case $j = m$**:** By proposition 3.3.3, $r(\theta)$ is applicable to M_{m+1}. Because T is fair, there exists a $k > m+1$ such that $r_k(\theta_k) \equiv r(\theta)$. So $r'_k(\theta'_k) = r_k(\theta_k) \equiv (r, \theta)$.

Case $j > m$**:** Because T is fair, there exists a $k > j$ such that $r_k(\theta_k) \equiv r(\theta)$. Because $k > j > m$, $k > m+1$, so $r'_k(\theta'_k) = r_k(\theta_k) \equiv (r, \theta)$.

We conclude that $(m+1, m) \cdot T$ is fair whenever T is fair. □

Proposition 3.3.14. *Let $\mathcal{R}$ be interference-free from M_0 and let $T = (M_0, (r_i; (\theta_i, \xi_i))_{i\in I})$ be a trace, an execution, or a fair trace. For all cycles $(m+k, \dots, m+1, m) \in S_I$ with $k \geq 1$, if $r_{(m+k)}(\theta_{m+k})$ is applicable to M_{m-1}, then $(m+k, \dots, m+1, m) \cdot T$ is respectively a trace, an execution, or a fair trace. It is equal to T after the $(m+k)$-th step.*

Proof. By induction on k. If $k = 1$, then we are done by lemma 3.3.13 and non-interference. Assume the result for some k', and consider the case $k = k' + 1$. Because $\mathcal{R}$ is interference-free from M_0, it follows that if $r_{(m+k)}(\theta_{m+k})$ is applicable to M_{m-1}, then it is also applicable to M_m. By the induction hypothesis, $T' = (m+k, \dots, m+1) \cdot T$ is respectively a trace, an execution, or a fair trace. By lemma 3.3.13, so is $T'' = (m, m+1) \cdot T'$, for the $(m+1)$-th step in T' is $r_{(m+k)}(\theta_{m+k})$, and it is assumed to be applicable to M_{m-1}. The transposition $(m+1, m)$ does not alter T' beyond the $(m+k)$-th step, so T'' still agrees with T after the $(m+k)$-th step. Observe that

$$T'' = (m+1, m) \cdot T' = ((m+k, m) \circ (m+k, \dots, m+1)) \cdot T = (m+k, \dots, m+1, m) \cdot T,$$

so we are done. We note that the second equality in the above sequence is subtle: the transposition $(m+1, m)$ is relative to the ordering of rules in T'. It becomes $(m+k, m)$ on the right of the equality because the $(m+1)$-th step in T' is the $(m+k)$-th step of T. □

We conclude that finite permutations preserve fairness:

Proposition 3.3.15. *Let $\mathcal{R}$ be interference-free from M_0 and let $T = (M_0, (r_i; (\theta_i, \xi_i))_{i\in I})$ be a trace. Let $\sigma \in S_I$ be a finite permutation, i.e., assume that there exists an $n \in I$ such that $\sigma(i) = i$ for all $i > n$. Further assume that $\sigma \cdot T$ is a trace. If T is an execution or a fair trace, then $\sigma \cdot T$ is respectively an execution or a fair trace. The traces $\sigma \cdot T$ and T are equal after n-th step.*

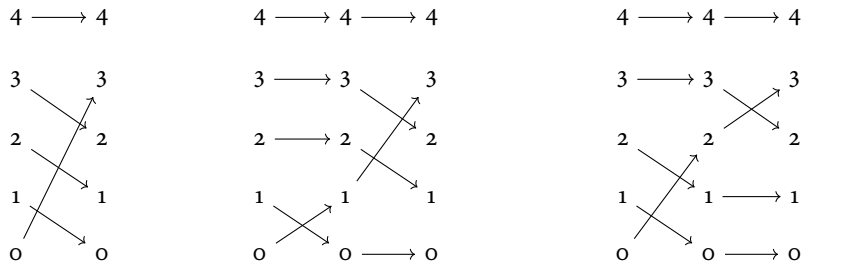
Proof. Informally, the approach is to decompose σ into a finite composition of cyclic permutations of the form $(\sigma(m), \dots, m+1, m)$. Proposition 3.3.14 ensures that each of these cycles preserves the desired properties.

Let m be minimal in I such that $\sigma(m) \neq m$; if no such m exists, i.e., if σ is the identity, then set $m = n$. We proceed by strong induction on $d = n - m$. If $d = 0$, then $\sigma \cdot T = T$ and we are done. Assume the result for some d', and consider the case $d = d' + 1$. By minimality of m, the traces T and $\sigma \cdot T$ are equal up until the multiset M_{m-1}. By the assumption that T and $\sigma \cdot T$ are both traces, it follows that $r_m(\theta_m)$ and $r_{\sigma(m)}(\theta_{\sigma(m)})$ are both applicable to M_{m-1}. By minimality, it also follows that $m < \sigma(m)$. By proposition 3.3.14, $T' = (\sigma(m), \dots, m+1, m) \cdot T$ is also respectively a trace, an execution, or a fair trace. By the same proposition, it also agrees with T on all multisets after the $\sigma(m)$-th. Since $\sigma(m) < n$, it follows that T' and T are equal after the n-th step. The trace T' agrees with $\sigma \cdot T$ on at least the first m steps. This decreases d by at least one. We conclude the result by the strong induction hypothesis on $(\sigma(m), \dots, m+1, m) \cdot T$ and $\sigma \cdot T$. □

Next, we show that infinite permutations preserve fairness. To do so, we use the following lemma to reduce arguments about infinite permutations to arguments about finite permutations. Intuitively, it decomposes any infinite permutation σ on $\mathbb{N}$ into the composition of a permutation τ that only permutes natural numbers less than some $\chi_\sigma(n)$, and of an infinite permutation ρ that only permutes natural numbers greater than $\chi_\sigma(n)$. The mechanics of the decomposition are best understood by means of a picture. We refer the reader to fig. 3.6 for an illustration, where we note that $\chi_\sigma(0) = 1$ and $\chi_\sigma(1) = 2$.

Lemma 3.3.16. *For all $n \in \mathbb{N}$ and permutations $\sigma : \mathbb{N} \to \mathbb{N}$, set $\chi_\sigma(n) = \max_{k\leq n} \sigma^{-1}(k)$. Then there exist permutations $\tau_n, \rho_n : \mathbb{N} \to \mathbb{N}$ such that $\sigma = \rho_n \circ \tau_n$, $\tau_n(k) = k$ for all $k > \chi_\sigma(n)$, and $\rho_n(k) = k$ for all $k \leq n$.*

Proof. Let $s_1 < \cdots < s_m$ be the elements of $\{0, 1, \dots, \chi_\sigma(n)\} \setminus \sigma^{-1}(\{0, \dots, n\})$. Explicitly, these are the natural numbers less than $\chi_\sigma(n)$ whose image under σ is greater than n.



(A) A permutation σ (B) Its decomposition with $n = 0$ (C) Its decomposition with $n = 1$

FIGURE 3.6. An illustration of decompositions of σ given by lemma 3.3.16

Let τ be given by

$$\tau(k) = \begin{cases} \sigma(k) & \sigma(k) \leq n \\ n + j & k = s_j \\ k & k > \chi_\sigma(n). \end{cases}$$

Intuitively, τ acts as σ on elements whose image is less than n; it then "stacks" the remaining elements less than $\chi_\sigma(n)$ in order on top of n (we refer the reader to fig. 3.6 for this spatial intuition); and it fixes elements greater than $\chi_\sigma(n)$.

Let ρ be given by

$$\rho(k) = \begin{cases} k & k \leq n \\ \sigma(s_{k-n}) & n < k \leq \chi_\sigma(n) \\ \sigma(k) & k > \chi_\sigma(n). \end{cases}$$

Intuitively, ρ takes the "stacked" elements and recovers their original value before applying σ; it directly applies σ to those greater than $\chi_\sigma(n)$.

We show that $\sigma = \rho \circ \tau$. Let $k \in \mathbb{N}$ be arbitrary. We proceed by case analysis:

CASE $\sigma(k) \leq n$**:** Then $\rho(\tau(k)) = \rho(\sigma(k))) = \sigma(k)$.

CASE $k = s_j$**:** Then $\rho(\tau(k)) = \rho(n + j) = \sigma(s_{(n+j)-n}) = \sigma(s_j) = \sigma(k)$.

CASE $k > \chi_\sigma(n)$**:** Then $\rho(\tau(k)) = \rho(k) = \sigma(k)$.

The function τ is clearly total and an isomorphism. Because σ is an isomorphism and $\sigma = \rho \circ \tau$, it follows that ρ is also an isomorphism. So τ and ρ are two permutations with the desired property. □

LEMMA 3.3.17. *Assume that $\mathcal{R}$ is interference-free from M_0, that T is a fair trace from M_0, and that $\sigma \cdot T$ is a permutation of T. For all n, $\tau_n \cdot T$ is a fair trace, where $\sigma = \rho_n \circ \tau_n$ is the decomposition given by lemma 3.3.16.*

Proof. This proof is analogous to the proof of proposition 3.3.15. Informally, we decompose τ_n as a potentially empty composition of cyclic permutations such that each successive cyclic permutation increases the length of the prefix of the trace that agrees with $\tau_n \cdot T$ and preserves the relative ordering of the s_i described in the proof of lemma 3.3.16.

Let m be minimal such that $\tau_n(m) \neq m$; if no such m exists, then $\tau_n \cdot T = T$ and we are done. Otherwise, the $\sigma(m)$-th step $r_{\sigma(m)}(\theta_{\sigma(m)})$ of T is applicable to M_{m-1}. Indeed, $\sigma(m) = \tau_n(m)$, and $\sigma \cdot T$ is a trace by assumption. By proposition 3.3.14, $T' = (\sigma(m), \ldots, m + 1, m) \cdot T$ is a fair trace. The trace T' agrees with $\tau_n \cdot T$ on at least the first m steps. Moreover, the relative ordering of the s_i in T' is the same as in T.

Iterating this procedure results in a trace T'' that agrees with $\tau_n \cdot T$ on the first n steps. Indeed, this procedure terminates because after each iteration, the number of steps in the first n that disagree decreases by one. The resulting trace T'' also agrees with $\tau_n \cdot T$ on all steps after the $(\chi_\sigma(n))$-th. Indeed, for each cycle $(\sigma(m), \ldots, m + 1, m)$, $\sigma(m) \leq \chi_\sigma(n)$. Because both τ_n and

the above procedure preserves the relative ordering of the s_i, and both result in permutations, it follows that their images agree on all of the steps between the n-th and the $\chi_\sigma(n)$-th. So T'' and $\tau_n \cdot T$ are equal. Because T'' is fair, we conclude that $\tau_n \cdot T$ is fair. □

Corollary 3.3.18. *Fairness is invariant under permutation, that is, if $\mathcal{R}$ is interference-free from M_0, T is a fair trace from M_0, and $\Sigma = \sigma \cdot T$ is a permutation of T, then Σ is also fair.*

Proof. Let $T = (M_0, (t_i; \delta_i)_i)$, and let Σ be the trace $M_0 = \Sigma_0 \xrightarrow{(t_{\sigma(1)};\delta_{\sigma(1)})} \Sigma_1 \xrightarrow{(t_{\sigma(2)};\delta_{\sigma(2)})} \cdots$. Consider some rule $r \in \mathcal{R}$ and θ such that $r(\theta)$ is applicable to Σ_i. We must show that there exists a j such that $\sigma(j) > \sigma(i)$, $t_{\sigma(j)}(\theta_{\sigma(j)}) \equiv r(\theta)$.

Let the factorization $\sigma = \rho \circ \tau$ be given by lemma 3.3.16 for $n = \sigma(i)$. The trace $\tau \cdot T$ is fair by lemma 3.3.17. By construction of τ, $\tau \cdot T$ and Σ agree on the first n steps and $n+1$ multisets. By fairness, there exists a $k > \sigma(i)$ such that the k-th step in $\tau \cdot T$ is $r(\theta)$. By construction of ρ, $\rho(k) > \sigma(i)$, so this step appears after Σ_i in Σ as desired. We conclude that Σ is fair. □

It is not the case that every permutation of the steps of a fair trace is a fair trace: it could fail to be a trace. Corollary 3.3.18 simply states that if the result of permuting the steps of a fair trace is a trace, then that trace is fair.

Corollary 3.3.18 established that permutations preserve fairness. Relatedly, all fair traces from a given multiset are permutations of each other. To show this, we construct a potentially infinite sequence of permutations. We use the following lemma to compose them:

Lemma 3.3.19. *Let $(\sigma_n)_{n \in I}$ be a family of bijections on I such that for all $m < n$,*

$$(\sigma_n \circ \cdots \circ \sigma_1)(m) = (\sigma_m \circ \cdots \circ \sigma_1)(m).$$

Let $\sigma : I \to I$ be given by $\sigma(m) = (\sigma_m \circ \cdots \circ \sigma_1)(m)$. Then σ is injective, but need not be surjective.

Proof. Let $m, n \in I$ be arbitrary such that $\sigma(m) = \sigma(n)$. Assume without loss of generality that $m \leq n$. Observe that

$$\sigma(m) = (\sigma_m \circ \cdots \circ \sigma_0)(m) = (\sigma_n \circ \cdots \circ \sigma_m \circ \cdots \circ \sigma_0)(m)$$

and

$$\sigma(n) = (\sigma_n \circ \cdots \circ \sigma_m \circ \cdots \circ \sigma_0)(n).$$

Because $\sigma_0, \ldots, \sigma_n$ are all bijections, so is their composition. It follows that $m = n$, so σ is injective.

To see that σ need not be surjective, consider the family $\sigma_n = (0, n)$ for $n \geq 1$. Then $\sigma(n) = n+1$ for all n. It follows that 0 is not in the image of σ. □

Recall from section 3.2.5 that, given an über fair trace T and an instantiation $t(\tau)$ applicable to its i-th multiset, $v_T(i, t, \tau)$ is the least $j > i$ such that the j-th step of T is equivalent to $t(\tau)$. The following lemma is a special case of proposition 3.3.14:

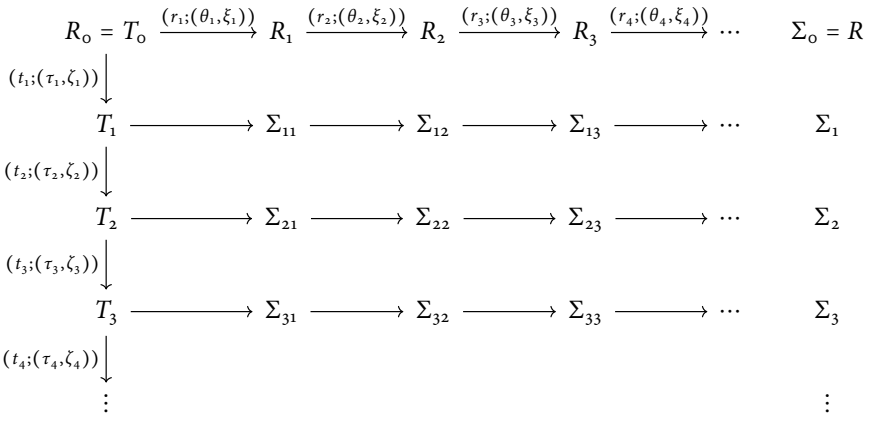
Lemma 3.3.20. *Let $\mathcal{R}$ be interference-free from M_0 and T a fair execution from M_0. If $t(\tau)$ is applicable to M_0, then $(v_T(0, t, \tau), \ldots, 0) \cdot T$ is a permutation of T with $t(\tau)$ as its first step, and it is a fair execution.*

Proposition 3.3.21. *If $\mathcal{R}$ is interference-free from M_0, then all fair executions from M_0 are permutations of each other.*

Proof. Consider traces $R = (R_0, (r_i; (\theta_i, \xi_i))_{i \in I})$ and $T = (T_0, (t_j; (\tau_j, \zeta_j))_{j \in J})$ where $R_0 = M_0 = T_0$. We construct a sequence of permutations $\sigma_0, \sigma_1, \ldots$, where $\Phi_0 = R$ and the step $\Phi_{n+1} = \sigma_{n+1} \cdot \Phi_n$ is given by lemma 3.3.20 such that Φ_{n+1} agrees with T on the first $n+1$ steps. We then assemble these permutations σ_n into an injection σ using lemma 3.3.19; fairness ensures that it is a surjection. We have $T = \sigma \cdot R$ by construction.

The construction is as follows. We write $\Sigma_0 = R$ horizontally, and then down from R_0, we write T. Let Γ_{n+1} be the fair trace obtained by applying lemma 3.3.20 to Σ_n and $(t_{n+1}; (\tau_{n+1}, \zeta_{n+1}))$; it is a permutation of Σ_n. Without loss of generality, we assume that the fresh constant substitutions ζ_{n+1}

and the corresponding ξ_k in $\Sigma_n, \ldots, \Sigma_0$ are equal; refreshing both S and T makes this possible. Let Σ_{n+1} be the tail of Γ_{n+1} starting at T_{n+1}; we write it horizontally to the right of T_{n+1}. We get the following picture:



Set $\Phi_0 = \Sigma_0$, and for each $n > 0$, let Φ_n be the trace given by the trace $T_0 \to \cdots \to T_n$ followed by the trace Σ_n. Each of the permutations $\mu_n : \Sigma_n \to \Gamma_{n+1}$ determines a permutation σ_n from Φ_n to Φ_{n+1} that fixes the first n steps. The family of these permutations satisfies the hypothesis of lemma 3.3.19, and gives us an injection $\sigma : I \to J$. It is a surjection by construction: $t_n(\tau_n)$ appears as some $r_k(\theta_k)$ by fairness, and $\sigma(k) = n$. To see that $\sigma \cdot R = T$, it is sufficient to observe that for all n, $\sigma \cdot R$ and T agree on the first n steps. □

Definition 3.3.22. Two traces $T = (M_0; (r_i, \delta_i)_I)$ and T' are **union-equivalent** if T' can be refreshed to a trace $[\eta]T'$ such that $\mathsf{supp}(T) = \mathsf{supp}([\eta]T')$. ◂

Lemma 3.3.23. *If T is a permutation of S, then T and S are union-equivalent.*

Proof. Consider a trace $(M_0, (r_i; \delta_i)_i)$. For all n, each fact in M_n appears either in M_0 or in the result of some rule r_i with $i \leq n$. Traces T and S start from the same multiset and have the same rule instantiations. It follows that they are union-equivalent. □

Corollary 3.3.24 will be key to showing in chapter 6 that processes have unique observations.

Corollary 3.3.24. *If $\mathcal{R}$ is interference-free from M, then all fair executions from M are union-equivalent.*

3.4. Related Work

Multiset rewriting systems with existential quantification were first introduced by Cervesato et al. [Cer+99]. They were used to study security protocols and were identified as the first-order Horn fragment of linear logic. Since, MRSs have modelled other security protocols, and strand spaces [Cer+00; Cer+05]. Cervesato and Scedrov [CS09] studied the relationship between MRSs and linear logic. These works do not explore fairness.

Substructural operational semantics [Sim12] based on multiset rewriting are widely used to specify the operational behaviour of session-typed languages arising from proofs-as-processes interpretations of linear logic and adjoint logic. Examples include functional languages with session-typed concurrency [TCP13], languages with run-time monitoring [GJP18], message-passing interpretations of adjoint logic [PP19a], and session-typed languages with sharing [BP17].

Fairness finds its roots in work of Lamport [Lam77] and Park [Par80]. Lamport [Lam77] studied the correctness of multiprocessor programs. He described fairness constraints on schedulers using clocks, where process clocks were assumed to advance a certain amount in every period of real time. Instead of using clocks, Park [Par80] defined fairness using a "fair merge" operator on traces. Early work [LPS81; Par80] on fairness was concerned with fair termination. Francez [Fra86]

gave a comprehensive analysis of fair termination. Weak and strong fairness were introduced by Apt and Olderog [AO82] and Park [Par82] in the context of do-od languages. Fairness was subsequently adapted to process calculi, e.g., by Grumberg, Francez, and Katz [GFK84] for CSP-like languages and by Costa and Stirling [CS87] for Milner's CCS. Hennessy [Hen87] studied fairness in the setting of asynchronous communicating processes. Leu et al. [Leu+88] introduced fairness for Petri nets. Kwiatkowska [Kwi89] surveys these notions and others, and gives a taxonomy of varieties of fairness. Bounded fairness [DJP03] places bounds on how long we must wait before an event occurs.

Fair scheduling algorithms are an active research area in the programming languages and systems communities. For example, Sistla [Sis83] studied the complexity of fair scheduling algorithms. Henry [Hen84] gave a fair scheduler for processes on UNIX systems. Muller, Westrick, and Acar [MWA19] and Muller, Acar, and Harper [MAH18] studied fair scheduling for interactive computation and in the presence of priorities. Lahav et al. [Lah+20] gave an account of process fairness under weak memory models.

CHAPTER 4

Fixed Points of Functors

Recursive types are ubiquitous in functional languages. For example, in Standard ML we can define the type of (unary) natural numbers as:

```
datatype nat = Zero | Succ of nat
```

This declaration specifies that a `nat` is either zero or the successor of some natural number. Semantically, we can think of `nat` as a domain D satisfying the domain equation

$$D \cong (\texttt{Zero} : \{\bot\}) \uplus (\texttt{Succ} : D),$$

where $\uplus$ forms the labelled disjoint union of domains. Equivalently, we can think of D as a fixed point of the functor $F_{\texttt{nat}}(X) = (\texttt{Zero} : \{\bot\}) \uplus (\texttt{Succ} : X)$ on a category of domains.

Mutually-recursive data types give rise to a similar interpretation. Consider, for example, the types of even and odd natural numbers:

```
datatype even = Zero | E of odd
     and odd  = O of even
```

This declaration specifies that an even number is either zero or the successor of an odd number, and that an odd number is the successor of an even one. The types `even` and `odd` respectively denote solutions $D_{\mathtt{e}}$ and $D_{\mathtt{o}}$ to the system of domain equations:

$$D_{\mathtt{e}} \cong (\texttt{Zero} : \{\bot\}) \uplus (\texttt{E} : D_{\mathtt{o}}) \quad \text{and} \quad D_{\mathtt{o}} \cong (\texttt{O} : D_{\mathtt{e}}).$$

These are solutions to the system of equations:

$$X_{\mathtt{e}} \cong F_{\texttt{even}}(X_{\mathtt{e}}, X_{\mathtt{o}}) \tag{26}$$

$$X_{\mathtt{o}} \cong F_{\texttt{odd}}(X_{\mathtt{e}}, X_{\mathtt{o}}) \tag{27}$$

where $F_{\texttt{even}}$ and $F_{\texttt{odd}}$ are the functors $F_{\texttt{even}}(X_{\mathtt{e}}, X_{\mathtt{o}}) = (\texttt{Zero} : \{\bot\}) \uplus (\texttt{E} : X_{\mathtt{o}})$ and $F_{\texttt{odd}}(X_{\mathtt{e}}, X_{\mathtt{o}}) = (\texttt{O} : X_{\mathtt{e}})$. We can use Bekič's rule [Bek84, § 2] to solve this system of equations. To do so, we think of eq. (26) as a family of equations parametrized by $X_{\mathtt{o}}$. If we could solve for $X_{\mathtt{e}}$, then we would get a parametrized family of solutions $F^{\dagger}_{\texttt{even}}(X_{\mathtt{o}})$ such that:

$$F^{\dagger}_{\texttt{even}}(X_{\mathtt{o}}) \cong F_{\texttt{even}}(F^{\dagger}_{\texttt{even}}(X_{\mathtt{o}}), X_{\mathtt{o}}) \tag{28}$$

for all domains $X_{\mathtt{o}}$. Substituting this for $X_{\mathtt{e}}$ in eq. (27) gives the domain equation

$$X_{\mathtt{o}} \cong F_{\texttt{odd}}(F^{\dagger}_{\texttt{even}}(X_{\mathtt{o}}), X_{\mathtt{o}}).$$

Solving for $X_{\mathtt{o}}$ gives the solution $D_{\mathtt{o}}$. Substituting $D_{\mathtt{o}}$ for $X_{\mathtt{o}}$ in eq. (28), we see that $D_{\mathtt{e}} = F^{\dagger}_{\texttt{even}}(D_{\mathtt{o}})$ is the other part of the solution.

The above example motivates techniques for solving parametrized domain equations. These techniques are well understood. For example, given a suitable functor $F : \mathbf{D} \times \mathbf{E} \to \mathbf{E}$ on suitable categories of domains, [AJ95, Proposition 5.2.7] gives a recipe for constructing a functor $F^{\dagger} : \mathbf{D} \to \mathbf{E}$ such that for all objects D of $\mathbf{D}$, $F^{\dagger}D \cong F(D, F^{\dagger}D)$. Is the mapping $F \mapsto F^{\dagger}$ functorial? Semantically, substitution is typically interpreted as composition [Cro93, § 3.4]. If the interpretations of recursive types are to respect substitution, then the mapping $F \mapsto F^{\dagger}$ must be natural in $\mathbf{D}$. Is it? What other properties does it satisfy?

Families of parametric fixed points arise elsewhere in mathematics. An external **dagger operation** [BÉ95, Definition 2.6; BÉ96, p. 7] on a cartesian closed category $\mathbf{C}$ is a family $\dagger_{A,B}$:

$\mathbf{C}(A \times B, B) \to \mathbf{C}(A, B)$ of set-theoretic functions for each pair of objects A and B in $\mathbf{C}$. Of particular interest are dagger operations that satisfy the (cartesian) Conway identities These identities imply many other identities [BÉ96, § 3.3] useful for semantic reasoning, such as Bekič's rule. They are also of independent interest. Indeed, they axiomatize a decidable theory [BÉ98], and dagger operations that satisfy them are closely related to the trace operator [JSV96; Has99, Theorem 7.1; BH03, p. 281]. Does the above dagger operation satisfy the Conway identities?

In this chapter, we present dagger operators in two different categorical settings. In sections 4.2 to 4.4, we work with ω-cocontinuous functors between categories with sufficiently many ω-colimits. In section 4.5, we work with locally continuous functors between $\mathbf{O}$-categories. $\mathbf{O}$-categories [SP82] generalize categories of domains to provide just the structure required to compute fixed points of functors and to show the limit-colimit coincidence theorem (theorem 2.2.53). In both cases, these dagger operators will enjoy a 2-categorical structure that will imply the Conway identities. As an application, in chapter 8 we see that properties of our dagger operation are essential for defining and reasoning about semantics of session-typed languages with recursion.

4.1. Background

Recall from section 2.2.1.1 that ω is the category with natural numbers as objects and with at most one arrow between each pair of objects, where $n \to m$ if and only if $n \leq m$. An **ω-chain** in a category $\mathbf{K}$ is a diagram $J : \omega \to \mathbf{K}$. An **$\omega$-category** [Leh76a, Definition 5] is a category with all colimits of ω-diagrams. We warn the reader that the definition of ω-category varies in the literature. Some [LS81, Definition 2.4] additionally require the existence of an initial object; we call such categories **IFP**-categories. This name stems from the fact that **IFP**-categories have the structure required for ω-functors to have initial fixed points (see corollary 4.2.14 below). An **ω-functor** [LS81, Definition 2.5] is an ω-cocontinuous functor, i.e., a functor that preserves all existing colimits of ω-diagrams. Small ω-categories and ω-functors between them form a 2-cartesian-closed subcategory ω-**Cat** of **Cat**. Small **IFP**-categories and ω-functors between them form a 2-cartesian-closed subcategory **IFP** of ω-**Cat**. A **parameterized ω-functor** is a functor $F : \mathbf{C} \times \mathbf{D} \to \mathbf{E}$ such that $F(C, -) : \mathbf{D} \to \mathbf{E}$ is an ω-functor for all objects C of $\mathbf{C}$.

4.2. Functoriality of Fixed Points

We show that constructing fixed points of ω-functors is itself a functorial operation. The initial fixed point of an ω-functor F on an ω-category is given by the colimit of the ω-chain $\bot \to F\bot \to F^2\bot \to \cdots$. Other fixed points can be constructed using a different "first link", i.e., by taking the colimit of a chain $K \to FK \to F^2K \to \cdots$ generated by a link $k : K \to FK$.

Recall that, given natural transformations $\eta : F \Rightarrow G : \mathbf{C} \to \mathbf{D}$ and $\phi : H \Rightarrow I : \mathbf{D} \to \mathbf{E}$, their horizontal composition is the natural transformation $\eta * \phi : FH \Rightarrow GI : \mathbf{C} \to \mathbf{E}$. Given a morphism $f : K \to L$ in $\mathbf{C}$, we abuse notation and write $\eta * f : FK \to GL$ for the naturality square $FK \xrightarrow{Ff} FL \xrightarrow{\eta_L} GL = FK \xrightarrow{\eta_K} GK \xrightarrow{Gf} GK$.

Definition 4.2.1. Fix an ω-category $\mathbf{K}$. Links form a category $\mathbf{Links}_\mathbf{K}$ where

- objects are triples (K, k, F) called "links", where K is an object of $\mathbf{K}$, $F : \mathbf{K} \to \mathbf{K}$ is an ω-functor, and $k : K \to FK$ is a morphism in $\mathbf{K}$;
- morphisms $(K, k, F) \to (L, l, G)$ are pairs (f, η) where $f : K \to L$ is a morphism of $\mathbf{K}$, $\eta : F \Rightarrow G$ is a natural transformation, and f and η satisfy $l \circ f = (\eta * f) \circ k : K \to GL$;
- composition is given component-wise: $(g, \rho) \circ (f, \eta) = (g \circ f, \rho \circ \eta)$. ◂

The condition on morphisms between links provide the structure required to define morphisms between the ω-chains they generate.

Proposition 4.2.2. *If* $\mathbf{K}$ *has an initial object, then the link* $(\bot, \bot, \Delta\bot)$ *is the initial link in* $\mathbf{Links}_\mathbf{K}$.

Proof. The object $\bot$ is initial in $\mathbf{K}$, and the constant functor $\Delta\bot$ is initial in $\mathbf{IFP}\,[\mathbf{K} \to \mathbf{K}]$. It follows that for every (L, l, G), there is at most one morphism $(\bot, \bot, \Delta\bot) \to (L, l, G)$. We check that $(\bot_L, \bot_G)$ is such a morphism. For every (L, l, G), initiality implies

$$\begin{array}{ccc} \bot & \xrightarrow{\bot} & \Delta\bot \\ {\scriptstyle \bot}\downarrow & & \downarrow{\scriptstyle \bot_G * \bot_L} \\ L & \xrightarrow{l} & GL \end{array}$$

commutes. We conclude that $(\bot_L, \bot_G)$ is a morphism in $\mathbf{Links}_{\mathbf{K}}$, and that $(\bot, \bot, \Delta\bot)$ is initial. □

The category $\mathbf{Links}_{\mathbf{K}}$ is ω-cocomplete, and proposition 4.2.4 below characterizes its ω-colimits. We first prove the following lemma. It extends [LS81, Lemma 4.3] to also specify the action of mediating morphisms of cocones:

Lemma 4.2.3. *Let* $\mathbf{C}$ *be a category and let* $T = F_0 \overset{\tau_0}{\Longrightarrow} F_1 \overset{\tau_1}{\Longrightarrow} \cdots$ *and* $T' = F'_0 \overset{\tau'_0}{\Longrightarrow} F'_1 \overset{\tau'_1}{\Longrightarrow} \cdots$ *be* ω*-chains in the functor category* $\mathbf{Cat}\,[\mathbf{C} \to \mathbf{C}]$. *Assume they have colimit cocones* $\sigma : T \Rightarrow F_\infty$ *and* $\sigma' : T' \Rightarrow F'_\infty$, *respectively. Then the* ω*-chain* $T'' = F'_0 F_0 \overset{\tau'_0 * \tau_0}{\Longrightarrow} F'_1 F_1 \overset{\tau'_1 * \tau_1}{\Longrightarrow} \cdots$ *has the colimit cocone* $\sigma' * \sigma : T'' \Rightarrow F'_\infty F_\infty$.

Let $\gamma : T \Rightarrow G$ *and* $\gamma' : T' \Rightarrow G'$ *be arbitrary cocones, and let* $\phi : (\sigma, F_\infty) \to (\gamma, G)$ *and* $\phi' : (\sigma', F'_\infty) \to (\gamma', G')$ *be the unique mediating morphisms of cocones. Then their horizontal composition* $\phi' * \phi : (\sigma' * \sigma, F'_\infty F_\infty) \to (\gamma' * \gamma, G', G)$ *is the unique mediating morphism of cocones on* T''.

Proof. See [LS81, Lemma 4.3] for the proof of the first paragraph. The second paragraph is an obvious corollary of the first. □

Let $\mathbf{2}$ be the category $(\bullet \to \bullet)$.

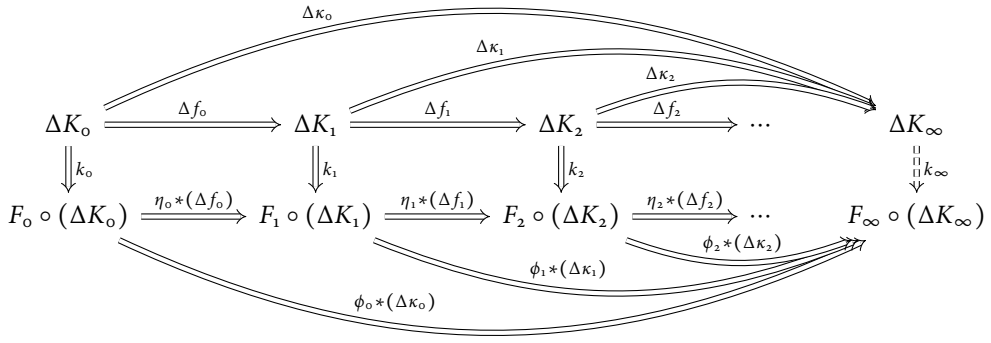
Proposition 4.2.4. *Let* $J = (K_0, k_0, F_0) \xrightarrow{(f_0, \eta_0)} (K_1, k_1, F_1) \xrightarrow{(f_1, \eta_1)} \cdots$ *be an* ω*-chain in* $\mathbf{Links}_{\mathbf{K}}$. *Let* (κ, K_∞) *be the colimit of the* ω*-chain* $K = K_0 \xrightarrow{f_0} K_1 \xrightarrow{F_1} \cdots$ *in* $\mathbf{K}$. *Let* (ϕ, F_∞) *be the colimit of the* ω*-chain* $\Phi = F_0 \xrightarrow{\eta_0} F_1 \xrightarrow{\eta_1} \cdots$ *in* ω-$\mathbf{Cat}\,[\mathbf{K} \to \mathbf{K}]$. *Then there exists a* $k_\infty : K_\infty \to F_\infty K_\infty$ *such that* $(\kappa, \phi) : J \Rightarrow (K_\infty, k_\infty, F_\infty)$ *is colimiting in* $\mathbf{Links}_{\mathbf{K}}$.

Proof. We can recognize each link (K_n, k_n, F_n) as a $\mathbf{2}$-diagram $\Delta K_n \overset{k_n}{\Longrightarrow} F_n \circ (\Delta K_n)$ in the category ω-$\mathbf{Cat}\,[\mathbf{K} \to \mathbf{K}]$. Furthermore, we can recognize J as a diagram $\hat{J} : \omega \to \mathbf{Cat}\,[\mathbf{2} \to \omega\text{-}\mathbf{Cat}\,[\mathbf{K} \to \mathbf{K}]]$:

$$\begin{array}{ccccccc} \Delta K_0 & \overset{\Delta f_0}{\Longrightarrow} & \Delta K_1 & \overset{\Delta f_1}{\Longrightarrow} & \Delta K_2 & \overset{\Delta f_2}{\Longrightarrow} & \cdots \\ \Downarrow{\scriptstyle k_0} & & \Downarrow{\scriptstyle k_1} & & \Downarrow{\scriptstyle k_2} & & \\ F_0 \circ (\Delta K_0) & \overset{\eta_0 * (\Delta f_0)}{\Longrightarrow} & F_1 \circ (\Delta K_1) & \overset{\eta_1 * (\Delta f_1)}{\Longrightarrow} & F_2 \circ (\Delta K_2) & \overset{\eta_2 * (\Delta f_2)}{\Longrightarrow} & \cdots \end{array}$$

Colimits in ω-$\mathbf{Cat}[\mathbf{K} \to \mathbf{K}]$ are determined component-wise, so the top row has colimit $(\Delta\kappa, \Delta K_\infty)$. By lemma 4.2.3, the bottom row has colimit $(\phi * \kappa, F_\infty \circ (\Delta K_\infty))$. Let $k = (k_n : \Delta K_n \to F_n \circ (\Delta K_n))$ be the natural transformation from the top row to the bottom row. Then $((\phi * \kappa) \circ k, F_\infty \circ (\Delta K_\infty))$ is a cocone on the top row, so there exists a unique cocone morphism

$k_\infty : (\Delta\kappa, \Delta K_\infty) \to ((\phi * \kappa) \circ k, F_\infty \circ (\Delta K_\infty))$:



In particular, $k_\infty : \Delta K_\infty \Rightarrow F_\infty \circ (\Delta K_\infty)$ is a natural transformation between two constant functors, so is given by a single morphism $K_\infty \to F_\infty K_\infty$ in $\mathbf{K}$. We conclude that $(K_\infty, k_\infty, F_\infty)$ is an object of $\mathbf{Links}_\mathbf{K}$. It is immediate from the above diagram that $(\kappa, \phi) : J \Rightarrow (K_\infty, k_\infty, F_\infty)$ is a cocone in $\mathbf{Links}_\mathbf{K}$.

We show that this cocone is colimiting in $\mathbf{Links}_\mathbf{K}$. Let $(\alpha, \beta) : J \Rightarrow (A, c, B)$ be any other cocone in $\mathbf{Links}_\mathbf{K}$. We begin by showing that there exists a cocone morphism $(a, b) : (K_\infty, k_\infty, F_\infty) \to (A, c, B)$. Observe that (α, β) determines a cocone $\alpha : K \Rightarrow A$ in $\mathbf{K}$, so a unique cocone morphism $a : (\kappa, K_\infty) \to (\alpha, A)$. It also determines a cocone $\beta : \Phi \Rightarrow B$ in $\omega\text{-}\mathbf{Cat}[\mathbf{K} \to \mathbf{K}]$, so a unique cocone morphism $b : (\phi, F_\infty) \to (\beta, B)$. Of course, (α, β) also induces a cocone $(\Delta\alpha, \beta * \Delta\alpha) : \hat{J} \Rightarrow (\Delta A \overset{c}{\Longrightarrow} B \circ (\Delta A))$. By lemma 4.2.3, the components of unique mediating $\hat{J}$-cocone morphism are given by the top and bottom row of the following diagram:

$$\begin{array}{ccc} \Delta K_\infty & \xRightarrow{\Delta a} & \Delta A \\ \Big\Downarrow{\scriptstyle k_\infty} & & \Big\Downarrow{\scriptstyle c} \\ F_\infty \circ (\Delta K_\infty) & \xRightarrow{b * (\Delta a)} & B \circ \Delta A \end{array}$$

These obviously induce a morphism $(a, b) : (K_\infty, k_\infty, F_\infty) \to (A, c, B)$ in $\mathbf{Links}_\mathbf{K}$.

To see uniqueness, consider any other cocone morphism $(a', b') : ((\kappa, \phi), (K_\infty, k_\infty, F_\infty)) \to ((\alpha, \beta), (A, c, B))$. Then $a' : (\kappa, K_\infty) \to (\alpha, A)$ is a cocone morphism in the category of cocones $\int \mathsf{Cone}(K, -)$, so $a' = a$. Analogously, $b' : (\phi, F_\infty) \to (\beta, B)$ is a cocone morphism in the category of cocones $\int \mathsf{Cone}(\Phi, -)$, so $b' = b$. We conclude that $(a', b') = (a, b)$, i.e., uniqueness of mediating morphism.

We conclude that $(\kappa, \phi) : J \Rightarrow (K_\infty, k_\infty, F_\infty)$ is colimiting. □

There exists a functor $\Omega : \mathbf{Links}_\mathbf{K} \to \omega\text{-}\mathbf{Cat}[\omega \to \mathbf{K}]$ that produces the ω-chain $K \xrightarrow{k} FK \xrightarrow{Fk} F^2K \xrightarrow{F^2k} \cdots$ from a link (K, k, F). Its action on morphisms uses the horizontal iteration of natural transformations. Consider functors $H, G : \mathbf{C} \to \mathbf{C}$ and a natural transformation $\eta : H \Rightarrow G$. We define the family of **horizontal iterates** $\eta^{(i)} : H^i \Rightarrow G^i$, $i \in \mathbb{N}$, by recursion on i. When $i = 0$, $H^0 = G^0 = \mathsf{id}_\mathbf{C}$ and we define $\eta^{(0)}$ to be the identity natural transformation on $\mathsf{id}_\mathbf{C}$. Given $\eta^{(i)}$, we set $\eta^{(i+1)} = \eta * \eta^{(i)}$.

We define the functor $\Omega : \mathbf{Links}_{\mathbf{K}} \to \omega\text{-}\mathbf{Cat}\,[\omega \to \mathbf{K}]$. The action of $\Omega(K, k, F)$ on morphisms $n \to n + k$ is defined by induction on k.

$$\Omega(K, k, F)(n) = F^n K \tag{29}$$

$$\Omega(K, k, F)(n \to n) = \mathsf{id}_{F^n K} \tag{30}$$

$$\Omega(K, k, F)(n \to n + k + 1) = F^{n+k} k \circ \Omega(K, k, F)(n \to n + k) \tag{31}$$

$$\Omega(f : K \to L, \eta : F \Rightarrow G)_n = \eta^{(n)} * f : F^n K \to G^n L \tag{32}$$

Proposition 4.2.7 generalizes the functor $S : \omega\text{-}\mathbf{Cat}\,[\mathbf{C} \to \mathbf{C}] \to \omega\text{-}\mathbf{Cat}\,[\omega \to \mathbf{C}]$ of [LS77, § 3; LS81, Lemma 4.2] to form chains with an arbitrary initial link in an ω-cocontinuous manner. Its existence depends on the following sequence of lemmas:

Lemma 4.2.5. *Let* $\mathbf{K}$ *be a category and* $F, G, H : \mathbf{K} \to \mathbf{K}$ *functors. Let* $\eta : F \Rightarrow G$ *and* $\rho : G \Rightarrow H$ *be natural transformations. Then for all* $n \geq 0$, $(\rho \circ \eta)^{(n)} = \rho^{(n)} \circ \eta^{(n)} : F^n \Rightarrow H^n$.

Proof. Let K be an object of $\mathbf{K}$. We proceed by induction on n to show that $(\rho \circ \eta)^{(n)}_K = (\rho^{(n)} \circ \eta^{(n)})_K : F^n K \to H^n K$. When $n = 0$, $F^0 K = K = H^0 K$ and $(\rho \circ \eta)^{(0)}_K = \mathsf{id}_K = (\rho^{(0)} \circ \eta^{(0)})_K$. Assume the result for some n, and consider the case $n + 1$. By the induction hypothesis and the middle four interchange law,

$$\begin{aligned}
&(\rho \circ \eta)^{(n+1)} \\
&= (\rho \circ \eta) * (\rho \circ \eta)^{(n)} \\
&= (\rho \circ \eta) * \left(\rho^{(n)} \circ \eta^{(n)}\right) \\
&= \left(\rho * \rho^{(n)}\right) \circ \left(\eta * \eta^{(n)}\right) \\
&= \rho^{(n+1)} \circ \eta^{(n+1)}.
\end{aligned}$$

We conclude the result by induction. □

Lemma 4.2.6. *Equations* (29) *to* (32) *define a functor* $\Omega : \mathbf{Links}_{\mathbf{K}} \to \omega\text{-}\mathbf{Cat}\,[\omega \to \mathbf{K}]$.

Proof. In this proof we show that:

(1) $\Omega(K, k, F)$ is a well-defined functor $\omega \to \mathbf{K}$ for all links (K, k, F);
(2) $\Omega(K, k, F)$ is an ω-functor for all links (K, k, F);
(3) $\Omega(f, \eta)$ is natural;
(4) Ω respects composition.

Let (K, k, F) be an arbitrary link and abbreviate $\Omega(K, k, F)$ by J. We must show that J is a well-defined functor $\omega \to \mathbf{K}^e$. It preserves identities by eq. (30). We must show that it respects composition. Let $l \to l+m$ and $l+m \to l+m+n$ be arbitrary, and note that $l \to l+m+n = (l+m \to l+m+n) \circ (l \to l+m)$. We must show that $J(l \to l+m+n) = J(l+m \to l+m+n) \circ J(l \to l+m)$. We proceed by nested strong induction on n. Assume first $n = 0$, then by eq. (30),

$$\begin{aligned}
&J(l \to l + m + n) \\
&= J(l \to l + m) \\
&= \mathsf{id}_{F^{l+m} K} \circ J(l \to l + m) \\
&= J(l + m \to l + m) \circ J(l \to l + m) \\
&= J(l + m \to l + m + n) \circ J(l \to l + m).
\end{aligned}$$

Now assume the result for some n, then by eq. (31),

$$\begin{aligned}
&J(l \to l+m+(n+1)) \\
&= F^{l+m+n}k \circ J(l \to l+m+n) \\
&= F^{l+m+n}k \circ (J(l+m \to l+m+n) \circ J(l \to l+m)) \\
&= (F^{l+m+n}k \circ J(l+m \to l+m+n)) \circ J(l \to l+m) \\
&= J(l+m \to l+m+(n+1)) \circ J(l \to l+m).
\end{aligned}$$

We conclude the result by induction.

We know that $\Omega(K,k,F)$ is ω-cocontinuous for all links (K,k,F) by proposition 4.A.16.

Next, we must show that Ω is well-defined on morphisms. Let $(f,\eta) : (K,k,F) \to (L,l,G)$ be arbitrary. We must show that $\Omega(f,\eta) : \Omega(K,k,F) \Rightarrow \Omega(L,l,G)$ is a natural transformation. Let $n \to n+m$ be an arbitrary morphism of ω. We must show that the following diagram commutes:

$$\begin{array}{ccc}
F^nK & \xrightarrow{\Omega(f,\eta)_n} & G^nL \\
{\scriptstyle \Omega(K,F,k)(n\to n+m)}\downarrow & & \downarrow{\scriptstyle \Omega(L,G,l)(n\to n+m)} \\
F^{n+m}K & \xrightarrow{\Omega(f,\eta)_{n+m}} & G^{n+m}L
\end{array} \tag{33}$$

We proceed by induction on m. Assume first that $m = 0$, then diagram 33 becomes

$$\begin{array}{ccc}
F^nK & \xrightarrow{\Omega(f,\eta)_n} & G^nL \\
{\scriptstyle \mathsf{id}}\downarrow & & \downarrow{\scriptstyle \mathsf{id}} \\
F^nK & \xrightarrow{\Omega(f,\eta)_n} & G^nL
\end{array}$$

and clearly commutes. Assume the result for some m, and consider the case $m+1$. We must show that the following diagram commutes:

$$\begin{array}{ccc}
F^nK & \xrightarrow{\Omega(f,\eta)_n} & G^nL \\
{\scriptstyle \Omega(K,F,k)(n\to n+m+1)}\downarrow & & \downarrow{\scriptstyle \Omega(L,G,l)(n\to n+m+1)} \\
F^{n+m+1}K & \xrightarrow{\Omega(f,\eta)_{n+m+1}} & G^{n+m+1}L
\end{array}$$

By eq. (31), this diagram is equal to the outer rectangle of the following diagram:

$$\begin{array}{ccc}
F^nK & \xrightarrow{\Omega(f,\eta)_n} & G^nL \\
{\scriptstyle \Omega(K,F,k)(n\to n+m)}\downarrow & & \downarrow{\scriptstyle \Omega(L,G,l)(n\to n+m)} \\
F^{n+m}K & \xrightarrow{\Omega(f,\eta)_{n+m}} & G^{n+m}L \\
{\scriptstyle F^{n+m}k}\downarrow & & \downarrow{\scriptstyle G^{n+m}l} \\
F^{n+m+1}K & \xrightarrow{\Omega(f,\eta)_{n+m+1}} & G^{n+m+1}L
\end{array} \tag{34}$$

The top square commutes by the induction hypothesis. The middle horizontal morphism is $\Omega(f,\eta)_{n+m} = \eta^{(n+m)} * f = G^{n+m}f \circ \eta_K^{(n+m)}$. Horizontal composition is associative, and the bottom morphism is $\Omega(f,\eta)_{n+m+1} = \eta^{(n+m+1)} * f = \eta * \eta^{(n+m)} * f = \eta^{(n+m)} * \eta * f$. The bottom square of diagram 34 is equal to the outer rectangle of the following diagram:

$$\begin{array}{ccccc}
F^{n+m}K & \xrightarrow{\eta_K^{(n+m)}} & G^{n+m}K & \xrightarrow{G^{n+m}f} & G^{n+m}L \\
{\scriptstyle F^{n+m}k}\downarrow & & \downarrow{\scriptstyle G^{n+m}k} & & \downarrow{\scriptstyle G^{n+m}l} \\
F^{n+m+1}K & \xrightarrow{\eta_{FK}^{(n+m)}} & G^{n+m}FK & \xrightarrow{G^{n+m}(\eta * f)} & G^{n+m+1}L
\end{array} \tag{35}$$

The left square of diagram 35 commutes by naturality of $\eta^{(n+m)}$. The right square commutes because $l \circ f = (\eta * f) \circ k$, which holds because (f, η) is a morphism. So diagram 35 commutes. We conclude that diagram 34 commutes.

Next, we must show that Ω respects composition. Let $(f, \eta) : (K, k, F) \to (L, l, G)$ and $(g, \rho) : (L, l, G) \to (M, m, H)$ be arbitrary morphisms. We must show that $\Omega(g \circ f, \rho \circ \eta) = \Omega(g, \rho) \circ \Omega(f, \eta)$. This entails showing for all $n \in \mathbb{N}$ that $\Omega(g \circ f, \rho \circ \eta)_n = \Omega(g, \rho)_n \circ \Omega(f, \eta)_n$. We proceed by induction on n. When $n = 0$, $\Omega(g \circ f, \rho \circ \eta)_0 = g \circ f = \Omega(g, \rho)_0 \circ \Omega(f, \eta)_0$. Assume the result for some n, and consider the case $n + 1$. Then $\Omega(g \circ f, \rho \circ \eta)_{n+1} = (\rho \circ \eta)^{n+1} * (g \circ f)$ is the diagonal of the following commuting square:

$$\begin{array}{ccc} F^{n+1}K & \xrightarrow{(\rho\circ\eta)_K^{(n+1)}} & H^{n+1}K \\ {\scriptstyle F^{n+1}(g\circ f)}\downarrow & \overset{\Omega(g\circ f,\rho\circ\eta)_{n+1}}{\searrow} & \downarrow{\scriptstyle H^{n+1}(g\circ f)} \\ F^{n+1}M & \xrightarrow[(\rho\circ\eta)_M^{(n+1)}]{} & H^{n+1}M. \end{array}$$

By lemma 4.2.5, $(\rho \circ \eta)^{(n+1)} = \rho^{(n+1)} \circ \eta^{(n+1)}$. We recognize the above diagram as the perimeter and diagonal of the following commuting diagram:

$$\begin{array}{ccccc} F^{n+1}K & \xrightarrow{\eta_K^{(n+1)}} & G^{n+1}K & \xrightarrow{\rho_K^{(n+1)}} & H^{n+1}K \\ {\scriptstyle F^{n+1}f}\downarrow & \overset{\Omega(f,\eta)_{n+1}}{\searrow} & {\scriptstyle G^{n+1}f}\downarrow & \overset{\Omega(f,\rho)_{n+1}}{\searrow} & \downarrow{\scriptstyle H^{n+1}f} \\ F^{n+1}L & \xrightarrow{\eta_L^{(n+1)}} & G^{n+1}L & \xrightarrow{\rho_L^{(n+1)}} & H^{n+1}L \\ {\scriptstyle F^{n+1}g}\downarrow & \overset{\Omega(g,\eta)_{n+1}}{\searrow} & {\scriptstyle G^{n+1}g}\downarrow & \overset{\Omega(g,\rho)_{n+1}}{\searrow} & \downarrow{\scriptstyle H^{n+1}f} \\ F^{n+1}M & \xrightarrow[\eta_M^{(n+1)}]{} & G^{n+1}M & \xrightarrow[\rho_M^{(n+1)}]{} & H^{n+1}M \end{array}$$

That is, we have $\Omega(g \circ f, \rho \circ \eta)_{n+1} = (\Omega(g, \rho) \circ \Omega(f, \eta))_{n+1}$. We conclude by induction that $\Omega(g \circ f, \rho \circ \eta) = \Omega(g, \rho) \circ \Omega(f, \eta)$. □

Proposition 4.2.7. *Equations* (29) *to* (32) *define an* ω*-functor* $\Omega : \mathbf{Links_K} \to \omega\text{-}\mathbf{Cat}[\omega \to \mathbf{K}]$.

Proof. The proof loosely follows [LS81, Theorem 4.1]. We know by lemma 4.2.6 that eqs. (29) to (32) define a functor $\Omega : \mathbf{Links_K} \to \omega\text{-}\mathbf{Cat}[\omega \to \mathbf{K}]$. We show that it preserves ω-colimits. Let $J : \omega \to \mathbf{Links_K}$ be arbitrary. Let $Jm = (K_m, k_m, F_m)$ and $J(m \to m+1) = (j_{m0}, \eta_m) : (K_m, k_m, F_m) \to (K_{m+1}, k_{(m+1)}, F_{m+1})$; the purpose of the zeros in the subscript will be made clear shortly. We can visualize ΩJ as a commuting grid in $\mathbf{K}$, with the m-axis pointing down and the n-axis pointing to the right. Indeed, by uncurrying $\Omega J : \omega \to \mathbf{Cat}[\omega \to \mathbf{Links_K}]$, we get a functor $G : \omega \times \omega \to \mathbf{Cat}[\omega \to \mathbf{Links_K}]$ where $G(m, n) = \Omega(Jm)n = F_m^n K_m$. The m-th row is given by the functor $H_m = G(m, -) : \omega \to \mathbf{Links_K}$, while the n-th column is given by the functor $V_n = G(-, n) : \omega \to \mathbf{Links_K}$. We use the following abbreviations in the grid:

$$\begin{aligned} j_m &= \Omega(J(m \to m+1)) : H_m \Rightarrow H_{m+1}, \\ j_{mn} &= G(m \to m+1, n) : G(m, n) \to G(m+1, n) \\ &= \eta_m^{(n)} * j_{m0} : F_m^n K_m \to F_{m+1}^n K_{m+1}, \\ j^{mn} &= G(m, n \to n+1) : G(m, n) \to G(m, n+1) \\ &= F_m^n k_m : F_m^n K_m \to F_m^{n+1} K_m, \\ j^n &= \{j^{mn} : G(m, n) \to G(m, n+1)\}_{m \in \mathbb{N}} : Vn \Rightarrow V(n+1). \end{aligned}$$

Then ΩJ and G determine the grid:

$$\begin{array}{ccccccc}
 & & V_0 & \xrightarrow{j^0} & V_1 & \xrightarrow{j^1} & V_2 \xrightarrow{j^2} \cdots \\
H_0: & & F_0^0 K_0 & \xrightarrow{j^{00}} & F_0^1 K_0 & \xrightarrow{j^{01}} & F_0^2 K_0 \xrightarrow{j^{02}} \cdots \\
\Downarrow j_0 & & \downarrow j_{00} & & \downarrow j_{01} & & \downarrow j_{02} \\
H_1: & & F_1^0 K_1 & \xrightarrow{j^{10}} & F_1^1 K_1 & \xrightarrow{j^{11}} & F_1^2 K_1 \xrightarrow{j^{12}} \cdots \\
\Downarrow j_1 & & \downarrow j_{10} & & \downarrow j_{11} & & \downarrow j_{12} \\
H_2: & & F_2^0 K_2 & \xrightarrow{j^{20}} & F_2^1 K_2 & \xrightarrow{j^{21}} & F_2^2 K_2 \xrightarrow{j^{22}} \cdots \\
\Downarrow j_2 & & \downarrow j_{20} & & \downarrow j_{21} & & \downarrow j_{22} \\
\vdots & & \vdots & & \vdots & & \vdots
\end{array}$$

Assume that J has a colimit $(\kappa, \phi) : J \Rightarrow (K_\infty, k_\infty, F_\infty)$. We must show that the cocone $(\Omega(\kappa,\phi), \Omega(K_\infty, k_\infty, F_\infty))$ is a colimit of $\Omega J : \omega \to \omega\text{-}\mathbf{Cat}[\omega \to \mathbf{K}]$. By proposition 4.2.4, $\kappa : V_0 \Rightarrow K_\infty$ and $\phi : \Phi \Rightarrow F_\infty$ are colimiting, where $\Phi = F_0 \overset{\eta_0}{\Longrightarrow} F_1 \overset{\eta_1}{\Longrightarrow} \cdots$. We can see each V_n as a diagram $\hat{V}_n$

$$F_0^n \circ (\Delta K_0) \overset{j_{01}}{\Longrightarrow} F_1^n \circ (\Delta K_1) \overset{j_{11}}{\Longrightarrow} \cdots$$

in ω-$\mathbf{Cat}[\mathbf{K} \to \mathbf{K}]$. By lemma 4.2.3, its colimit is $\Phi^{(n)} * \Delta\kappa : \hat{V}_n \Rightarrow F_\infty^n \circ (\Delta K_\infty)$. It follows that the cocone $\phi^{(n)} * \kappa : V_n \Rightarrow F_\infty^n K_\infty$ is colimiting in $\mathbf{K}$. Recall that colimits in ω-$\mathbf{Cat}[\omega \to \mathbf{K}]$ are computed component-wise, and observe that $\Omega(\kappa,\phi)_n = \phi^{(n)} * \kappa$ and $F_\infty^n K_\infty = \Omega(K_\infty, k_\infty, F_\infty)(n)$.

To show that $\Omega(\kappa,\phi) : \Omega J \Rightarrow \Omega(K_\infty, k_\infty, F_\infty)$ is indeed colimiting, it remains to show that for each n, $F_\infty^n k_\infty$ is a mediating morphism of cocones, i.e., that the following diagram commutes for all n:

$$\begin{array}{ccc}
V_n & \xrightarrow{j^n} & V_{n+1} \\
{\scriptstyle \phi^{(n)} * \kappa}\Downarrow & & \Downarrow{\scriptstyle \phi^{(n+1)} * \kappa} \\
F_\infty^n K_\infty & \xrightarrow{F_\infty^n k_\infty} & F_\infty^{n+1} K_\infty .
\end{array} \tag{36}$$

Observe that the following diagram commutes for all m:

$$\begin{array}{ccccc}
F_m^n K_m & \xrightarrow{\left(\phi_m^{(n)}\right)_{K_m}} & F_\infty^n K_m & \xrightarrow{F_\infty \kappa_m} & F_\infty^n K_\infty \\
{\scriptstyle F_m^n k_m}\downarrow & & {\scriptstyle F_\infty^n k_m}\downarrow & & \downarrow{\scriptstyle F_\infty^n k_\infty} \\
F_m^{n+1} K_m & \xrightarrow{\left(\phi_m^{(n)}\right)_{F_m K_m}} & F_\infty^n F_m K_m & \xrightarrow{F_\infty^n (\phi_m * \kappa_m)} & F_\infty^{n+1} K_\infty
\end{array} \tag{37}$$

Indeed, the left square commutes by naturality of $\phi_m^{(n)} : F_m^n \Rightarrow F_\infty^n$. To see that the right square commutes, recall that by definition of colimit in $\mathbf{Links}_\mathbf{K}$, the following square commutes:

$$\begin{array}{ccc}
K_m & \xrightarrow{\kappa_m} & K_\infty \\
{\scriptstyle k_m}\downarrow & & \downarrow{\scriptstyle k_\infty} \\
F_m K_m & \xrightarrow{\eta_m * \kappa_m} & F_\infty K_\infty .
\end{array}$$

Functors (including F_∞) preserve commuting diagrams, so the right square commutes. We recognize the perimeter of diagram 37 as:

$$\begin{array}{ccc} V_n m & \xrightarrow{\left(\phi^{(n)} * \kappa\right)_m} & F_\infty^n K_\infty \\ {\scriptstyle j^{mn}}\downarrow & & \downarrow{\scriptstyle F_\infty^n k_\infty} \\ V_{n+1} m & \xrightarrow{\left(\phi^{(n+1)} * \kappa\right)_m} & F_\infty^{n+1} K_\infty . \end{array}$$

Because this square commutes for all m, we conclude that diagram 36 commutes. So $F_\infty^n k_\infty$ is a mediating morphism of cocones.

It follows that $\Omega(\kappa, \phi) : \Omega J \Rightarrow \Omega(K_\infty, k_\infty, F_\infty)$ is colimiting, i.e., that Ω is an ω-functor. □

4.2.1. General Fixed Points. We define a generalized-fixed-point ω-functor, i.e., an ω-functor $\mathsf{GFIX} : \mathbf{Links}_{\mathbf{K}} \to \mathbf{K}$ such that for each link (K, k, F), there is an isomorphism $\mathsf{GFIX}(K, k, F) \cong F(\mathsf{GFIX}(K, k, F))$. We assume that whenever $\mathbf{K}$ is an ω-category, an ω-colimit has been chosen for each ω-chain. This choice determines an ω-colimit functor $\mathrm{colim}_\omega : \omega\text{-}\mathbf{Cat}\left[\omega \to \mathbf{K}\right] \to \mathbf{K}$, itself an ω-functor. The following result generalizes [LS81, Theorem 4.1]:

Proposition 4.2.8. *Let* $\mathbf{K}$ *be an* ω*-category. The following composition defines an* ω*-functor:*

$$\mathsf{GFIX} = \mathrm{colim}_\omega \circ \Omega : \mathbf{Links}_{\mathbf{K}} \to \mathbf{K}.$$

Proof. The functor Ω is ω-cocontinuous by proposition 4.2.7. The colimit functor is a left adjoint [Rie16, Proposition 4.5.1], and left adjoints preserve colimits [Rie16, Theorem 4.5.3]. The result then immediately follows from the fact that ω-functors are closed under composition. □

We claim that the isomorphism $\mathsf{GFIX}(K, k, F) \cong F(\mathsf{GFIX}(K, k, F))$ is natural in the link (K, k, F). To show this, we begin by defining an "unfolding" functor:

Proposition 4.2.9. *Let* $\mathbf{K}$ *be an* ω*-category. The following defines an* ω*-functor* $\mathsf{UNF} : \mathbf{Links}_{\mathbf{K}} \to \mathbf{K}$*:*

- *On objects:* $\mathsf{UNF}(K, k, F) = F(\mathsf{GFIX}(K, k, F))$,
- *on morphisms:* $\mathsf{UNF}(f, \eta) = \eta * \mathsf{GFIX}(f, \eta)$.

Proof. It clearly defines a functor. We show that it is ω-cocontinuous. Let $J = (K_0, k_0, F_0) \xrightarrow{(f_0, \eta_0)} (K_1, k_1, F_1) \xrightarrow{(f_1, \eta_1)} \cdots$ be an arbitrary ω-chain in $\mathbf{Links}_{\mathbf{K}}$. Let $((\kappa, \phi), (K_\infty, k_\infty, F_\infty))$ be colimiting for J. We must show that $\mathsf{UNF}(\kappa, \phi) : \mathsf{UNF} J \Rightarrow \mathsf{UNF}(K_\infty, k_\infty, F_\infty)$ is colimiting in $\mathbf{K}$, i.e., that the following diagram is colimiting:

$$\begin{array}{ccccccc} & {\scriptstyle \eta_0 * \mathsf{GFIX}(f_0, \eta_0)} & & {\scriptstyle \eta_1 * \mathsf{GFIX}(f_1, \eta_1)} & & {\scriptstyle \eta_2 * \mathsf{GFIX}(f_2, \eta_2)} & \\ F_0(\mathsf{GFIX}(K_0, k_0, F_0)) & & F_1(\mathsf{GFIX}(K_1, k_1, F_1)) & & F_2(\mathsf{GFIX}(K_2, k_2, F_2)) & & \cdots \\ & & \downarrow{\scriptstyle \phi_1 * \mathsf{GFIX}(\kappa_1, \phi_1)} & & & & \\ & {\scriptstyle \phi_0 * \mathsf{GFIX}(\kappa_0, \phi_0)} & F_\infty(\mathsf{GFIX}(K_\infty, k_\infty, F_\infty)) & & {\scriptstyle \phi_2 * \mathsf{GFIX}(\kappa_2, \phi_2)} & & \end{array} \tag{38}$$

We can recognize the above cocone as the image of an ω-colimit under an ω-functor. Define the ω-chain $W : \omega \to \omega\text{-}\mathbf{Cat}\left[\mathbf{K} \to \mathbf{K}\right] \times \omega\text{-}\mathbf{Cat}\left[\mathbf{1} \to \mathbf{K}\right]$ by:

$$\begin{aligned} Wn &= (F_n, \Delta(\mathsf{GFIX}(K_n, k_n, F_n))), \\ W(n \to n+1) &= (\eta_n, \Delta(\mathsf{GFIX}(f_n, \eta_n))), \end{aligned}$$

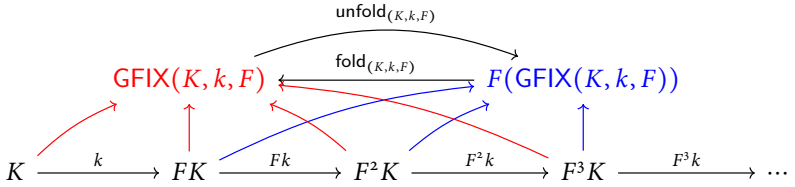
where $\Delta : \mathbf{K} \to \omega\text{-}\mathbf{Cat}\left[\mathbf{1} \to \mathbf{K}\right]$ is the constant-diagram functor. Colimits in product categories are computed component-wise. By proposition 4.2.4, (ϕ, F_∞) is colimiting in the first component. Because GFIX is an ω-functor, and $(\Delta(\mathsf{GFIX}(\kappa, \phi)), \Delta(\mathsf{GFIX}(K_\infty, k_\infty, F_\infty)))$ is colimiting in the second component. So $((\phi, \Delta(\mathsf{GFIX}(\kappa, \phi))), (F_\infty, \Delta(\mathsf{GFIX}(K_\infty, k_\infty, F_\infty))))$ is colimiting for W. The composition functor $\circ : \omega\text{-}\mathbf{Cat}\left[\mathbf{K} \to \mathbf{K}\right] \times \omega\text{-}\mathbf{Cat}\left[\mathbf{1} \to \mathbf{K}\right] \to \omega\text{-}\mathbf{Cat}\left[\mathbf{1} \to \mathbf{K}\right]$ is ω-cocontinuous

by proposition 4.A.14. The evaluation functor $\omega\text{-}\mathbf{Cat}\,[\mathbf{1} \to \mathbf{K}] \times \mathbf{1} \to \mathbf{K}$ is ω-cocontinuous by lemma 4.A.8, so $\mathsf{ev}_\bullet : \omega\text{-}\mathbf{Cat}\,[\mathbf{1} \to \mathbf{K}] \to \mathbf{K}$ is ω-cocontinuous. The image of W and its colimiting cocone under the composition $\mathsf{ev}_\bullet \circ (\circ)$ is exactly diagram 38. Because the composition is ω-cocontinuous, we conclude that diagram 38 is colimiting. □

We now construct a natural isomorphism $\mathsf{GFIX} \cong \mathsf{UNF}$. Each of its components is a mediating morphism of cocones induced by "shifting" the ω-chain the link generates. When $J : \omega \to \mathbf{K}$, write $\blacktriangleright J$ for the ω-chain induced by shifting J by one, i.e., by taking $\blacktriangleright J(n) = J(n+1)$. Observe that inclusion determines a natural transformation $\triangleright_J : J \Rightarrow \blacktriangleright J$, and every cocone (γ, G) on $\blacktriangleright J$ induces a cocone $(\gamma \circ \triangleright_J, G)$ on J. Also observe that $\blacktriangleright\Omega(K,k,F) = F\Omega(K,k,F)$. Building on these observations, we get the desired natural isomorphism:

Proposition 4.2.10. *Let* $\mathbf{K}$ *be an* ω*-category. There exists a natural isomorphism* $\mathsf{unfold} : \mathsf{GFIX} \Rightarrow \mathsf{UNF}$ *with inverse* $\mathsf{fold} : \mathsf{UNF} \Rightarrow \mathsf{GFIX}$. *Where* $\kappa : \Omega(K,k,F) \Rightarrow \mathsf{GFIX}(K,k,F))$ *is colimiting, the* (K,k,F)*-component of* unfold *is the unique morphism* $(\kappa, \mathsf{GFIX}(K,k,F)) \to (F\kappa \circ \triangleright_{\Omega(K,k,F)}, F(\mathsf{GFIX}(K,k,F)))$ *in* $\int \mathsf{Cone}(\Omega(K,k,F), -)$.

Proof. We begin by showing that the components are all isomorphisms. Fix some arbitrary link (K,k,F). Then $\Omega(K,k,F)$ generates the ω-chain at the bottom of the following diagram:



The colimit cocone $(\kappa, \mathsf{GFIX}(K,k,F))$ is in red on the left. Because F is an ω-functor, the blue cocone $(F\kappa, F(\mathsf{GFIX}(K,k,F)))$ on the right is again colimiting. It defines a cocone $(F\kappa \circ \triangleright_{\Omega(K,k,F)}, F(\mathsf{GFIX}(K,k,F)))$ on $\Omega(K,k,F)$ through composition. Let κ^+ be the restriction of κ to $F\Omega(K,k,F)$, i.e., $\kappa^+_n = \kappa_{n+1}$.

Consider the following mediating cocone morphisms

$$\mathsf{unfold}_{(K,k,F)} : (\kappa, \mathsf{GFIX}(K,k,F)) \to (F\kappa \circ \triangleright_{\Omega(K,k,F)}, F(\mathsf{GFIX}(K,k,F)))$$
$$\mathsf{fold}_{(K,k,F)} : (F\kappa, F(\mathsf{GFIX}(K,k,F))) \to (\kappa^+, \mathsf{GFIX}(K,k,F))$$

in $\int \mathsf{Cone}(\Omega(K,k,F), -)$ and $\int \mathsf{Cone}(F\Omega(K,k,F), -)$, respectively. We show that these two morphisms are mutual inverses in $\mathbf{K}$.

We begin by showing that $\mathsf{fold}_{(K,k,F)} \circ \mathsf{unfold}_{(K,k,F)} = \mathsf{id}_{\mathsf{GFIX}(K,k,F)}$. To do so, we show that $\mathsf{fold}_{(K,k,F)} \circ \mathsf{unfold}_{(K,k,F)}$ is a cocone morphism $(\kappa, \mathsf{GFIX}(K,k,F)) \to (\kappa, \mathsf{GFIX}(K,k,F))$. Because $(\kappa, \mathsf{GFIX}(K,k,F))$ is initial in $\int \mathsf{Cone}(\Omega(K,k,F), -)$, it will immediately follow that the morphism must be equal to the identity morphism. We must show for all n that $\kappa_n = (\mathsf{fold}_{(K,k,F)} \circ \mathsf{unfold}_{(K,k,F)}) \circ \kappa_n$. We compute:

$$\begin{aligned}
&\left(\mathsf{fold}_{(K,k,F)} \circ \mathsf{unfold}_{(K,k,F)}\right) \circ \kappa_n \\
&= \mathsf{fold}_{(K,k,F)} \circ \left(\mathsf{unfold}_{(K,k,F)} \circ \kappa_n\right) \\
&= \mathsf{fold}_{(K,k,F)} \circ \left(F\kappa \circ \triangleright_{\Omega(K,k,F)}\right)_n \\
&= \mathsf{fold}_{(K,k,F)} \circ (F\kappa)_n \circ \Omega(K,k,F)(n \to n+1) \\
&= \kappa^+_n \circ \Omega(K,k,F)(n \to n+1) \\
&= \kappa_{n+1} \circ \Omega(K,k,F)(n \to n+1) \\
&= \kappa_n.
\end{aligned}$$

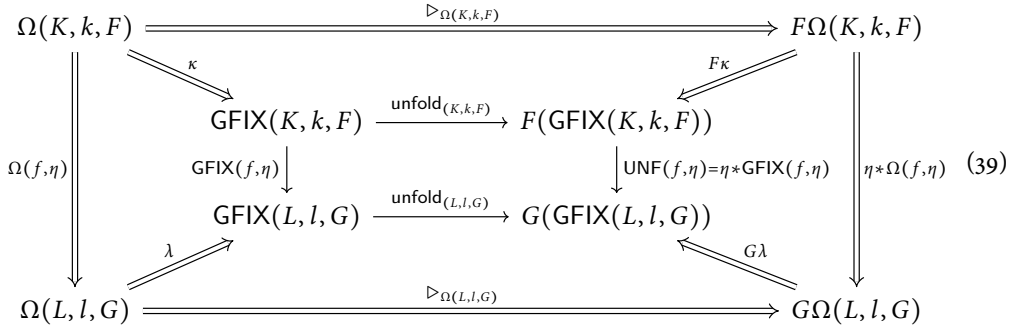
We conclude that $\mathsf{fold}_{(K,k,F)} \circ \mathsf{unfold}_{(K,k,F)} = \mathsf{id}_{\mathsf{GFIX}(K,k,F)}$.

Next, we show that $\mathsf{unfold}_{(K,k,F)} \circ \mathsf{fold}_{(K,k,F)} = \mathsf{id}_{F(\mathsf{GFIX}(K,k,F))}$ using an analogous argument. We compute:

$$\begin{aligned}
&(\mathsf{unfold}_{(K,k,F)} \circ \mathsf{fold}_{(K,k,F)}) \circ (F\kappa)_n \\
&= \mathsf{unfold}_{(K,k,F)} \circ \left(\mathsf{fold}_{(K,k,F)} \circ (F\kappa)_n\right) \\
&= \mathsf{unfold}_{(K,k,F)} \circ \kappa_n^+ \\
&= \mathsf{unfold}_{(K,k,F)} \circ \kappa_{n+1} \\
&= (F\kappa)^-_{n+1} \\
&= (F\kappa)_n.
\end{aligned}$$

We conclude that each of unfold is an isomorphism.

Next, we show that unfold is natural. Let $(f,\eta) : (K,k,F) \to (L,l,G)$ be an arbitrary morphism of $\mathbf{Links_K}$. Let $(\kappa, \mathsf{GFIX}(K,k,F))$ and $(\lambda, \mathsf{GFIX}(L,l,G))$ respectively be the colimiting cocones of $\Omega(K,k,F)$ and $\Omega(L,l,G)$. We show that the following diagram commutes in $\mathbf{K}$:



(39)

The top, left, and bottom trapezoids commute by definition of unfold and GFIX. To see that the right trapezoid commutes, observe that the following diagram commutes by definition of GFIX:

$$\begin{array}{ccc}
\mathsf{GFIX}(K,k,F) & \xleftarrow{\kappa} & \Omega(K,k,F) \\
{\scriptstyle \mathsf{GFIX}(f,\eta)}\downarrow & & \Downarrow{\scriptstyle \Omega(f,\eta)} \\
\mathsf{GFIX}(L,l,G) & \xleftarrow{\lambda} & \Omega(L,l,G)
\end{array}$$

commutes by definition of GFIX. Its image under F is the top rectangle in the following diagram; the bottom rectangle commutes by naturality of η:

$$\begin{array}{ccc}
F\mathsf{GFIX}(K,k,F) & \xleftarrow{F\kappa} & F\Omega(K,k,F) \\
{\scriptstyle F\mathsf{GFIX}(f,\eta)}\downarrow & & \Downarrow{\scriptstyle F\Omega(f,\eta)} \\
F\mathsf{GFIX}(L,l,G) & \xleftarrow{F\lambda} & F\Omega(L,l,G) \\
{\scriptstyle \eta_{F\mathsf{GFIX}(L,l,G)}}\downarrow & & \Downarrow{\scriptstyle \eta\Omega(L,l,G)} \\
G\mathsf{GFIX}(L,l,G) & \xleftarrow{G\lambda} & G\Omega(L,l,G)
\end{array}$$

The outer perimeter of this rectangle is exactly the right trapezoid of diagram 39. To see that the outer perimeter of diagram 39 commutes, we must show that it commutes at each component n. Observe that for arbitrary n,

$$\begin{array}{ccc}
\Omega(K,k,F)(n) = F^n k & \xrightarrow{(\triangleright_{\Omega(K,k,F)})_n = \Omega(K,k,F)(n \to n+1)} & F\Omega(K,k,F)(n) = FF^n k \\
{\scriptstyle \Omega(f,n)_n}\downarrow & & \downarrow{\scriptstyle (\eta * \Omega(f,n))_n = \Omega(f,n)_{n+1}} \\
\Omega(L,l,G)(n) = G^n L & \xrightarrow{(\triangleright_{\Omega(L,l,G)})_n = \Omega(L,l,G)(n \to n+1)} & G\Omega(L,l,G)(n) = GG^n L
\end{array}$$

is exactly the square given by the functoriality of Ω. So we conclude that the outer perimeter of diagram 39 commutes. It follows that the two paths around the outer perimeter define equal cocones:

$$\begin{aligned}&(G\lambda \circ (\eta * \Omega(f,\eta)) \circ \triangleright_{\Omega(K,k,F)}, G(\mathsf{GFIX}(L,l,G)))\\ &= (G\lambda \circ \triangleright_{\Omega(L,l,G)} \circ \Omega(f,n), G(\mathsf{GFIX}(L,l,G))).\end{aligned}$$

The inner rectangle of diagram 39 then describes two cocone morphisms

$$(\kappa, \mathsf{GFIX}(K,k,F)) \to (G\lambda \circ \triangleright_{\Omega(L,l,G)} \circ \Omega(f,n), G(\mathsf{GFIX}(L,l,G)))$$

in $\int \mathsf{Cone}(\Omega(K,k,F),-)$. Because $(\kappa, \mathsf{GFIX}(K,k,F))$ is initial, these two cocone morphisms must be equal, i.e., the inner rectangle commutes. We conclude that unfold is natural. □

We can specialize the above constructions to produce initial functor algebras. Indeed, given an **IFP**-category $\mathbf{K}$, the category $\mathbf{IFP}[\mathbf{K} \to \mathbf{K}]$ embeds fully and faithfully into $\mathbf{Links}_{\mathbf{K}}$ via the functor that maps objects $F : \mathbf{K} \to \mathbf{K}$ to the link $(\bot, \bot, F)$ and natural transformations $\eta : F \Rightarrow G$ to the morphism $(\mathsf{id}_\bot, \eta)$. We define the **initial-fixed-point functor** $\mathsf{FIX} : \mathbf{IFP}[\mathbf{K} \to \mathbf{K}] \to \mathbf{K}$ as the composition $\mathbf{IFP}[\mathbf{K} \to \mathbf{K}] \hookrightarrow \mathbf{Links}_{\mathbf{K}} \xrightarrow{\mathsf{GFIX}} \mathbf{K}$. The following proposition is standard:

Proposition 4.2.11 ([LS81, Theorem 4.1]). *The initial-fixed-point functor* $\mathsf{FIX} : \mathbf{IFP}[\mathbf{K} \to \mathbf{K}] \to \mathbf{K}$ *is an ω-functor.*

Proposition 4.2.12. *Let* $\mathbf{K}$ *be an ω-category. Let* $I : \mathbf{IFP}[\mathbf{K} \to \mathbf{K}] \to \mathbf{Links}_{\mathbf{K}}$ *be the functor given by* $I(F) = (\bot, \bot, F)$ *and* $I(\eta) = (\mathsf{id}_\bot, \eta)$*. Then I is a full and faithful ω-functor.*

Proof. The mapping I is clearly functorial. Let $\eta, \rho : F \Rightarrow G$ be two morphisms in $\mathbf{IFP}[\mathbf{K} \to \mathbf{K}]$ and assume $I(\eta) = I(\rho)$. Then $(\mathsf{id}_\bot, \eta) = (\mathsf{id}_\bot, \rho)$. It follows that $\eta = \rho$, so I is faithful.

Let $F, G : \mathbf{K} \to \mathbf{K}$ be two ω-functors. Let $(f, \eta) : I(F) \to I(G)$ be a morphism in $\mathbf{Links}_{\mathbf{K}}$. Then $(f, \eta) : (\bot, \bot, F) \to (\bot, \bot, G)$. This implies that $f : \bot \to \bot$. But $\bot$s is the initial object, and there exists a unique morphism $\bot \to \bot$, namely, $\mathsf{id}_\bot$. It follows that $(f, \eta) = I(\eta)$. We conclude that I is full.

Consider some diagram $J : \omega \to \mathbf{IFP}[\mathbf{K} \to \mathbf{K}]$ with colimit $\kappa : J \Rightarrow F_\infty$. Then $(I\kappa, IF_\infty)$ is colimiting for IJ by proposition 4.2.4. We conclude that I is ω-cocontinuous. □

We use the close relationship between cocones and functor algebras to show that FIX does indeed produce initial fixed points. Write $F^\omega : \omega \to \mathbf{K}$ for $\Omega(\bot, \bot, F)$. The following proposition tells us that every F-algebra induces a cocone on F^ω. This construction of cocones from algebras is not new: it appears in the proofs of [SP82, Lemma 2; AMM18, Theorem 3.5]. However, to the best of our knowledge, the fact that this action on objects extends to a full and faithful functor, and the initiality result are new. These facts will be used repeatedly in proofs below.

Proposition 4.2.13. *Let* $\mathbf{K}$ *be a category with an initial object, and let* $F : \mathbf{K} \to \mathbf{K}$ *be a functor. The following defines a full and faithful functor* $\mathsf{Cone}^F : \mathbf{K}^F \to \int \mathsf{Cone}(F^\omega, -)$ *from the category* $\mathbf{K}^F$ *of F-algebras to the category* $\int \mathsf{Cone}(F^\omega, -)$ *of cocones on* F^ω*:*

- *on objects:* $\mathsf{Cone}^F(A, a) = (\alpha, A)$ *where* $\alpha : F^\omega \Rightarrow A$ *is inductively defined by* $\alpha_0 = \bot_A$ *and* $\alpha_{n+1} = a \circ F\alpha_n$
- *on morphisms:* $\mathsf{Cone}^F f = f$.

If F is an ω-functor and $\mathbf{K}$ *is an* **IFP***-category, then* $\mathsf{Cone}^F\left(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)}\right)$ *is initial.*

Proof. We begin by checking that the functor is well-defined on objects. Let (A, a) be an F-algebra. We show that α is a cocone on $\Omega(\bot, \bot, F)$ with nadir A. We must show that for all n, $\alpha_n = \alpha_{n+1} \circ \Omega(\bot, \bot, F)(n \to n+1)$. We do so by induction on n. When $n = 0$, we have by initiality that

$$\alpha_0 = \bot_A = a \circ F\bot_A \circ \bot_{F\bot} = \alpha_1 \circ \Omega(\bot, \bot, F)(0 \to 1).$$

Assume the result for some n, then

$$\begin{aligned}\alpha_{n+1} &= a \circ F(\alpha_n) \\ &= a \circ F(\alpha_{n+1} \circ \Omega(\bot, \bot, F)(n \to n+1)) \\ &= a \circ F(\alpha_{n+1}) \circ F(\Omega(\bot, \bot, F)(n \to n+1)) \\ &= a \circ F(\alpha_{n+1}) \circ \Omega(\bot, \bot, F)(n+1 \to n+2) \\ &= \alpha_{n+2} \circ \Omega(\bot, \bot, F)(n+1 \to n+2).\end{aligned}$$

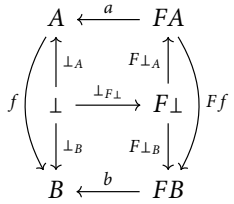
We conclude that α is a cocone.

The action of Cone^F on morphisms is clearly functorial. Let (A, a) and (B, b) be F-algebras and let (α, A) and (β, B) be their respective images under Cone^F. We must show that if $f : (A, a) \to (B, b)$ is an F-algebra homomorphism, then it is a morphism of cocones. In particular, we must show that for all $n \in \mathbb{N}$, $f \circ \alpha_n = \beta_n$. We do so by induction on n. When $n = 0$, we have by initiality that $f \circ \alpha_0 = \bot_B = \beta_0$. Assume the result for some n. Because f is an F-algebra homomorphism, $f \circ a = b \circ Ff$. It follows that:

$$f \circ \alpha_{n+1} = f \circ a \circ F\alpha_n = b \circ Ff \circ F\alpha_n = b \circ F(f \circ \alpha_n) = b \circ F\beta_n = \beta_{n+1}.$$

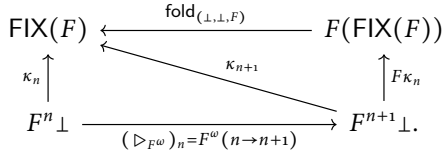
We conclude the result by induction.

The functor is clearly faithful. We show that it is full. Let (A, a) and (B, b) be arbitrary F-algebras, and let $f : \mathsf{Cone}^F(A, a) \to \mathsf{Cone}^F(B, b)$ be arbitrary. We claim that $f : (A, a) \to (B, b)$ is an F-algebra homomorphism. We must show that $f \circ a = b \circ Ff$. Consider the following diagram:



The top and bottom squares commute by definition of $\mathsf{Cone}^F(A, a)$ and $\mathsf{Cone}^F(B, b)$. The left circular segment commutes by initiality, while the right circular segment commutes because functors preserve commuting diagrams. So the whole diagram commutes. We conclude that $f : (A, a) \to (B, b)$ is an F-algebra homomorphism. It follows that Cone^F is full.

Assume F is an ω-functor and let $\kappa : F^\omega \Rightarrow \mathsf{FIX}(F)$ be colimiting. Then $(\kappa, \mathsf{FIX}(F))$ is initial in $\int \mathsf{Cone}(F^\omega, -)$. We show that $\mathsf{Cone}^F(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)}) = (\kappa, \mathsf{FIX}(F))$. Set $(\phi, \mathsf{FIX}(F)) = \mathsf{Cone}^F(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)})$. We show by induction that $\kappa_n = \phi_n$ for all n. When $n = 0$, the result is immediate by initiality: $\kappa_0 = \bot_{\mathsf{FIX}(F)} = \phi_0$. Now assume the result for some n. Recall that $\mathsf{fold}_{(\bot,\bot,F)}$ is by definition (proposition 4.2.10) the unique cocone morphism $(F\kappa \circ \rhd_{F^\omega}, F(\mathsf{FIX}(F))) \to (\kappa, \mathsf{FIX}(F))$. This implies that perimeter of the following diagram commutes for all n:



(40)

The bottom triangle commutes by definition of κ. We show that the top triangle commutes. Recall that F is an ω-functor, so $F\kappa : FF^\omega \Rightarrow F(\mathsf{FIX}(F))$ is colimiting. Observe that $(\kappa^+)_n = \kappa_{n+1} : FF^\omega \Rightarrow \mathsf{FIX}(F)$ is a cocone. It follows that there exists a unique morphism $(F\kappa, F(\mathsf{FIX}(F))) \to (\kappa^+, \mathsf{FIX}(F))$ in $\int \mathsf{Cone}(FF^\omega, -)$, i.e., a unique $f : F(\mathsf{FIX}(F)) \to \mathsf{FIX}(F)$ in $\mathbf{K}$ such that

$$f \circ F\kappa_n = \kappa_{n+1} : F^{n+1}\bot \to \mathsf{FIX}(F)$$

for all n. We claim that $f = \mathsf{fold}_{(\bot,\bot,F)}$. To see that this is so, observe that the following diagram commutes for all n:

$$\begin{array}{ccc} \mathsf{FIX}(F) & \xleftarrow{\quad f \quad} & F(\mathsf{FIX}(F)) \\ {\scriptstyle \kappa_n}\uparrow & \nwarrow^{\kappa_{n+1}} & \uparrow{\scriptstyle F\kappa_n} \\ F^n\bot & \xrightarrow[(\rhd_{F^\omega})_n = F^\omega(n \to n+1)]{} & F^{n+1}\bot. \end{array}$$

Indeed, the top triangle commutes by definition of f, while the bottom triangle commutes by definition of κ. The perimeter of this diagram implies that f is a cocone morphism $(F\kappa \circ \rhd_{F^\omega}, F(\mathsf{FIX}(F))) \to (\kappa, \mathsf{FIX}(F))$. But $\mathsf{fold}_{(\bot,\bot,F)}$ is the unique such morphism, so we have $f = \mathsf{fold}_{(\bot,\bot,F)}$. It follows that the top triangle of diagram 40 commutes, so the entire diagram commutes.

We are now ready to show that $\phi_{n+1} = \kappa_{n+1}$. Recall that $\phi_{n+1} = \mathsf{fold}_{(\bot,\bot,F)} \circ F\phi_n$ by definition. By the induction hypothesis, $\phi_n = \kappa_n$, so $\phi_{n+1} = \mathsf{fold}_{(\bot,\bot,F)} \circ F\kappa_n$. By diagram 40, it follows that $\phi_{n+1} = \kappa_{n+1}$. We conclude by induction that $\phi = \kappa$. It follows that $\mathsf{Cone}^F(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)}) = (\kappa, \mathsf{FIX}(F))$ is initial. □

The following corollary is again standard, but its proof is new and its statement clarifies the nature of the mediating F-algebra homomorphism:

Corollary 4.2.14 ([SP82, Lemma 2; AMM18, Theorem 3.5]). *Let* $\mathbf{K}$ *be an* **IFP***-category. The initial algebra of an* ω*-functor* $F : \mathbf{K} \to \mathbf{K}$ *is* $(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)})$*. Given any other* F*-algebra* (A, a)*, the unique* F*-algebra homomorphism* $(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)}) \to (A, a)$ *is the unique cocone morphism* $\mathsf{Cone}^F(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)}) \to \mathsf{Cone}^F(A, a)$.

Proof. The cocone $\mathsf{Cone}^F(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)})$ is initial in $\int \mathsf{Cone}(F^\omega, -)$ by proposition 4.2.13. Recall that initial objects are given by the limit of the identity functor [Rie16, Lemma 3.7.1], and that full and faithful functors reflect any limits that are present in its codomain [Rie16, Lemma 3.3.5]. Because Cone^F is full and faithful, it follows that $\left(\mathsf{FIX}(F), \mathsf{fold}_{(\bot,\bot,F)}\right)$ is initial and that the unique morphism is as described. □

4.3. 2-Categorical Structure of Parametrized Fixed Points

In this section, we explore the 2-categorical properties of the parametrized-fixed-point functor given by Lehmann and Smyth [LS77, § 3]. This 2-categorical structure is, to the best of our knowledge, new. From these properties, we deduce that the parametrized-fixed-point functor defines a dagger operation that satisfies the Conway identities.

We begin by observing that their parametrized-fixed-point functor is a 2-natural transformation. This answers the first question of the introduction: the definition of $(\cdot)^\dagger : \mathbf{IFP}\left[\mathbf{D} \times \mathbf{E} \to \mathbf{E}\right] \to \mathbf{IFP}\left[\mathbf{D} \to \mathbf{E}\right]$ is natural in $\mathbf{D}$. In fact, naturality does not require $\mathbf{D}$ to be an ω-category. Given a category $\mathbf{D}$ and an **IFP**-category $\mathbf{E}$, let $\mathbf{Cat}\left[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}\right]$ be the category of parametrized ω-functors $F : \mathbf{D} \times \mathbf{E} \to \mathbf{E}$, i.e., functors such that $F(D, -) : \mathbf{E} \to \mathbf{E}$ is an ω-functor for all objects D of $\mathbf{D}$.

Proposition 4.3.1. *Let* $\mathbf{E}$ *be an* ω*-category. The following family of functors forms a 2-natural transformation* $(\cdot)^\dagger : \mathbf{Cat}\left[- \times \mathbf{E} \to_\omega \mathbf{E}\right] \Rightarrow \mathbf{Cat}\left[- \to \mathbf{E}\right] : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{Cat}$*:*

$$(\cdot)^\dagger_{\mathbf{D}} = \mathbf{Cat}\left[\mathsf{id}_{\mathbf{D}} \to \mathsf{FIX}\right] \circ \Lambda : \mathbf{Cat}\left[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}\right] \to \mathbf{Cat}\left[\mathbf{D} \to \mathbf{E}\right].$$

It restricts to a 2-natural transformation $(\cdot)^\dagger : \mathbf{IFP}\left[- \times \mathbf{E} \to \mathbf{E}\right] \Rightarrow \mathbf{IFP}\left[- \to \mathbf{E}\right] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP}$.

Proof. We begin by showing naturality. We must show for all $G : \mathbf{C} \to \mathbf{D}$ that the following diagram commutes:

$$\begin{array}{ccc} \mathbf{Cat}\left[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}\right] & \xrightarrow{(\cdot)^\dagger_{\mathbf{D}}} & \mathbf{Cat}\left[\mathbf{D} \to \mathbf{E}\right] \\ {\scriptstyle \mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]}\downarrow & & \downarrow{\scriptstyle \mathbf{Cat}[G \to \mathbf{E}]} \\ \mathbf{Cat}\left[\mathbf{C} \times \mathbf{E} \to_\omega \mathbf{E}\right] & \xrightarrow{(\cdot)^\dagger_{\mathbf{C}}} & \mathbf{Cat}\left[\mathbf{C} \to \mathbf{E}\right] \end{array}$$

We first show that the two functors defined by the two paths around the square agree on objects. Let $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$ be arbitrary and abbreviate $F_G = \mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}](F) = F \circ (G \times \mathsf{id}_\mathbf{E})$. By naturality of Λ:

$$\Lambda F_G = \Lambda(F \circ (G \times \mathsf{id}_\mathbf{E})) = (\Lambda F) \circ G.$$

Then by definition of $(\cdot)^\dagger$,

$$\begin{aligned}
&(F_G)^\dagger_\mathbf{C} \\
&= ([\mathsf{id}_\mathbf{C} \to \mathsf{FIX}] \circ \Lambda)(F_G) \\
&= [\mathsf{id}_\mathbf{C} \to \mathsf{FIX}](\Lambda F \circ G) \\
&= \mathsf{FIX} \circ \Lambda F \circ G \circ \mathsf{id}_\mathbf{C} \\
&= \mathsf{FIX} \circ \Lambda F \circ \mathsf{id}_\mathbf{D} \circ G \\
&= [\mathsf{id}_\mathbf{D} \to \mathsf{FIX}](\Lambda F) \circ G \\
&= F^\dagger_\mathbf{D} \circ G.
\end{aligned}$$

Now let $\eta : F \Rightarrow F'$ be an arbitrary natural transformation. We compute:

$$\begin{aligned}
&\left((\cdot)^\dagger_\mathbf{C} \circ \mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]\right)\eta \\
&= (\eta * (G \times \mathsf{id}_\mathbf{E}))^\dagger_\mathbf{C} \\
&= [\mathsf{id}_\mathbf{C} \to \mathsf{FIX}]\,(\Lambda\,(\eta * (G \times \mathsf{id}_\mathbf{E}))) \\
&= [\mathsf{id}_\mathbf{C} \to \mathsf{FIX}]\,(\Lambda\eta * G) \\
&= \mathsf{FIX} * (\Lambda\eta * G) * \mathsf{id}_\mathbf{C} \\
&= \mathsf{FIX} * \Lambda\eta * \mathsf{id}_\mathbf{D} * G \\
&= (\Lambda\eta)^\dagger_\mathbf{D} * G.
\end{aligned}$$

We conclude that the two paths around the diagram define equal functors, i.e., that $(\cdot)^\dagger$ is natural.

We show that it is 2-natural. Let $\alpha : G \Rightarrow G' : \mathbf{C} \to \mathbf{D}$ be an arbitrary 2-cell in **Cat**. We must show that the two following 2-cells (i.e., natural transformations) are equal in **Cat**:

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \;\underset{\mathbf{Cat}[G' \times \mathbf{E} \to_\omega \mathbf{E}]}{\overset{\mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]}{\Downarrow \mathbf{Cat}[\alpha \times \mathbf{E} \to_\omega \mathbf{E}]}}\; \mathbf{Cat}[\mathbf{C} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow{(\cdot)^\dagger_\mathbf{C}} \mathbf{Cat}\,[\mathbf{C} \to \mathbf{E}]\,,$$

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow{(\cdot)^\dagger_\mathbf{D}} \mathbf{Cat}\,[\mathbf{D} \to \mathbf{E}] \;\underset{\mathbf{Cat}[G' \to \mathbf{E}]}{\overset{\mathbf{Cat}[G \to \mathbf{E}]}{\Downarrow \mathbf{Cat}\,[\alpha \to \mathbf{E}]}}\; \mathbf{Cat}\,[\mathbf{C} \to \mathbf{E}]\,.$$

For an arbitrary component $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$, we compute:

$$\begin{aligned}
&\left((\cdot)^\dagger_\mathbf{C} * \mathbf{Cat}[\alpha \times \mathbf{E} \to_\omega \mathbf{E}]\right)_F \\
&= (F * (\alpha \times \mathsf{id}_\mathbf{E}) : F \circ (G \times \mathsf{id}_\mathbf{E}) \Rightarrow F \circ (G' \times \mathsf{id}_\mathbf{E}))^\dagger_\mathbf{C} \\
&= ([\mathsf{id}_\mathbf{C} \to \mathsf{FIX}] \circ \Lambda)\,(F * (\alpha \times \mathsf{id}_\mathbf{E}) : F \circ (G \times \mathsf{id}_\mathbf{E}) \Rightarrow F \circ (G' \times \mathsf{id}_\mathbf{E})) \\
&= [\mathsf{id}_\mathbf{C} \to \mathsf{FIX}]\,(\Lambda F * \alpha : FG \Rightarrow FG') \\
&= \mathsf{FIX} * \Lambda F * \alpha \\
&= \mathsf{FIX} * \Lambda F * \mathsf{id}_\mathbf{D} * \alpha \\
&= F^\dagger_\mathbf{D} * \alpha \\
&= \left(\mathbf{Cat}\,[\alpha \to \mathbf{E}] * (\cdot)^\dagger_\mathbf{D}\right)_F.
\end{aligned}$$

We conclude 2-naturality.

We show that $(\cdot)^\dagger$ restricts to a 2-natural transformation $\mathbf{IFP}\left[- \times \mathbf{E} \to \mathbf{E}\right] \Rightarrow \mathbf{IFP}\left[- \to \mathbf{E}\right] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP}$. To do so, we note that each $\mathbf{D}$-component restricts to an ω-functor

$$\mathbf{IFP}\left[\mathsf{id}_{\mathbf{D}} \to \mathsf{FIX}\right] \circ \Lambda : \mathbf{IFP}\left[- \times \mathbf{E} \to \mathbf{E}\right] \Rightarrow \mathbf{IFP}\left[- \to \mathbf{E}\right] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP}.$$

by lemma 4.A.8 and propositions 4.A.10 and 4.2.11. The same argument as above shows that it is 2-natural. □

Explicitly, given an $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$ and an object D of $\mathbf{D}$, $F^\dagger_{\mathbf{D}} D = \mathsf{FIX}(F(D, -))$. Proposition 4.3.1 implies that $(\cdot)^\dagger$ defines an external dagger operation on horizontal morphisms in **IFP**:

Definition 4.3.2. Let $\mathbf{C}$ be a cartesian category. An **external dagger operation** in product form [BÉ96, § 3.1; BÉ95, Definition 2.6] is a family of set-theoretic functions $\dagger = (\dagger_{A,B})$ indexed by pairs of objects A, B in $\mathbf{C}$, where $\dagger_{A,B} : \mathbf{C}(A \times B, A) \to \mathbf{C}(B, A)$ is a function of hom-sets. Given an external dagger operation $\dagger_{A,B} : \mathbf{C}(A \times B, B) \to \mathbf{C}(A, B)$ and a morphism $f : A \times B \to B$, we write $f^\dagger$ for $\dagger_{A,B}(f)$. ◂

We will show that $(\cdot)^\dagger$ produces parametrized fixed points. To do so, we begin by defining a family of functors that gives their unrollings:

Proposition 4.3.3. *Let* $\mathbf{E}$ *be an* ω*-category. The following family of functors forms a 2-natural transformation* $\mathsf{UNR}(\cdot) : \mathbf{Cat}[- \times \mathbf{E} \to_\omega \mathbf{E}] \Rightarrow \mathbf{Cat}\left[- \to \mathbf{E}\right] : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{Cat}$*, where* $\mathbf{D}$ *ranges over small categories:*

$$\mathsf{UNR}_{\mathbf{D}}(F) = F \circ \langle \mathsf{id}_{\mathbf{D}}, F^\dagger \rangle$$
$$\mathsf{UNR}_{\mathbf{D}}(\eta) = \eta * \langle \mathsf{id}_{\mathbf{D}}, \eta^\dagger \rangle$$

It restricts to a 2-natural transformation $\mathsf{UNR}(\cdot) : \mathbf{IFP}\left[- \times \mathbf{E} \to \mathbf{E}\right] \Rightarrow \mathbf{IFP}\left[- \to \mathbf{E}\right] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP}$.

Proof. Each component is a well-defined functor. We first show that the components assemble into a natural transformation. Let $G : \mathbf{C} \to \mathbf{D}$ be arbitrary. We must show that the following diagram commutes:

$$\begin{array}{ccc} \mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] & \xrightarrow{\mathsf{UNR}_{\mathbf{D}}} & \mathbf{Cat}\left[\mathbf{D} \to \mathbf{E}\right] \\ {\scriptstyle \mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]}\downarrow & & \downarrow{\scriptstyle \mathbf{Cat}[G \to \mathbf{E}]} \\ \mathbf{Cat}[\mathbf{C} \times \mathbf{E} \to_\omega \mathbf{E}] & \xrightarrow{\mathsf{UNR}_{\mathbf{C}}} & \mathbf{Cat}\left[\mathbf{C} \to \mathbf{E}\right]. \end{array}$$

We show that the two paths around the diagram agree on objects. Let $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$ be arbitrary, then

$$\begin{aligned} &\left(\mathbf{Cat}\left[G \to \mathbf{E}\right] \circ \mathsf{UNR}_{\mathbf{D}}\right)(F) \\ &= \mathsf{UNR}_{\mathbf{D}}(F) \circ G \\ &= F \circ \langle \mathsf{id}_{\mathbf{D}}, F^\dagger \rangle \circ G \\ &= F \circ \langle \mathsf{id}_{\mathbf{D}} \circ G, F^\dagger \circ G \rangle \\ &= F \circ \langle G \circ \mathsf{id}_{\mathbf{C}}, \mathsf{id}_{\mathbf{E}} \circ F^\dagger \circ G \rangle \\ &= (F \circ (G \times \mathsf{id}_{\mathbf{E}})) \circ \langle \mathsf{id}_{\mathbf{C}}, F^\dagger \circ G \rangle \end{aligned}$$

which by proposition 4.3.1,

$$\begin{aligned} &= (F \circ (G \times \mathsf{id}_{\mathbf{E}})) \circ \langle \mathsf{id}_{\mathbf{C}}, (F \circ (G \times \mathsf{id}_{\mathbf{E}}))^\dagger \rangle \\ &= \mathsf{UNR}_{\mathbf{C}}(F \circ (G \times \mathsf{id}_{\mathbf{E}})) \\ &= \left(\mathsf{UNR}_{\mathbf{C}} \circ \mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]\right)(F). \end{aligned}$$

An almost identical derivation gives that the two paths around the diagram agree on morphisms. We conclude naturality.

We now show 2-naturality. Let $\alpha : G \Rightarrow G' : \mathbf{C} \to \mathbf{D}$ be an arbitrary 2-cell in **Cat**. We must show that the following 2-cells (i.e., natural transformations) are equal in **Cat**:

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow[\mathbf{Cat}[G' \times \mathbf{E} \to_\omega \mathbf{E}]]{\mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]} \Downarrow \mathbf{Cat}[\alpha \times \mathbf{E} \to_\omega \mathbf{E}] \quad \mathbf{Cat}[\mathbf{C} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow{\mathsf{UNR}(\cdot)_{\mathbf{C}}} \mathbf{Cat}[\mathbf{C} \to \mathbf{E}],$$

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow{\mathsf{UNR}(\cdot)_{\mathbf{D}}} \mathbf{Cat}[\mathbf{D} \to \mathbf{E}] \xrightarrow[\mathbf{Cat}[G' \to \mathbf{E}]]{\mathbf{Cat}[G \to \mathbf{E}]} \Downarrow \mathbf{Cat}[\alpha \to \mathbf{E}] \quad \mathbf{Cat}[\mathbf{C} \to \mathbf{E}].$$

For an arbitrary component $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$, we compute:

$$\begin{aligned}
&(\mathsf{UNR}(\cdot)_{\mathbf{C}} * \mathbf{Cat}[\alpha \times \mathbf{E} \to_\omega \mathbf{E}])_F \\
&= \mathsf{UNR}(F * (\alpha \times \mathsf{id}_{\mathbf{E}}))_{\mathbf{C}} \\
&= F * (\alpha \times \mathsf{id}_{\mathbf{E}}) * \langle \mathsf{id}_{\mathbf{C}}, (F * (\alpha \times \mathsf{id}_{\mathbf{E}}))^\dagger \rangle \\
&= F * \langle \alpha, F^\dagger * \alpha \rangle \\
&= F * \langle \mathsf{id}_{\mathbf{D}}, F^\dagger \rangle * \alpha \\
&= \mathsf{UNR}(F)_{\mathbf{D}} * \alpha \\
&= (\mathbf{Cat}[\alpha \to \mathbf{E}] \circ \mathsf{UNR}(\cdot)_{\mathbf{D}})_F .
\end{aligned}$$

We conclude 2-naturality.

To show that 2-natural transformation restricts to $\mathsf{UNR}(\cdot) : \mathbf{IFP}[- \times \mathbf{E} \to \mathbf{E}] \Rightarrow \mathbf{IFP}[- \to \mathbf{E}] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP}$, we show that each component is an ω-functor. But this is obvious by proposition 4.2.11. The proof of 2-naturality carries over unchanged. □

We usually expect a dagger operations to satisfy the fixed-point identity [BÉ96, p. 7]. It states that $f^\dagger = f \circ \langle \mathsf{id}_A, f^\dagger \rangle$ for all $f : A \times B \to B$, i.e., that a dagger operation gives parametrized fixed points. The fixed-point identity does not hold in general for dagger operations on functors: $F^\dagger$ and $F \circ \langle \mathsf{id}, F^\dagger \rangle$ need not be equal on the nose. However, it holds up to natural isomorphism, giving an analog of proposition 4.2.10 for solutions to parametrized equations. Proposition 4.3.4 gives a new 2-categorical formulation of the fixed-point identity. Not only do we have a natural isomorphism $F^\dagger \cong F \circ \langle \mathsf{id}, F^\dagger \rangle$ for each F, but these natural isomorphisms assemble to form a modification, i.e., a morphism between the 2-natural transformations $(\cdot)^\dagger$ and UNR.

Proposition 4.3.4 (Fixed-Point Identity). *Let* **E** *be an* **IFP***-category. There is a modification*

$$\mathsf{Unfold} : (\cdot)^\dagger \to \mathsf{UNR} : \mathbf{Cat}[- \times \mathbf{E} \to_\omega \mathbf{E}] \Rightarrow \mathbf{Cat}[- \to \mathbf{E}] : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{Cat}$$

that is an isomorphism; we call its inverse Fold. *For each category* **D***, parametrized* ω*-functor* $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$*, and object* D *of* **D***, the corresponding component is the isomorphism*

$$(\mathsf{Unfold}^F_{\mathbf{D}})_D = \mathsf{unfold}_{(\perp, \perp, F(D, -))} : F^\dagger D \to F(D, F^\dagger D)$$

given by proposition 4.2.10.

Unfold *restricts to a modificative isomorphism* $(\cdot)^\dagger \to \mathsf{UNR} : \mathbf{IFP}[- \times \mathbf{E} \to \mathbf{E}] \Rightarrow \mathbf{IFP}[- \to \mathbf{E}] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP}$.

Proof (sketch). We must show that for each small category **D**, we have a 2-cell

$$\mathsf{Unfold}_{\mathbf{D}} : (\cdot)^\dagger_{\mathbf{D}} \Rightarrow \mathsf{UNR}_{\mathbf{D}} : \mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \to \mathbf{Cat}[\mathbf{D} \to \mathbf{E}]$$

such that for all $G : \mathbf{C} \to \mathbf{D}$, the two following 2-cells are equal:

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow[\mathsf{UNR}_{\mathbf{D}}]{(\cdot)^\dagger_{\mathbf{D}}} \Downarrow \mathsf{Unfold}_{\mathbf{D}} \quad \mathbf{Cat}[\mathbf{D} \to \mathbf{E}] \xrightarrow{\mathbf{Cat}[G \to \mathbf{E}]} \mathbf{Cat}[\mathbf{C} \to \mathbf{E}],$$

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow{\mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]} \mathbf{Cat}[\mathbf{C} \times \mathbf{E} \to_\omega \mathbf{E}] \underset{\mathsf{UNR}_\mathbf{C}}{\overset{(\cdot)^\dagger_\mathbf{C}}{\Downarrow \mathsf{Unfold}_\mathbf{C}}} \mathbf{Cat}[\mathbf{C} \to \mathbf{E}].$$

It follows easily from proposition 4.2.10 that $\mathsf{Unfold}_\mathbf{D}$ is a 2-cell. To see that it satisfies the desired equality, consider some arbitrary $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$ and object C of $\mathbf{C}$. Then the F, C-component of the top 2-cell is

$$\begin{aligned}
&\left(\mathsf{Unfold}^F_\mathbf{D} G\right)_C \\
&= \left(\mathsf{Unfold}^F_\mathbf{D}\right)_{GC} \\
&= \mathsf{unfold}_{(\bot, \bot, F(GC, -))} \\
&= \mathsf{unfold}_{(\bot, \bot, (F \circ (G \times \mathsf{id}_\mathbf{E}))(C, -))} \\
&= \left(\mathsf{Unfold}^{F \circ (G \times \mathsf{id}_\mathbf{E})}_\mathbf{C}\right)_C,
\end{aligned}$$

which we recognize as the F, C-component of the bottom 2-cell. Because F, C, and G were chosen arbitrarily, we conclude the desired equality and that Unfold is a modification. It is clearly an isomorphism, and the restriction clearly has the desired properties. □

Proof. We begin by showing that Unfold is a modification. We must show that for each small category $\mathbf{D}$, we have a 2-cell

$$\mathsf{Unfold}_\mathbf{D} : (\cdot)^\dagger_\mathbf{D} \Rightarrow \mathsf{UNR}_\mathbf{D} : \mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \to \mathbf{Cat}[\mathbf{D} \to \mathbf{E}]$$

such that for all $G : \mathbf{C} \to \mathbf{D}$, the two following 2-cells are equal:

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \underset{\mathsf{UNR}_\mathbf{D}}{\overset{(\cdot)^\dagger_\mathbf{D}}{\Downarrow \mathsf{Unfold}_\mathbf{D}}} \mathbf{Cat}[\mathbf{D} \to \mathbf{E}] \xrightarrow{\mathbf{Cat}[G \to \mathbf{E}]} \mathbf{Cat}[\mathbf{C} \to \mathbf{E}],$$

$$\mathbf{Cat}[\mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}] \xrightarrow{\mathbf{Cat}[G \times \mathbf{E} \to_\omega \mathbf{E}]} \mathbf{Cat}[\mathbf{C} \times \mathbf{E} \to_\omega \mathbf{E}] \underset{\mathsf{UNR}_\mathbf{C}}{\overset{(\cdot)^\dagger_\mathbf{C}}{\Downarrow \mathsf{Unfold}_\mathbf{C}}} \mathbf{Cat}[\mathbf{C} \to \mathbf{E}].$$

In particular, we must show that for all $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$,

$$\mathsf{Unfold}^F_\mathbf{D} G = \mathsf{Unfold}^{F \circ (G \times \mathsf{id}_\mathbf{E})}_\mathbf{C} \tag{41}$$

define equal natural transformations from $F^\dagger \circ G$ to $F^\dagger \circ \langle \mathsf{id}_\mathbf{D}, F^\dagger \rangle \circ G$.

We begin by showing that $\mathsf{Unfold}_\mathbf{D}$ is a 2-cell, i.e., that $\mathsf{Unfold}^F_\mathbf{D}$ is natural in F, i.e., that for any natural transformation $\eta : F \Rightarrow G : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$, the following square commutes:

$$\begin{array}{ccc}
F^\dagger & \xRightarrow{\mathsf{Unfold}^F_\mathbf{D}} & F \circ \langle \mathsf{id}, F^\dagger \rangle \\
{\scriptstyle \eta^\dagger}\Downarrow & & \Downarrow{\scriptstyle \eta * \langle \mathsf{id}, \eta^\dagger \rangle} \\
G^\dagger & \xRightarrow{\mathsf{Unfold}^G_\mathbf{D}} & G \circ \langle \mathsf{id}, G^\dagger \rangle
\end{array}$$

This square commutes if and only if every component does, i.e., if and only if for every object D of $\mathbf{D}$, the following square commutes:

$$\begin{array}{ccc}
F^\dagger D & \xrightarrow{\left(\mathsf{Unfold}^F_\mathbf{D}\right)_D} & (F \circ \langle \mathsf{id}, F^\dagger \rangle) D \\
{\scriptstyle (\eta^\dagger)_D}\downarrow & & \downarrow{\scriptstyle (\eta * \langle \mathsf{id}, \eta^\dagger \rangle)_D} \\
G^\dagger D & \xrightarrow{\left(\mathsf{Unfold}^G_\mathbf{D}\right)_D} & (G \circ \langle \mathsf{id}, G^\dagger \rangle) D.
\end{array}$$

It is exactly the following square:

$$\begin{array}{ccc} \mathsf{GFIX}(\bot,\bot,F_D) & \xrightarrow{\mathsf{unfold}_{(\bot,\bot,F_D)}} & \mathsf{UNF}(\bot,\bot,F_D) \\ {\scriptstyle \mathsf{GFIX}(\mathsf{id},(\Lambda\eta)_D)}\downarrow & & \downarrow{\scriptstyle \mathsf{UNF}(\mathsf{id},(\Lambda\eta)_D)} \\ \mathsf{GFIX}(\bot,\bot,G_D) & \xrightarrow{\mathsf{unfold}_{(\bot,\bot,G_D)}} & \mathsf{UNF}(\bot,\bot,G_D). \end{array}$$

It commutes by proposition 4.2.10. Because F and D were arbitrary, we conclude that $\mathsf{Unfold}_{\mathbf{D}}$ is a 2-cell.

We now show that Consider some arbitrary object C of $\mathbf{C}$. Then the C-component is

$$\begin{aligned} &\left(\mathsf{Unfold}^F_{\mathbf{D}} G\right)_C \\ &= \left(\mathsf{Unfold}^F_{\mathbf{D}}\right)_{GC} \\ &= \mathsf{unfold}_{(\bot,\bot,F(GC,-))} \\ &= \mathsf{unfold}_{(\bot,\bot,(F\circ(G\times \mathsf{id}_{\mathbf{E}}))(C,-))} \\ &= \left(\mathsf{Unfold}^{F\circ(G\times\mathsf{id}_{\mathbf{E}})}_{\mathbf{C}}\right)_C. \end{aligned}$$

Because F and C were chose arbitrarily, we conclude eq. (41). Because G was chose arbitrarily, we conclude that Unfold is a modification.

The modification Unfold is clearly an isomorphism: for each $\mathbf{D}$, the component $\mathsf{Unfold}_{\mathbf{D}} : (\cdot)^\dagger_{\mathbf{D}} \Rightarrow \mathsf{UNR}_{\mathbf{D}}$ is an isomorphism. Indeed, each F-component $\mathsf{Unfold}^F_{\mathbf{D}}$ is an isomorphism, for each of its D-components $\left(\mathsf{Unfold}^F_{\mathbf{D}}\right)_D$ is an isomorphism by proposition 4.2.10.

The modification Unfold clearly restricts to

$$(\cdot)^\dagger \to \mathsf{UNR} : \mathbf{IFP}\left[- \times \mathbf{E} \to \mathbf{E}\right] \Rightarrow \mathbf{IFP}\left[- \to \mathbf{E}\right] : \mathbf{IFP}^{\mathrm{op}} \to \mathbf{IFP},$$

all while remaining a modification and an isomorphism. □

Proposition 4.3.4 abstracts considerable information. We unpack its definitions to get several corollaries. The first corollary is a special case of [LMZ19, Theorem 4.4.8] when N and M are identity functors. It will be key to defining the interpretations of recursive session types in chapter 8.

Corollary 4.3.5. *Let* $\mathbf{D}$ *be a small category. Then* $\mathsf{Unfold}^F_{\mathbf{D}}$ *and* $\mathsf{Fold}^F_{\mathbf{D}}$ *are natural in* F*, i.e., given any natural transformation* $\eta : F \Rightarrow G : \mathbf{D} \times \mathbf{E} \to \mathbf{E}$*, the two following squares commute:*

$$\begin{array}{ccc} F^\dagger & \xRightarrow{\mathsf{Unfold}^F_{\mathbf{D}}} & F \circ \langle \mathsf{id}, F^\dagger \rangle \\ {\scriptstyle \eta^\dagger}\Downarrow & & \Downarrow{\scriptstyle \eta * \langle \mathsf{id}, \eta^\dagger \rangle} \\ G^\dagger & \xRightarrow{\mathsf{Unfold}^G_{\mathbf{D}}} & G \circ \langle \mathsf{id}, G^\dagger \rangle \end{array} \qquad \begin{array}{ccc} F \circ \langle \mathsf{id}, F^\dagger \rangle & \xRightarrow{\mathsf{Fold}^F_{\mathbf{D}}} & F^\dagger \\ {\scriptstyle \eta * \langle \mathsf{id}, \eta^\dagger \rangle}\Downarrow & & \Downarrow{\scriptstyle \eta^\dagger} \\ G \circ \langle \mathsf{id}, G^\dagger \rangle & \xRightarrow{\mathsf{Fold}^G_{\mathbf{D}}} & G^\dagger \end{array}$$

Corollary 4.3.6 gives identities that will be useful in chapter 8. Equations (42) to (44) are immediate from the definitions of 2-natural transformation and proposition 4.3.1. Equations (45) and (46) are immediate from the definition of modification and proposition 4.3.4.

Corollary 4.3.6 (Parameter Identity). *Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be small categories and let* $\mathbf{E}$ *be an* **IFP***-category. Let* $F, H : \mathbf{D} \times \mathbf{E} \to \mathbf{E}$ *be parametrized* ω*-functors and let* $G, I : \mathbf{C} \to \mathbf{D}$ *be functors. Set* $F_G = F \circ (G \times \mathsf{id}_E) : \mathbf{C} \times \mathbf{E} \to \mathbf{E}$*, and analogously for* H_I*. Let* $\phi : F \Rightarrow H$ *and* $\gamma : G \Rightarrow I$ *be natural transformations. Then*

$$F^\dagger_G = F^\dagger \circ G : \mathbf{C} \to \mathbf{E}, \tag{42}$$

$$F_G \circ \langle \mathsf{id}_{\mathbf{C}}, F^\dagger_G \rangle = F \circ \langle \mathsf{id}_{\mathbf{D}}, F^\dagger \rangle \circ G : \mathbf{C} \to \mathbf{E}, \tag{43}$$

$$\left(\phi * (\gamma \times \mathsf{id}_{\mathbf{E}})\right)^\dagger = \phi^\dagger * \gamma : F^\dagger_G \Rightarrow H^\dagger_I, \tag{44}$$

$$\mathsf{Fold}^{F_G}_{\mathbf{C}} = \mathsf{Fold}^F_{\mathbf{D}} G : F_G \circ \langle \mathsf{id}_{\mathbf{C}}, F^\dagger_G \rangle \Rightarrow F^\dagger_G, \tag{45}$$

$$\mathsf{Unfold}^{F_G}_{\mathbf{C}} = \mathsf{Unfold}^F_{\mathbf{D}} G : F^\dagger_G \Rightarrow F_G \circ \langle \mathsf{id}_{\mathbf{C}}, F^\dagger_G \rangle. \tag{46}$$

Proposition 4.3.7 generalizes corollary 4.2.14 to parametrized fixed points. Given a horizontal morphism $f : A \times B \to B$ in a 2-cartesian category, an **f-algebra** [BÉ95, Definition 2.3] is a pair (g, u) where $g : A \to B$ is a horizontal morphism and $u : f * \langle \mathsf{id}_A, g \rangle \Rightarrow g$ is vertical. An **f-algebra homomorphism** $(g, u) \to (h, v)$ is a vertical morphism $w : g \Rightarrow h$ such that $w \cdot u = v \cdot (f * \langle \mathsf{id}_A, w \rangle)$. These f-algebras and f-algebra homomorphisms form a category. If we restrict our attention to the 2-cartesian category **Cat**, we get the parametrized F-algebras of [Fio94, Definition 6.1.8]. By additionally requiring $A = \mathbf{1}$, we recover the usual notion of F-algebras.

Proposition 4.3.7. *Let* $\mathbf{D}$ *be a category and* $\mathbf{E}$ *be an* **IFP***-category. Let* $F : \mathbf{D} \times \mathbf{E} \to_\omega \mathbf{E}$ *be a parametrized* ω*-functor. The initial* F*-algebra is* $\left(F^\dagger, \mathsf{Fold}^F_{\mathbf{D}}\right)$*. Given any other* F*-algebra* (G, γ)*, the mediating morphism* $\phi : F^\dagger \to G$ *is a natural transformation. The component* ϕ_D *is the unique* $F(D, -)$*-algebra homomorphism* $\left(F^\dagger D, \left(\mathsf{Fold}^F_{\mathbf{D}}\right)_D\right) \to (GD, \gamma_D)$ *given by corollary 4.2.14.*

Proof. Let (G, γ) be an arbitrary F-algebra. We begin by showing that there exists an F-algebra homomorphism $\phi : \left(F^\dagger, \mathsf{Fold}^F_{\mathbf{D}}\right) \to (G, \gamma)$. Given an object D of $\mathbf{D}$, write F_D for the partial application $F(D, -)$. For every object D, $\left(F^\dagger D, \left(\mathsf{Fold}^F_{\mathbf{D}}\right)_D\right)$ is the initial F_D-algebra by corollary 4.2.14. This implies that there exists a unique F_D-homomorphism $\phi_D : F^\dagger D \to GD$ making the following square commute:

$$\begin{array}{ccc} F_D(F^\dagger D) & \xrightarrow{\left(\mathsf{Fold}^F_{\mathbf{D}}\right)_D} & F^\dagger D \\ {\scriptstyle F_D \phi_D}\downarrow & & \downarrow{\scriptstyle \phi_D} \\ F_D(GD) & \xrightarrow{\gamma_D} & GD. \end{array}$$

We claim that these morphisms ϕ_D assemble into a natural transformation $\phi : F^\dagger \to G$. It will immediately follow that ϕ is an F-algebra homomorphism from $(F^\dagger, \mathsf{Fold})$ to (G, γ).

To show that ϕ is natural, let $f : A \to B$ be an arbitrary morphism in $\mathbf{D}$. We must show that the following square commutes:

$$\begin{array}{ccc} F^\dagger A & \xrightarrow{\phi_A} & GA \\ {\scriptstyle F^\dagger f}\downarrow & & \downarrow{\scriptstyle Gf} \\ F^\dagger B & \xrightarrow{\phi_B} & GB. \end{array}$$

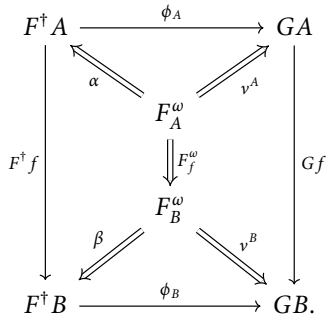
Recall that, given an $L : \mathbf{E} \to \mathbf{E}$, we write L^ω for the functor $\Omega(\bot, \bot, L)$. Let $\alpha : F^\omega_A \Rightarrow F^\dagger A$ and $\beta : F^\omega_B \to F^\dagger B$ be colimiting. Let

$$(\nu^A, GA) = \mathsf{Cone}(\cdot F_A)(GA, \gamma_A),$$
$$(\nu^B, GB) = \mathsf{Cone}(\cdot F_B)(GB, \gamma_B)$$

be cocones on F^ω_A and F^ω_B, respectively, induced by proposition 4.2.13. By this same proposition, ϕ_A and ϕ_B are cocone morphisms

$$\phi_A : (\alpha, F^\dagger A) \to (\nu^A, GA) \text{ in } \int \mathsf{Cone}(F^\omega_A, -),$$
$$\phi_B : (\beta, F^\dagger B) \to (\nu^B, GB) \text{ in } \int \mathsf{Cone}(F^\omega_B, -).$$

Write F_f for the natural transformation $\Lambda F f : F_A \Rightarrow F_B$. We then have the following diagram in $\mathbf{E}$:



(47)

We show that $\phi_B \circ F^\dagger f$ and $Gf \circ \phi_A$ are both mediating morphisms from the colimiting cone α to the cocone $\nu^B \circ F^\omega_f$. It will then follow by uniqueness of mediating morphisms that they are equal and that ϕ is natural.

We begin with $\phi_B \circ F^\dagger f$. By definition of $F^\dagger f$, $F^\dagger f$ is a mediating morphism from α to $\beta \circ F^\omega_f$. By the remarks above, ϕ_B is a mediating morphism from β to ν^B, so it is also a mediating morphism from $\beta \circ F^\omega_f$ to $\nu^B \circ F^\omega_f$. So going around the left and bottom sides of diagram 47, we get a mediating morphism $\phi_B \circ F^\dagger f$ from α to $\nu^B \circ F^\omega_f$.

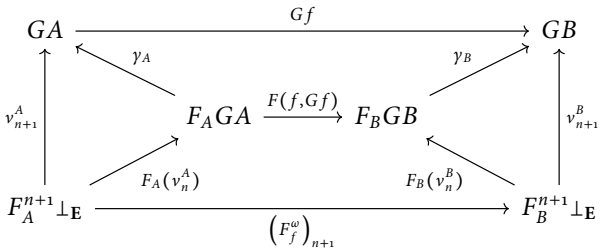
We next show that $Gf \circ \phi_A$ is a mediating morphism. By definition, ϕ_A is a mediating morphism from α to ν^A. We must now show that Gf is a mediating morphism from ν^A to $\nu^B \circ F^\omega_f$, i.e., we must show that for all n,

$$Gf \circ \nu^A_n = \nu^B_n \circ \left(F^\omega_f\right)_n : F^n_A\bot \to GB \tag{48}$$

are equal morphisms. We do so by induction on n. When $n = 0$, initiality gives us

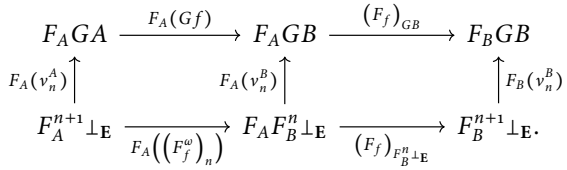
$$Gf \circ \nu^A_0 = \bot_{GB} = \nu^B_0 \circ \left(F^\omega_f\right)_0 .$$

Assume the result for some n. To show the result for $n+1$ we must show that the outer rectangle of diagram 49 commutes:



(49)

The upper trapezoid commutes by definition of F-algebra and the assumption that (G, γ) was an F-algebra. The two triangles of diagram 49 commute by definition of ν^A_{n+1} and ν^B_{n+1} (see proposition 4.2.13). The bottom trapezoid is equal to the perimeter of diagram 50:



(50)

Indeed, the top morphism of diagram 50 is exactly $F(f, Gf)$:

$$F(f, Gf) = F(f, \mathsf{id}_{GB}) \circ F(\mathsf{id}_A, Gf) = \left(F_f\right)_{GB} \circ F_A(Gf).$$

The bottom morphisms are equal by definition of F_f^ω (eq. (32)):

$$\begin{aligned}
&\left(F_f^\omega\right)_{n+1} \\
&= \Omega(\mathsf{id}_\perp, F_f^\omega)_{n+1} \\
&= F_f^{(n+1)} * \mathsf{id}_{\perp_{\mathbf{E}}} \\
&= \left(F_f^{(n+1)}\right)_{\perp_{\mathbf{E}}} \\
&= \left(F_f * F_f^{(n)}\right)_{\perp_{\mathbf{E}}} \\
&= \left(F_f\right)_{F_B^n \perp_{\mathbf{E}}} \circ F_A\left(\left(F_f^{(n)}\right)_{\perp_{\mathbf{E}}}\right) \\
&= \left(F_f\right)_{F_B^n \perp_{\mathbf{E}}} \circ F_A\left(\left(F_f^\omega\right)_n\right).
\end{aligned}$$

To see that diagram 50 commutes, we note that the left square commutes by applying F_A to the square given by the induction hypothesis. The right square commutes by naturality of F_f. By pasting, the perimeter commutes. So we conclude that the bottom trapezoid of diagram 49 commutes.

By pasting the two trapezoids and two triangles, we get that diagram 49 commutes. Equation (48) then holds by induction, so Gf is a mediating morphism from ν^A to $\nu^B \circ F_f^\omega$. By composing around the top and right sides of diagram 47, we get a mediating morphism from α to $\nu^B \circ F_f^\omega$.

By the remarks following diagram 47, we conclude that ϕ is a natural transformation from $F^\dagger$ to G.

We show that ϕ is unique. Consider any F-algebra homomorphism $\alpha : \left(F^\dagger, \mathsf{Fold}_{\mathbf{D}}^F\right) \Rightarrow (G, \gamma)$, and let D be an arbitrary object of $\mathbf{D}$. By definition of F-algebra homomorphism, the following square then commutes:

$$\begin{array}{ccc}
F_D(F^\dagger D) & \xrightarrow{\left(\mathsf{Fold}_{\mathbf{D}}^F\right)_D} & F^\dagger D \\
{\scriptstyle F_D \alpha_D}\downarrow & & \downarrow{\scriptstyle \alpha_D} \\
F_D(GD) & \xrightarrow{\gamma_D} & GD.
\end{array}$$

This implies that $\alpha_D : \left(F^\dagger D, \left(\mathsf{Fold}_{\mathbf{D}}^F\right)_D\right) \to (GD, \gamma_D)$ is an F_D-algebra homomorphism. But ϕ_D is the unique such morphism, so $\alpha_D = \phi_D$. Because D was an arbitrary component, we conclude $\alpha = \phi$.

Having established that there exists a unique morphism $\phi : \left(F^\dagger, \mathsf{Fold}_{\mathbf{D}}^F\right) \Rightarrow (G, \gamma)$ for all F-algebras (G, γ), we conclude that $\left(F^\dagger, \mathsf{Fold}_{\mathbf{D}}^F\right)$ is the initial F-algebra. □

Proposition 4.3.7 presents the converse of a class of external daggers on horizontal morphisms considered in [BÉ95, § 2.2]. Given a horizontal morphism $f : A \times B \to B$ in a 2-cartesian category, they define $f^\dagger = g$ where (g, ν) is the initial f-algebra. They do not consider the action of this dagger on vertical morphisms. In contrast, we gave a dagger operation that determines initial f-algebras. It induces an action on both horizontal and vertical morphisms. By proposition 4.3.4 and corollary 4.3.6, its action on vertical morphisms coheres with its action on horizontal morphisms.

4.4. Conway Identities

Semantics of programming languages should, ultimately, help users reason about programs. To this end, it is useful to have an arsenal of identities for the mathematical objects used to define the semantics. In our case, the semantics of recursive types motivated the definition of a dagger operation in section 4.3. In that section, we studied its 2-categorical properties. We now show how these 2-categorical properties imply a large class of identities useful for reasoning about recursive types. In particular, we show that they imply the *Conway identities* [BÉ95; BÉ96] up to isomorphism.

The Conway identities in turn imply a class of identities useful in the semantics of programming languages.

The Conway identities are also of independent interest. For example, the (cartesian) Conway identities together with an additional identity axiomatize the class of iteration theories [BÉ96, Remark 3.4]. Moreover, Hasegawa [Has99, Theorem 7.1] and Hyland independently discovered [BH03, p. 281] that a cartesian category has a trace operator [JSV96] if and only if it has an external dagger operator satisfying the (cartesian) Conway identities.

We begin by presenting the Conway identities. The identities' names vary in the literature. We give those of Bloom and Ésik [BÉ95; BÉ96] and of Simpson and Plotkin [SP00, Definitions 2.2 and 2.4]. An external dagger † satisfies:

(1) the **parameter identity** or **naturality** if for all $f : B \times C \to C$ and $g : A \to B$, $(f \circ (g \times \mathsf{id}_C))^\dagger = f^\dagger \circ g$.
(2) the **composition identity** or **parametrized dinaturality** if for all $f : P \times A \to B$ and $g : P \times B \to A$, $(g \circ \langle \pi_P^{P\times A}, f\rangle)^\dagger = g \circ \langle \mathsf{id}_P, (f \circ \langle \pi_P^{P\times B}, g\rangle)^\dagger\rangle$.
(3) the **double dagger identity** or **diagonal property** if for all $f : A \times B \times B \to B$, $(f^\dagger)^\dagger = (f \circ (\mathsf{id}_A \times \langle \mathsf{id}_B, \mathsf{id}_B\rangle))^\dagger$.
(4) the **abstraction identity** if the following diagram commutes:

$$\begin{array}{ccccc} [A \times B \times C \to C] & & \xrightarrow{\quad [\mathsf{id}_A \times \langle \pi_B, \mathsf{ev}_{B,C}\rangle \to \mathsf{id}_C] \quad} & & [A \times [B \to C] \times B \to C] \\ {\scriptstyle \dagger_{A\times B,C}}\downarrow & & & & \downarrow{\scriptstyle \Lambda} \\ [A \times B \to C] & \xrightarrow{\Lambda} & [A \to [B \to C]] & \xleftarrow{\dagger_{A,[B\to C]}} & [A \times [B \to C] \to [B \to C]] \end{array}$$

(5) the **power identities** if for all $f : A \times B \to B$ and $n > 1$, $(f^n)^\dagger = f^\dagger$, where $f^n : A \times B \to B$ is inductively defined by $f^0 = \pi_B^{A\times B}$ and $f^{n+1} = f \circ \langle \pi_A^{A\times B}, f^n\rangle$.

An external dagger satisfies the cartesian Conway identities if it satisfies properties 1 to 3. It satisfies **Conway identities** if it additionally satisfies property 4. Theorem 4.4.1 answers the last question of this chapter's introduction:

THEOREM 4.4.1. *The external dagger operation of proposition 4.3.1 satisfies the Conway identities and the power identities up to isomorphism.*

Proof. The category **IFP** is 2-cartesian closed. By proposition 4.3.7, each ω-functor $F : \mathbf{D} \times \mathbf{E} \to \mathbf{E}$ in **IFP** has an initial F-algebra $\left(F^\dagger, \mathsf{Fold}_{\mathbf{D}}^F\right)$. By corollary 4.3.6, these initial algebras are related such that $\left(F^\dagger \circ G, \mathsf{Fold}_{\mathbf{D}}^F G\right)$ is the initial $(F \circ (G \times \mathsf{id}_{\mathbf{E}}))$-algebra for each $G : \mathbf{C} \to \mathbf{D}$. It then follows by [BÉ95, Theorem 7.1] that the external dagger operator induced by proposition 4.3.1 satisfies the Conway identities and the power identities up to isomorphism. □

The Conway identities imply the **pairing identity**, sometimes called Bekič's identity [BÉ96, p. 10], which relates the two main approaches for solving systems of simultaneous equations. Consider such a system

$$\begin{aligned} B &\cong F(A, B, C) \\ C &\cong G(A, B, C). \end{aligned}$$

We can solve it by pairing F and G, and solving the single equation $(B, C) \cong \langle F, G\rangle(A, B, C)$. Alternatively, we can use a Gaussian-elimination-style approach, e.g., as we did in the introduction for the functors defining data types `even` and `odd`. The pairing identity tells us that these two approaches yield isomorphic solutions:

COROLLARY 4.4.2 (Pairing Identity). *Let* $\mathbf{A}$, $\mathbf{B}$, *and* $\mathbf{C}$ *be small* **IFP***-categories, and let* $F : \mathbf{A} \times \mathbf{B} \times \mathbf{C} \to \mathbf{B}$ *and* $G : \mathbf{A} \times \mathbf{B} \times \mathbf{C} \to \mathbf{C}$ *be* ω*-functors. Set* $H = \mathbf{A} \times \mathbf{B} \xrightarrow{\langle \mathsf{id}_{\mathbf{A}}, G^\dagger\rangle} \mathbf{A} \times \mathbf{B} \times \mathbf{C} \xrightarrow{F} \mathbf{B}$. *Then*

$$\langle F, G\rangle^\dagger \cong \langle G^\dagger \circ \langle \mathsf{id}_{\mathbf{A}}, H^\dagger\rangle, H^\dagger\rangle : \mathbf{A} \to \mathbf{B} \times \mathbf{C}.$$

The Conway identities also imply the **left zero identity** [BÉ96, p. 10]. Semantically, it describes the interplay between weakening and the formation of recursive types.

Corollary 4.4.3 (Left Zero Identity). *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be small* **IFP***-categories, and let* $F : \mathbf{B} \to \mathbf{A}$ *be an* ω*-functor. Then*

$$\left(\mathbf{A} \times \mathbf{B} \xrightarrow{\pi_{\mathbf{B}}} \mathbf{B} \xrightarrow{F} \mathbf{A}\right)^{\dagger} \cong \mathbf{B} \xrightarrow{F} \mathbf{A}.$$

4.5. Canonical and Parametrized Fixed Points for O-Categories

In this section, we consider an order-theoretic variation of sections 4.2 and 4.3. We do so in the setting of **O**-categories and locally continuous functors introduced in section 2.2.2. This setting generalizes categories of domains to provide just the amount of order-theoretic structure required for taking fixed points of functors. **O**-categories are more concrete than ω-categories, and their order-theoretic characterization of ω-colimits and of ω-functors is useful in applications. They also have enough structure to have *canonical* fixed points.

4.5.1. Local Continuity and ω-Continuity. Locally continuous functors preserve **O**-colimits. Every locally continuous functor $F : \mathbf{D} \to \mathbf{E}$ restricts to a functor $F^e : \mathbf{D}^e \to \mathbf{E}^e$ [SP82, Lemma 4]. When $\mathbf{D}$ is **O**-cocomplete, F^e is an ω-functor [SP82, Theorem 3] and $\mathbf{D}^e$ is an ω-category by proposition 2.2.63. These observations raise the question: why not use $\mathbf{Links}_{\mathbf{K}^e}$ and the results of sections 4.2 and 4.3 to study fixed points of locally continuous functors?

The reason is that such an approach does not handle all natural transformations between locally continuous functors, but only those between functors on $\mathbf{K}^e$. This is because natural transformations $\eta : F \Rightarrow G$ do not in general restrict to natural transformations $F^e \Rightarrow G^e$. By adapting the techniques of the previous sections to **O**-categories and locally continuous functors, we get fixed-point operators defined on all natural transformations between locally continuous functors. The fixed-point operators are also themselves locally continuous.

4.5.2. Canonical Fixed Points. By slightly modifying our category of links, we can construct canonical fixed points. Given a functor $F : \mathbf{K} \to \mathbf{K}$, we say that a fixed point $f : FX \cong X$ is **canonical** if (X, f) is an initial F-algebra and (X, f^{-1}) is a terminal F-coalgebra. Given an **O**-category $\mathbf{K}$, let $\mathbf{OLinks}_{\mathbf{K}}$ be the category where

- objects are triples (K, k, F) called "links", where K is an object of $\mathbf{K}$, $F : \mathbf{K} \to \mathbf{K}$ is locally continuous, and $k : K \to FK$ is an embedding;
- morphisms and composition are defined as before.

Proposition 4.5.1. *Equations* (29) *to* (32) *define a locally continuous functor* $\Omega : \mathbf{OLinks}_{\mathbf{K}} \to \mathbf{O}[\omega \to \mathbf{K}]$. *For all links* (K, k, F), $\Omega(K, k, F) : \omega \to \mathbf{K}^e$. *The natural transformation* $\Omega(f, \eta)$ *lies in* $\mathbf{K}^e$ *whenever* f *and* η *do.*

Let $\mathbf{O}[\omega \to \mathbf{K}^e \hookrightarrow \mathbf{K}]$ be the subcategory of $\mathbf{O}[\omega \to \mathbf{K}]$ whose objects are functors $\omega \to \mathbf{K}^e$ and whose morphisms are natural transformations in $\mathbf{K}$. It is an **O**-category.

Proposition 4.5.2. *Let* $\mathbf{K}$ *be an* **O***-cocomplete* **O***-category. A choice of* **O***-colimit in* $\mathbf{K}$ *for each diagram* $\omega \to \mathbf{K}^e \hookrightarrow \mathbf{K}$ *defines the action on objects of a locally continuous functor* $\operatorname{colim}_\omega : [\omega \to \mathbf{K}^e \hookrightarrow \mathbf{K}] \to \mathbf{K}$. *The morphism* $\operatorname{colim}_\omega \eta$ *lies in* $\mathbf{K}^e$ *whenever* η *does.*

Proof (sketch). The action of $\operatorname{colim}_\omega$ on morphisms follows immediately from proposition 2.2.63. Indeed, where $\phi : \Phi \Rightarrow \operatorname{colim}_\omega \Phi$ and $\gamma : \Gamma \Rightarrow \operatorname{colim}_\omega \Gamma$ are the chosen **O**-colimits in $\mathbf{K}$, a natural transformation $\eta : \Phi \Rightarrow \Gamma$ induces a cocone $\gamma \circ \eta : \Phi \Rightarrow \operatorname{colim}_\omega \Gamma$. By proposition 2.2.63, the unique mediating morphism of cocones is then:

$$\operatorname{colim}_\omega(\eta : \Phi \Rightarrow \Gamma) = \bigsqcup_{n \in \mathbb{N}}^{\uparrow} \gamma_n \circ \eta_n \circ \phi_n^p.$$

This action on morphisms is easily seen to be locally continuous. □

Remark 4.5.3. The reader may ask: why do we specify in proposition 4.5.2 and elsewhere that "the morphism $F\eta$ lies in $\mathbf{K}^e$ whenever η does"? If F is locally continuous and locally continuous functors take embeddings to embeddings, is it not an immediate $F\eta$ is an embedding whenever η is an embedding? Indeed, it is. But our statement is more general. One must distinguish between "natural families of embeddings" (natural transformations whose every component is an embedding) and "natural embeddings" (a natural transformation that is an embedding in the corresponding functor category). Though every natural embedding is a natural family of embeddings, the converse is false: the corresponding family of projections need not be natural.

Proposition 4.5.4. *Let* $\mathbf{K}$ *be an* $\mathbf{O}$*-cocomplete* $\mathbf{O}$*-category. The functor* $\mathsf{GFIX} = \mathrm{colim}_\omega \circ \Omega : \mathbf{OLinks_K} \to \mathbf{K}$ *is locally continuous. The morphism* $\mathsf{GFIX}(f, \eta)$ *lies in* $\mathbf{K}^e$ *whenever* f *and* η *do.*

The recipe given by proposition 4.2.9 gives a locally continuous functor $\mathsf{UNF} : \mathbf{OLinks_K} \to \mathbf{K}$. Again, $\mathsf{UNF}(f, \eta)$ lies in $\mathbf{K}^e$ whenever f and η do. The functors GFIX and UNF are related by the same natural isomorphism as proposition 4.2.10.

We say that an $\mathbf{O}$-category $\mathbf{K}$ has **strict morphisms** if it has zero morphisms and o_{AB} is the least element of $\mathbf{K}(A, B)$ for all objects A and B. We say that $\mathbf{K}$ **supports canonical fixed points** if it has an initial object, strict morphisms, and is $\mathbf{O}$-cocomplete. Let $\mathbf{CFP}$ be the full subcategory of $\mathbf{O}$ whose objects are $\mathbf{O}$-categories that support canonical fixed points. It is also known as $\mathbf{Kind}$ [Fio94, § 7.3.2]. It is 2-cartesian closed [Fio94, Theorem 7.3.11].

Assume $\mathbf{K}$ supports canonical fixed points. Then $\bot$ is also the initial object of $\mathbf{K}^e$, and we can fully and faithfully embed $\mathbf{O}\,[\mathbf{K} \to \mathbf{K}]$ into $\mathbf{Links_K}$ using the same approach as before. This embedding is locally continuous. We define the locally continuous **canonical-fixed-point functor** $\mathsf{CFIX} : \mathbf{CAT}\,[\mathbf{CFP} \to \mathbf{K}]\,\mathbf{K} \to \mathbf{K}$ as the composition $\mathbf{CAT}\,[\mathbf{CFP} \to \mathbf{K}]\,\mathbf{K} \hookrightarrow \mathbf{Links_K} \xrightarrow{\mathsf{GFIX}} \mathbf{K}$. The following result is standard:

Proposition 4.5.5. *If* $\mathbf{K}$ *supports canonical fixed points and* F *is a locally continuous functor on* $\mathbf{K}$*, then* $\mathsf{fold} : F(\mathsf{CFIX}(F)) \to \mathsf{FIX}(F)$ *is a canonical fixed point.*

We can mimic the results of sections 4.3 and 4.4, generally replacing FIX by CFIX, ω-$\mathbf{Cat}$ by $\mathbf{O}$, $\mathbf{IFP}$ by $\mathbf{CFP}$, and ω-functor by locally continuous functor. In particular, the parametrized fixed point functor $(\cdot)^\dagger$ is locally continuous and again satisfies the Conway identities up to isomorphism. It also produces canonical parametrized families of fixed points:

Proposition 4.5.6. *Let* $\mathbf{D}$ *and* $\mathbf{E}$ *be* $\mathbf{O}$*-categories, and assume* $\mathbf{E}$ *supports canonical fixed points. Let* $F : \mathbf{D} \times \mathbf{E} \to \mathbf{E}$ *be a locally continuous functor. Then* $(F^\dagger, \mathsf{Fold})$ *and* $(F^\dagger, \mathsf{Unfold})$ *are respectively the initial* F*-algebra and terminal* F*-coalgebra.*

1. *Given any other* F*-algebra* (G, γ)*, the mediating morphism* $\phi : F^\dagger \to G$ *is a natural family of embeddings whenever* γ *is an embedding. The component* ϕ_D *is the unique* F_D*-algebra homomorphism* $(F^\dagger D, \mathsf{Fold}_D) \to (GD, \gamma_D)$.
2. *Given any other* F*-coalgebra* (Γ, γ)*, the mediating morphism* $\rho : \Gamma \to F^\dagger$ *is a natural family of projections whenever* γ *is a projection. The component* ρ_D *is the unique* F_D*-coalgebra homomorphism* $(GD, \gamma_D) \to (F^\dagger D, \mathsf{Unfold}_D)$.

Proof (Sketch). The key new result relative to proposition 4.3.7 is that ϕ is a natural family of embeddings whenever γ is an embedding. By corollary 4.2.14, ϕ_D is a mediating morphism from an $\mathbf{O}$-colimit to the cocone induced by the F-algebra (G, γ). This cocone is in $\mathbf{E}^e$ whenever γ is an embedding. In this case, ϕ_D is an embedding by proposition 2.2.63. □

4.6. Related Work

Scott [Sco72] introduced inverse limit constructions to construct fixed points of functors. In particular, Scott used an inverse limit of a chain of projections to construct a continuous lattice $D \cong [D \to D]$, so that D is isomorphic to the lattice of continuous functions from D to D. Until this point, the only tools for constructing fixed points were variations on Tarski's least fixed-point

theorem [Leh76b, p. 9]. Lehmann [Leh76b] generalized these ideas to find fixed points of ω-cocontinuous functors on ω-cocomplete categories. These ideas were further explored by Lehmann and Smyth [LS77; LS81] to give semantics to data types. We built on these ideas to define a general fixed-point functor GFIX. Using GFIX, we were able to show that a functor's fixed points assemble into a natural isomorphism. Their fixed-point functor is exactly FIX, while their parametrized fixed-point functor is our $(\cdot)^\dagger$.

Wand [Wan77] introduced the definitions of **O**-categories and locally continuous functors. Smyth and Plotkin [SP77; SP82] introduced **O**-(co)limits and generalized Scott's limit-colimit coincidence theorem to **O**-categories. **O**-categories generalize categories of domains to provide just the structure required to solve recursive domain equations in a categorical setting. Smyth and Plotkin's "basic lemma" [SP82, Lemma 2] gives a recipe for constructing fixed points of covariant locally continuous functors on **O**-categories.

Some took the existence of fixed points of functors as their starting point. Freyd [Fre91] studied algebraically complete categories, that is, categories **C** where every covariant functor $T : \mathbf{C} \to \mathbf{C}$ has an initial T-algebras. Freyd also studied properties of functors on algebraically complete categories. Freyd [Fre92] extended this analysis to algebraically compact categories, i.e., algebraically complete categories where initial algebra and terminal co-algebras are canonically isomorphic.

Fiore [Fio94] investigated axiomatic categorical domain theory for application to the denotational semantics of deterministic programming languages. In chapter 6, Fiore used initiality to define a dagger operation on functors between certain algebraically complete **O**-categories. Under certain conditions, this dagger operation is functorial. It satisfies the parameter identity on functors, i.e., it satisfies eq. (42) above. Our category **CFP** appears as the category **Kind** [Fio94, Definition 7.3.11].

Dagger operation and the Conway identities arose in a separate line of research. Iteration theories [BÉ93] were introduced to study the syntax and semantics of flowchart algorithms, and they are defined in terms of a dagger operation. Bloom and Ésik [BÉ96] studied external dagger operations on cartesian closed categories and showed that for many of the categories used in semantics, the least fixed point operator induces a dagger operation satisfying the Conway identities. They generalized this work to 2-cartesian closed categories in [BÉ95] and gave sufficient conditions for a dagger on horizontal morphisms to satisfy the Conway identities. They did not explore the 2-cartesian structure of daggers or the action of daggers on vertical morphisms.

Simpson and Plotkin [SP00] gave an axiomatic treatment of dagger operations satisfying Conway identities. They gave a purely syntactic account of free iteration theories. They give a precise characterization of the circumstances in which the iteration theory axioms are complete for categories with an iteration operator.

Linear logic enjoys other proofs-as-programs interpretations. Benton [Ben95; Ben94] introduced the LNL calculus, a mixed linear and non-linear calculus. It is interpreted by an "LNL" or "adjoint" model: a symmetric monoidal closed category and a cartesian closed category related by a pair of adjoint functors. Benton and Wadler [BW96] used this model to relate translation of the λ-calculus in Moggi's computational metalanguage [Mog91] and translations of intuitionistic logic into intuitionistic linear logic. Lindenhovius, Mislove, and Zamdzhiev [LMZ19] introduced the "linear/non-linear fixpoint calculus" (LNL-FPC), a type system with mixed linear and non-linear recursive types. They use the dagger operator of [LS81] to model arbitrary recursive types in a linear category and non-linear recursive types in a cartesian category. These two interpretations are strongly related by suitable mediating functors and natural isomorphisms, which allow them to define substructural operations on non-linear types. To give fixed points to contravariant functors, they used standard order-theoretic techniques [LS81, Theorem 3] to reduce contravariant functors to covariant functors.

4.A. General Results on ω-Categories

In this section, we present various results on ω-categories and **IFP**-categories. Many of these results are standard and we present them only for ease of reference.

Recall that the category **IFP** of small **IFP**-categories is a subcategory of ω-**Cat**, which is itself a subcategory of the category **Cat** of small categories.

PROPOSITION 4.A.1. *The categories ω-**Cat** and **IFP** inherit their terminal objects from* **Cat**.

Proof. The one-object category $\mathbf{1}$ is the terminal object of **Cat** and is clearly an ω-category and an **IFP**-category. Given any other category $\mathbf{C}$, the functor $\mathbf{C} \to \mathbf{1}$ witnessing terminality in **Cat** is clearly an ω-functor, so it lies in ω-**Cat** and **IFP** as well. Uniqueness is inherited from **Cat**. □

LEMMA 4.A.2. *If $\mathbf{A}$ and $\mathbf{B}$ are categories with initial objects $\bot_{\mathbf{A}}$ and $\bot_{\mathbf{B}}$, then $(\bot_{\mathbf{A}}, \bot_{\mathbf{B}})$ is the initial object of the product category $\mathbf{A} \times \mathbf{B}$.*

Proof. Immediate from the definition of morphism in $\mathbf{A} \times \mathbf{B}$. □

LEMMA 4.A.3. *Let Δ be a small category, and let $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ be locally small categories.*

(1) Given diagrams $J_{\mathbf{A}} : \Delta \to \mathbf{A}$ and $J_{\mathbf{B}} : \Delta \to \mathbf{B}$, and colimiting cocones $\kappa_{\mathbf{A}} : J_{\mathbf{A}} \Rightarrow A$ and $\kappa_{\mathbf{B}} : J_{\mathbf{B}} \Rightarrow B$, the cocone $(\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}) : \langle J_{\mathbf{A}}, J_{\mathbf{B}} \rangle \Rightarrow (A, B)$ is colimiting in $\mathbf{A} \times \mathbf{B}$.

(2) If $\mathbf{A}$ and $\mathbf{B}$ have all Δ-colimits, then so does the product category $\mathbf{A} \times \mathbf{B}$.

(3) The projection functors $\pi_{\mathbf{A}} : \mathbf{A} \times \mathbf{B} \to \mathbf{A}$ and $\pi_{\mathbf{B}} : \mathbf{A} \times \mathbf{B} \to \mathbf{B}$ preserve Δ-colimits.

(4) If $A : \mathbf{C} \to \mathbf{A}$ and $B : \mathbf{C} \to \mathbf{B}$ preserve Δ-colimits, then so does their pairing $\langle A, B \rangle : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$, where $\langle A, B \rangle C = (AC, BC)$ and $\langle A, B \rangle (f : C \to C') = (Af, Bf) : (AC, BC) \to (AC', BC')$.

*(5) Let Δ-**Cat** be the 2-category of small Δ-cocomplete categories. If $\mathbf{A}$ and $\mathbf{B}$ are small and Δ-cocomplete, then there is a 2-natural isomorphism*

$$\langle -, - \rangle : \Delta\text{-}\mathbf{Cat}(-, \mathbf{A}) \times \Delta\text{-}\mathbf{Cat}(-, \mathbf{B}) \cong \Delta\text{-}\mathbf{Cat}(-, \mathbf{A} \times \mathbf{B}) : \Delta\text{-}\mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT}$$

inherited from **Cat**.

(6) Assuming the axiom of choice, if the product category $\mathbf{A} \times \mathbf{B}$ is non-empty and has all Δ-colimits, then so do $\mathbf{A}$ and $\mathbf{B}$.

Proof. Let $J_{\mathbf{A}} : \Delta \to \mathbf{A}$ and $J_{\mathbf{B}} : \Delta \to \mathbf{B}$ be arbitrary diagrams of shape Δ, and assume they have colimiting cocones $\kappa_{\mathbf{A}} : J_{\mathbf{A}} \Rightarrow A$ and $\kappa_{\mathbf{B}} : J_{\mathbf{B}} \Rightarrow B$. We claim that $(\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}) : \langle J_{\mathbf{A}}, J_{\mathbf{B}} \rangle \Rightarrow (A, B)$ is colimiting in $\mathbf{A} \times \mathbf{B}$. Let $(\alpha, \beta) : \langle J_{\mathbf{A}}, J_{\mathbf{B}} \rangle \Rightarrow (A, B)$ be any other cocone in $\mathbf{A} \times \mathbf{B}$. There exist unique cocone morphisms $a : (\kappa_{\mathbf{A}}, A) \to (\alpha, A)$ and $b : (\kappa_{\mathbf{B}}, B) \to (\beta, B)$. They assemble to form the unique cocone morphism $(a, b) : ((\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}), (A, B)) \to ((\alpha, \beta), (A, B))$. This means that $((\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}), (A, B))$ is initial in $\int \mathsf{Cone}(\langle J_{\mathbf{A}}, J_{\mathbf{B}} \rangle, -)$, i.e., it is the colimit of $\langle J_{\mathbf{A}}, J_{\mathbf{B}} \rangle$ in $\mathbf{A} \times \mathbf{B}$.

Now assume that $\mathbf{A}$ and $\mathbf{B}$ have all Δ-colimits; we show that the product category $\mathbf{A} \times \mathbf{B}$ has all Δ-colimits. Let $J : \Delta \to \mathbf{A} \times \mathbf{B}$ be an arbitrary diagram of shape Δ. It determines two diagrams of shape Δ:

$$J_{\mathbf{A}} = \pi_{\mathbf{A}} \circ J : \Delta \to \mathbf{A}, \tag{51}$$

$$J_{\mathbf{B}} = \pi_{\mathbf{B}} \circ J : \Delta \to \mathbf{B}. \tag{52}$$

By hypothesis, the colimiting cocones $\kappa_{\mathbf{A}} : J_{\mathbf{A}} \Rightarrow \mathrm{colim}_{\Delta} J_{\mathbf{A}}$ and $\kappa_{\mathbf{B}} : J_{\mathbf{B}} \Rightarrow \mathrm{colim}_{\Delta} J_{\mathbf{B}}$ exist in $\mathbf{A}$ and $\mathbf{B}$, respectively. By the above, they form a colimiting cocone $(\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}) : J \Rightarrow (\mathrm{colim}_{\Delta} J_{\mathbf{A}}, \mathrm{colim}_{\Delta} J_{\mathbf{B}})$. Because J was arbitrary, we conclude that $\mathbf{A} \times \mathbf{B}$ has all Δ-colimits.

We now show that the projection functors preserve Δ-colimits. Let $J : \Delta \to \mathbf{A} \times \mathbf{B}$ be an arbitrary diagram of shape Δ. It determines two diagrams of shape Δ:

$$J_{\mathbf{A}} = \pi_{\mathbf{A}} \circ J : \Delta \to \mathbf{A}, \tag{53}$$

$$J_{\mathbf{B}} = \pi_{\mathbf{B}} \circ J : \Delta \to \mathbf{B}. \tag{54}$$

Let $\kappa : J \Rightarrow \mathrm{colim}_{\Delta} J$ be its colimiting cocone in $\mathbf{A} \times \mathbf{B}$. By definition of $\mathbf{A} \times \mathbf{B}$, $\mathrm{colim}_{\Delta} J = (A, B)$ for some objects A in $\mathbf{A}$ and B in $\mathbf{B}$. Moreover, κ is given by a pair of cocones $(\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}) : \langle J_{\mathbf{A}}, J_{\mathbf{B}} \rangle \Rightarrow (A, B)$. We show that $\pi_{\mathbf{A}} \kappa = \kappa_{\mathbf{A}} : J_{\mathbf{A}} \Rightarrow A$ is colimiting in $\mathbf{A}$; the result for $\mathbf{B}$ will follow by symmetry. Let $\alpha : J_{\mathbf{A}} \Rightarrow A'$ be any other cocone in $\mathbf{A}$. Then $(\alpha, \kappa_{\mathbf{B}}) : J \Rightarrow (A', B)$ is a cocone in $\mathbf{A} \times \mathbf{B}$ and there exists a unique cocone morphism $((a, b) : ((\kappa_{\mathbf{A}}, \kappa_{\mathbf{B}}), (A, B)) \to ((\alpha, \kappa_{\mathbf{B}}), (A', B))$. In particular,

this implies there exists a unique cocone morphism $a : (\kappa_{\mathbf{A}}, A) \to (\alpha, A')$. So $\pi_{\mathbf{A}}\kappa : \pi_{\mathbf{A}}J \Rightarrow A$ is colimiting. We conclude that $\pi_{\mathbf{A}}$ preserves Δ-colimits.

Assume $A : \mathbf{C} \to \mathbf{A}$ and $B : \mathbf{C} \to \mathbf{B}$ preserve Δ-colimits. We show that $\langle A, B\rangle : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$ preserves Δ-colimits. Let $J : \Delta \to \mathbf{C}$ be an arbitrary diagram of shape Δ, and assume that $\kappa : J \Rightarrow C$ is colimiting. Then $A\kappa : AJ \Rightarrow AC$ and $B\kappa : BJ \Rightarrow BC$ are colimiting in $\mathbf{A}$ and $\mathbf{B}$, respectively. By the above, the cocone $(A\kappa, B\kappa) : \langle AJ, BJ\rangle \Rightarrow (AC, BC)$ is colimiting in $\mathbf{A} \times \mathbf{B}$. But this cocone is exactly $\langle A, B\rangle\kappa : \langle A, B\rangle J \Rightarrow \langle A, B\rangle C$, so we conclude that $\langle A, B\rangle$ preserves Δ-colimits.

Now let $\mathbf{A}$ and $\mathbf{B}$ be small Δ-cocomplete categories. There exists a 2-natural isomorphism

$$\langle -, -\rangle : \mathbf{Cat}(-, \mathbf{A}) \times \mathbf{Cat}(-, \mathbf{B}) \cong \mathbf{Cat}(-, \mathbf{A} \times \mathbf{B}) : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT}$$

whose action on functors is the above-described pairing. Its inverse is given by the above-described projections:

$$\langle \pi_{\mathbf{A}}, \pi_{\mathbf{B}}\rangle : \mathbf{Cat}(-, \mathbf{A} \times \mathbf{B}) \cong \mathbf{Cat}(-, \mathbf{A}) \times \mathbf{Cat}(-, \mathbf{B}) : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT}.$$

We show that these 2-natural isomorphisms restrict to form a 2-natural isomorphism

$$\Delta\text{-}\mathbf{Cat}(-, \mathbf{A}) \times \Delta\text{-}\mathbf{Cat}(-, \mathbf{B}) \cong \Delta\text{-}\mathbf{Cat}(-, \mathbf{A} \times \mathbf{B}) : \Delta\text{-}\mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT}.$$

2-naturality is inherited from **Cat**, so it is sufficient to show that they give an isomorphism of categories. Let $\mathbf{C}$ be an arbitrary small Δ-cocomplete category. We show that

$$\langle -, -\rangle_{\mathbf{C}} : \Delta\text{-}\mathbf{Cat}(\mathbf{C}, \mathbf{A}) \times \Delta\text{-}\mathbf{Cat}(\mathbf{C}, \mathbf{B}) \cong \Delta\text{-}\mathbf{Cat}(\mathbf{C}, \mathbf{A} \times \mathbf{B})$$

and

$$\langle \pi_{\mathbf{A}}, \pi_{\mathbf{B}}\rangle : \Delta\text{-}\mathbf{Cat}(\mathbf{C}, \mathbf{A} \times \mathbf{B}) \cong \Delta\text{-}\mathbf{Cat}(\mathbf{C}, \mathbf{A}) \times \Delta\text{-}\mathbf{Cat}(\mathbf{C}, \mathbf{B}) : \Delta\text{-}\mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT}.$$

form an isomorphism of categories. We begin by checking that their domains and codomains are well defined. Let $\alpha : A \Rightarrow A' : \mathbf{C} \to \mathbf{A}$ and $\beta : B \Rightarrow B' : \mathbf{C} \to \mathbf{B}$ be arbitrary 2-cells in Δ-**Cat**. By the above result on pairing, $\langle A, B\rangle : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$ and $\langle A', B'\rangle : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$ both preserve Δ-colimits, so they are arrows in Δ-**Cat** (and objects in Δ-$\mathbf{Cat}(\mathbf{C}, \mathbf{A} \times \mathbf{B})$). It follows that the 2-cell

$$\langle \alpha, \beta\rangle_{\mathbf{C}} : \langle A, B\rangle_{\mathbf{C}} \Rightarrow \langle A', B'\rangle_{\mathbf{C}} : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$$

is a morphism in Δ-$\mathbf{Cat}(\mathbf{C}, \mathbf{A} \times \mathbf{B})$. In the opposite direction, let $\delta : C \Rightarrow C' : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$ be an arbitrary 2-cell in Δ-**Cat**. By the above result on projections, $\pi_{\mathbf{A}}\delta : \pi_{\mathbf{A}}C \Rightarrow \pi_{\mathbf{A}}C' : \mathbf{C} \to \mathbf{A}$ and $\pi_{\mathbf{B}}\delta : \pi_{\mathbf{B}}C \Rightarrow \pi_{\mathbf{B}}C' : \mathbf{C} \to \mathbf{B}$ are again 2-cells in Δ-**Cat**, so

$$\langle \pi_{\mathbf{A}}, \pi_{\mathbf{B}}\rangle\delta : \langle \pi_{\mathbf{A}}, \pi_{\mathbf{B}}\rangle C \Rightarrow \langle \pi_{\mathbf{A}}, \pi_{\mathbf{B}}\rangle C'$$

is a morphism in Δ-$\mathbf{Cat}(\mathbf{C}, \mathbf{A}) \times \Delta$-$\mathbf{Cat}(\mathbf{C}, \mathbf{B})$. So the domains and codomains are all well defined. A routine check gives that they remain mutual inverses, i.e., that they form an isomorphism of categories.

Now assume the axiom of choice and that the product category $\mathbf{A} \times \mathbf{B}$ has all Δ-colimits; we show that $\mathbf{A}$ and $\mathbf{B}$ have all Δ-colimits. In particular, we show that $\mathbf{A}$ has all Δ-colimits; $\mathbf{B}$ will follow by symmetry. Let $J : \Delta \to \mathbf{A}$ be an arbitrary diagram of shape Δ. Because $\mathbf{A} \times \mathbf{B}$ is non-empty, so is $\mathbf{B}$. Choose an object B of $\mathbf{B}$ and let $KB : \Delta \to \mathbf{B}$ be the constant functor onto it. Then $\langle J, KB\rangle : \Delta \to \mathbf{A} \times \mathbf{B}$ is a diagram of shape Δ in $\mathbf{A} \times \mathbf{B}$. It has a colimit, which is preserved by the projection functor $\pi_{\mathbf{A}} : \mathbf{A} \times \mathbf{B} \to \mathbf{A}$. We conclude that J has a colimit in $\mathbf{A}$. □

PROPOSITION 4.A.4. *The category* ω-**Cat** *inherits its 2-product structure from* **Cat**.

Proof. The 2-product structure in **Cat** is given by the product category structure. To show that ω-**Cat** inherits this structure, we must show that if $\mathbf{A}$ and $\mathbf{B}$ are small ω-categories, then:

(1) so is $\mathbf{A} \times \mathbf{B}$;
(2) the 2-natural isomorphism

$$\omega\text{-}\mathbf{Cat}(-, \mathbf{A}) \times \omega\text{-}\mathbf{Cat}(-, \mathbf{B}) \cong \omega\text{-}\mathbf{Cat}(-, \mathbf{A} \times \mathbf{B}) : \omega\text{-}\mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT}$$

from **Cat** is also a 2-natural isomorphism in ω-**Cat**.

The category $\mathbf{A} \times \mathbf{B}$ is an ω-category by lemma 4.A.3. The 2-natural isomorphism is inherited by lemma 4.A.3. □

Corollary 4.A.5. *The category* **IFP** *inherits its 2-product structure from* ω-**Cat**.

Proof. Immediate from lemma 4.A.2 and proposition 4.A.4. □

Lemma 4.A.6 ([Leh76a, Lemma IV.2]). *If* **A** *and* **B** *are* ω*-categories, then so is* ω-$\mathbf{Cat}(\mathbf{A}, \mathbf{B})$.

Lemma 4.A.7. *If* **A** *and* **B** *are* **IFP***-categories, then so is* $\mathbf{IFP}(\mathbf{A}, \mathbf{B})$.

Proof. Recall that limits and colimits in functor-categories are computed pointwise [Mac98, Theorem V.3.1]. It follows that category $\mathbf{IFP}(\mathbf{A}, \mathbf{B})$ has all ω-colimits (it also follows by lemma 4.A.6). To see that it also has an initial object, note that initial objects are given by the limit of the identity functor. □

Lemma 4.A.8 ([Leh76a, Lemmas IV.5 and IV.6]). *Let* **A**, **B**, *and* **C** *be small* ω*-categories. The evaluation functor*

$$\mathsf{ev}_{\mathbf{A},\mathbf{B}} : \omega\text{-}\mathbf{Cat}(\mathbf{A}, \mathbf{B}) \times \mathbf{A} \to \mathbf{B}$$

and the abstraction functor

$$\Lambda_{\mathbf{A}} : \omega\text{-}\mathbf{Cat}(\mathbf{A} \times \mathbf{B}, \mathbf{C}) \to \omega\text{-}\mathbf{Cat}(\mathbf{A}, \mathbf{IFP}(\mathbf{B}, \mathbf{C}))$$

are both ω*-functors.*

Corollary 4.A.9. *Let* **A**, **B**, *and* **C** *be small* **IFP***-categories. The evaluation functor*

$$\mathsf{ev}_{\mathbf{A},\mathbf{B}} : \mathbf{IFP}(\mathbf{A}, \mathbf{B}) \times \mathbf{A} \to \mathbf{B}$$

and the abstraction functor

$$\Lambda_{\mathbf{A}} : \mathbf{IFP}(\mathbf{A} \times \mathbf{B}, \mathbf{C}) \to \mathbf{IFP}(\mathbf{A}, \mathbf{IFP}(\mathbf{B}, \mathbf{C}))$$

are both ω*-functors.*

Proposition 4.A.10. *The 2-exponential* ω-$\mathbf{Cat}[\mathbf{A} \to \mathbf{B}]$ *of categories* **A** *and* **B** *in* ω-**Cat** *is given by* ω-$\mathbf{Cat}(\mathbf{A}, \mathbf{B})$.

Proof. Let **A**, **B** and **C** be arbitrary small ω-categories. By lemma 4.A.7, ω-$\mathbf{Cat}(\mathbf{B}, \mathbf{C})$ is an ω-**Cat**, and we write ω-$\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]$ for it. We claim that the abstraction functor Λ defines a 2-natural isomorphism

$$\omega\text{-}\mathbf{Cat}(- \times \mathbf{B}, \mathbf{C}) \to \omega\text{-}\mathbf{Cat}(-, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]). \tag{55}$$

A routine check confirms that it defines a family of isomorphisms of categories. To verify naturality, let $F : \mathbf{D} \to \mathbf{A}$ be an arbitrary ω-functor. We must check that the following diagram commutes in **CAT**:

$$\begin{array}{ccc} \omega\text{-}\mathbf{Cat}(\mathbf{A} \times \mathbf{B}, \mathbf{C}) & \xrightarrow{\Lambda_{\mathbf{A}}} & \omega\text{-}\mathbf{Cat}(\mathbf{A}, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]) \\ {\scriptstyle \omega\text{-}\mathbf{Cat}(F \times \mathbf{B}, \mathbf{C})}\downarrow & & \downarrow{\scriptstyle \omega\text{-}\mathbf{Cat}(F, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}])} \\ \omega\text{-}\mathbf{Cat}(\mathbf{D} \times \mathbf{B}, \mathbf{C}) & \xrightarrow{\Lambda_{\mathbf{D}}} & \omega\text{-}\mathbf{Cat}(\mathbf{D}, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]). \end{array} \tag{56}$$

We begin by showing that both paths around the square agree on objects. Let $K : \mathbf{A} \times \mathbf{B} \to \mathbf{C}$ be an arbitrary ω-functor. Going around the top and the right, we get $(\Lambda_{\mathbf{A}} K) \circ F$. Going around the left and bottom, we get $\Lambda_{\mathbf{D}}(K \circ (F \times \mathsf{id}_{\mathbf{B}}))$. We must show that these are equal functors $\mathbf{D} \to \omega$-$\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]$. Let D and B be arbitrary objects in **D** and **B**, respectively, then

$$((\Lambda_{\mathbf{A}} K) \circ F)DB = (\Lambda_{\mathbf{A}} K)(FD)B = K(FD, B) = (K \circ (F \times \mathsf{id}_{\mathbf{B}}))(D, B) = (\Lambda_{\mathbf{D}}(K \circ (F \times \mathsf{id}_{\mathbf{B}})))DB.$$

An analogous check gives that the functors agree on morphisms. So the two functors are equal.

Next, we show that both paths around the square agree on morphisms. Let $\alpha : K \Rightarrow L : \mathbf{A}\times\mathbf{B} \to \mathbf{C}$ be arbitrary in ω-$\mathbf{Cat}$. Going around the top and the right, we get the natural transformation

$$(\omega\text{-}\mathbf{Cat}(F, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]) \circ \Lambda_{\mathbf{A}})(\alpha) \tag{57}$$

$$= \omega\text{-}\mathbf{Cat}(F, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}])(\Lambda_{\mathbf{A}}\alpha : \Lambda_{\mathbf{A}}K \Rightarrow \Lambda_{\mathbf{A}}L : \mathbf{A} \to \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}] \tag{58}$$

$$= (\Lambda_{\mathbf{A}}\alpha)F : (\Lambda_{\mathbf{A}}K) \circ F \Rightarrow (\Lambda_{\mathbf{A}}K) \circ F : \mathbf{D} \to \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]. \tag{59}$$

Going around the left and the bottom, we get the natural transformation

$$(\Lambda_{\mathbf{D}} \circ \omega\text{-}\mathbf{Cat}(F \times \mathbf{B}, \mathbf{C}))(\alpha) \tag{60}$$

$$= \Lambda_{\mathbf{D}}(\alpha(F \times \mathsf{id}_{\mathbf{B}})) : \Lambda_{\mathbf{D}}(K \circ (F \times \mathsf{id}_{\mathbf{B}})) \Rightarrow \Lambda_{\mathbf{D}}(L \circ (F \times \mathsf{id}_{\mathbf{B}})) : \mathbf{D} \to \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]. \tag{61}$$

By the above, they both have equal domains and codomains. To check that they are equal natural transformations, we must show that they have equal components. Let D and B be arbitrary objects of $\mathbf{D}$ and $\mathbf{B}$, respectively. We must show that

$$(((\Lambda_{\mathbf{A}}\alpha)F)_D)_B = ((\Lambda_{\mathbf{D}}(\alpha(F \times \mathsf{id}_{\mathbf{B}})))_D)_B .$$

We compute that

$$(((\Lambda_{\mathbf{A}}\alpha)F)_D)_B = \alpha_{(FD,B)} = \alpha_{(F\times\mathsf{id}_{\mathbf{B}})(D,B)} = ((\Lambda_{\mathbf{D}}(\alpha(F \times \mathsf{id}_{\mathbf{B}})))_D)_B .$$

So we conclude naturality.

We now show that the isomorphism (55) is 2-natural. Let $\rho : F \Rightarrow F' : \mathbf{D} \to \mathbf{A}$ be an arbitrary 2-cell in ω-$\mathbf{Cat}$. We must show that the two following 2-cells (natural transformations) are equal:

$$\omega\text{-}\mathbf{Cat}(\mathbf{A} \times \mathbf{B}, \mathbf{C}) \;\underset{\omega\text{-}\mathbf{Cat}(F'\times\mathbf{B},\mathbf{C})}{\overset{\omega\text{-}\mathbf{Cat}(F\times\mathbf{B},\mathbf{C})}{\Downarrow \omega\text{-}\mathbf{Cat}(\rho \times \mathbf{B}, \mathbf{C})}}\; \omega\text{-}\mathbf{Cat}(\mathbf{D} \times \mathbf{B}, \mathbf{C}) \xrightarrow{\Lambda_{\mathbf{D}}} \omega\text{-}\mathbf{Cat}(\mathbf{D}, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]),$$

$$\omega\text{-}\mathbf{Cat}(\mathbf{A} \times \mathbf{B}, \mathbf{C}) \xrightarrow{\Lambda_A} \omega\text{-}\mathbf{Cat}(\mathbf{A}, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]) \;\underset{\omega\text{-}\mathbf{Cat}(F',\omega\text{-}\mathbf{Cat}[\mathbf{B}\to\mathbf{C}])}{\overset{\omega\text{-}\mathbf{Cat}(F,\omega\text{-}\mathbf{Cat}[\mathbf{B}\to\mathbf{C}])}{\Downarrow \omega\text{-}\mathbf{Cat}(\rho, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}])}}\; \omega\text{-}\mathbf{Cat}(\mathbf{D}, \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}]).$$

Consider an arbitrary component $G : \mathbf{A} \times \mathbf{B} \to \mathbf{B}$. The G-component of the top natural transformation is

$$(\Lambda_{\mathbf{D}} * \omega\text{-}\mathbf{Cat}(\rho \times \mathbf{B}, \mathbf{C}))_G$$

$$= \Lambda_{\mathbf{D}}(G * (\rho \times \mathsf{id}_{\mathbf{B}})) : \Lambda_{\mathbf{D}}(G(F \times \mathsf{id}_{\mathbf{B}})) \Rightarrow \Lambda_{\mathbf{D}}(G(F' \times \mathsf{id}_{\mathbf{B}})) : \mathbf{D} \to \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}].$$

Let D be an arbitrary object in $\mathbf{D}$, then the D-component is:

$$(\Lambda_{\mathbf{D}}(G * (\rho \times \mathsf{id}_{\mathbf{B}})))_D : \Lambda_{\mathbf{D}}(G(F \times \mathsf{id}_{\mathbf{B}}))D \Rightarrow \Lambda_{\mathbf{D}}(G(F' \times \mathsf{id}_{\mathbf{B}}))D$$

$$= G(\rho_D, \mathsf{id}_{\mathbf{B}}) : G(FD, \mathsf{id}_{\mathbf{B}}) \Rightarrow G(F'D, \mathsf{id}_{\mathbf{B}})$$

$$= ((\Lambda_{\mathbf{A}}G) * \rho)_D : (\Lambda_{\mathbf{A}}G)F \Rightarrow (\Lambda_{\mathbf{A}}G)F' : \mathbf{D} \to \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}],$$

which is we recognize as the G, D-component of the bottom 2-cell. Because the G and D were arbitrary components, we conclude that the two 2-cells are equal, and we conclude 2-naturality. □

Corollary 4.A.11. *The 2-exponential* $\mathbf{IFP}[\mathbf{A} \to \mathbf{B}]$ *of categories* $\mathbf{A}$ *and* $\mathbf{B}$ *in* $\mathbf{IFP}$ *is given by* $\mathbf{IFP}(\mathbf{A}, \mathbf{B})$.

Proof. Immediate by lemma 4.A.7, corollary 4.A.9, and proposition 4.A.10. □

Corollary 4.A.12. *The category* ω-$\mathbf{Cat}$ *is 2-cartesian closed.*

Proof. Immediate by propositions 4.A.1, 4.A.4 and 4.A.10. □

Corollary 4.A.13. *The category* $\mathbf{IFP}$ *is 2-cartesian closed.*

Proof. Immediate by proposition 4.A.1 and corollaries 4.A.5 and 4.A.11. □

Proposition 4.A.14 ([Leh76a, Lemma VI.4]). *The composition functor* $\circ : \omega\text{-}\mathbf{Cat}[\mathbf{B} \to \mathbf{C}] \times \omega\text{-}\mathbf{Cat}[\mathbf{A} \to \mathbf{B}] \to \omega\text{-}\mathbf{Cat}[\mathbf{A} \to \mathbf{C}]$ *is an* ω*-functor.*

Lemma 4.A.15. *If* $J : \omega \to \omega$ *has an colimit, then there exists a least* $n \in \mathbb{N}$ *such that for all* $k \geq n$, $Jk = Jn$. *The colimit of* J *is the cocone* (j, Jn) *where* $j_k = J(k \to n) : Jk \to Jn$ *for* $k < n$ *and* $j_k = \mathsf{id}_{Jn}$ *for* $k \geq n$.

Proof. A functor $J : \omega \to \omega$ is a monotone map on the poset ω, and the colimit of J is the least upper bound of its image. Every set of integers bounded above has a maximum element, and this maximum element is its least upper bound. So $\bigsqcup J$ is in the image of J. By the well-ordering principle, the J-preimage of $\bigsqcup J$ has a least element n. Then for all $k \geq n$, we have $Jn \leq Jk \leq \bigsqcup J = J_n$. It follows that $Jk = Jn$ for all $k \geq n$.

The characterization of the cocone j in the statement follows readily from the definition of ω as a poset. □

Proposition 4.A.16. *Every functor* $F : \omega \to \mathbf{K}$ *preserves* ω*-colimits.*

Proof. Let $J : \omega \to \omega$ be arbitrary and assume it has a colimit. By lemma 4.A.15, it is of the form (j, Jn) for the least n such that $Jk = Jn$ for all $k \geq n$. We must show that (Fj, FJn) is an ω-colimit of $FJ : \omega \to \mathbf{K}$.

Let (α, A) be any other cocone on FJ. We must show that there exists a unique cocone morphism $a : (Fj, FJn) \to (\alpha, A)$. In particular, we must show that there exists a unique morphism $a : FJn \to A$ in $\mathbf{K}$ such that for all k, $\alpha_k = a \circ Fj_k : FJk \to A$.

We begin with existence. Set $a = \alpha_n$. Then for all $k < n$,

$$\alpha_k = \alpha_n \circ FJ(k \to n) = \alpha_n \circ Fj_k = a \circ Fj_k.$$

For all $k \geq n$, observe that

$$\alpha_k = \alpha_k \circ \mathsf{id}_{FJk} = \alpha_k \circ \mathsf{id}_{FJn} = \alpha_k \circ FJ(n \to k) = \alpha_n = a,$$

so

$$\alpha_k = a = a \circ \mathsf{id}_{FJn} = a \circ F\mathsf{id}_{Jn} = a \circ Fj_k.$$

This establishes that $a = \alpha_n$ is a cocone morphism.

Next, we show uniqueness. Let $b : (Fj, FJn) \to (\alpha, A)$ be any other cocone morphism. Then for all k, $\alpha_k = b \circ Fj_k : FJk \to A$. In particular,

$$\alpha_n = b \circ Fj_n = b \circ F\mathsf{id}_{Jn} = b \circ \mathsf{id}_{FJn} = b.$$

This uniquely characterizes the morphism and establishes uniqueness.

We conclude that (Fj, FJn) is the ω-colimit of $FJ : \omega \to \mathbf{K}$. Because J was arbitrary, we conclude that F preserves ω-colimits. □

Part 2

Polarized SILL

CHAPTER 5

Statics and Dynamics

The Polarized SILL programming language [TCP13; PG15] cohesively integrates functional computation and message-passing concurrent computation. Its concurrent computation layer arises from a proofs-as-processes correspondence between intuitionistic linear logic and the session-typed π-calculus [CP10]. We give an overview of its statics and dynamics in sections 5.1 and 5.2 before presenting the language in section 5.3. In section 5.7, we describe general properties of relations on its programs.

5.1. Overview of Statics

Processes are computational agents that interact with their environment solely through communication. In Polarized SILL, communication happens over named channels, which we can intuitively think of as wires that carry messages. Moreover, communication on channels is bidirectional: in general, a process can both send and receive communications along the same channel. Each channel has an associated session type A. Session types [THK94; Hon93] specify communication protocols, i.e., rules for communicating along channels. Equivalently, we can think of session types as classifying communications, analogously to how data types classify values. A channel's session type then specifies which communications are permitted on that channel. The type system for processes ensures that communication on a channel of type A respects the protocol specified by the session type A.

Processes in Polarized SILL are organized according to a client-server architecture, and we can think of session types as describing services provided or used along channels. A process P always **provides** or is a server for a service A on a channel c, and it **uses** or is a client of zero or more services A_i on channels c_i. We write $c : A$ to mean that the channel c has type A. The used services form a linear context $\Delta = c_1 : A_1, \ldots, c_n : A_n$. The process P can use values from the functional layer. These are abstracted by a structural context Ψ of functional variables. These data are captured by the inductively defined judgment $\Psi \,;\, \Delta \vdash P :: c : A$. We say that the process P is **closed** if it does not depend on any free variables, i.e., if $\cdot \,;\, \Delta \vdash P :: c : A$. This judgment is both generic and parametric (see section 2.5): it is closed under renaming of channel names in $\Delta, c : A$, and it is closed under renaming and substitution of functional variables in Ψ.

At any given point in a computation, communication flows in a single direction on a channel $c : A$. The direction of communication is determined by the *polarity* of the type A, where session types are partitioned as **positive** or **negative** [PG15]. Consider a process judgment $\Psi \,;\, \Delta \vdash P :: c_o : A_o$. Communication on positively typed channels flows from left-to-right in this judgment: if A_o is positive, then P can only send output on c_o, while if A_i is positive for $1 \le i \le n$, then P can only receive input on c_i. Symmetrically, communication on negatively typed channels flows from right-to-left in the judgment. Bidirectional communication arises from the fact that the type of a channel evolves over the course of a computation, sometimes becoming positive, sometimes becoming negative. We write $B\ \mathsf{type}_s^+$ to mean B is positive and $B\ \mathsf{type}_s^-$ to mean B is negative. Most session types have a polar-dual session type, where the direction of the communication is reversed.

Open session types are given by the inductively defined judgment $\Xi \vdash A\ \mathsf{type}_s^p$, where Ξ is a structural context of polarized type variables $\alpha_i\ \mathsf{type}_s^{p_i}$ and $p, p_i \in \{-, +\}$. We abbreviate the judgment as $\Xi \vdash A\ \mathsf{type}_s$ when the polarity is unambiguous. The session type A is closed if it does

$$\frac{}{\lambda x : \tau.M \Downarrow \lambda x : \tau.M}\ (\text{EV-Fun}) \qquad \frac{M \Downarrow \lambda x : \tau.M' \quad N \Downarrow w \quad [w/x]M' \Downarrow v}{MN \Downarrow v}\ (\text{EV-App})$$

$$\frac{}{c_o \leftarrow \{P\} \leftarrow \overline{c_i} \Downarrow c_o \leftarrow \{P\} \leftarrow \overline{c_i}}\ (\text{EV-Proc}) \qquad \frac{[\mathsf{fix}\, x.M/x]M \Downarrow v}{\mathsf{fix}\, x.M \Downarrow v}\ (\text{EV-Fix})$$

FIGURE 5.1. Big-step semantics underlying Polarized SILL's functional layer

not depend on any free variables, i.e., if $\cdot \vdash A\ \mathsf{type_s}$. We explicitly treat the inductive definition of open session types for two reasons. First, it is useful for expository purposes because it lets us make polarities explicit. Second, the denotation of a session type $\Xi \vdash A\ \mathsf{type_s}$ is recursively defined on its derivation, and an explicit definition of this judgment simplifies the definition of its denotations.

The functional layer is the simply-typed λ-calculus with a fixed-point operator and a call-by-value evaluation semantics. A judgment $\Psi \Vdash M : \tau$ means the functional term M has functional type τ under the structural context Ψ of functional variables $x_i : \tau_i$. This judgment's inductive definition is standard. We say that the term M is closed if it does not depend on any free variables, i.e., if $\cdot \Vdash M : \tau$. We use the judgment $\Xi \vdash \tau\ \mathsf{type_f}$?? to mean that τ is a functional type depending on polarized type variables $\alpha_i\ \mathsf{type_s}^{p_i}$. The type τ is closed if it does not depend on any free variables. New is the base type $\{a : A \leftarrow \overline{a_i : A_i}\}$ of quoted processes, where we abbreviate ordered lists using an overline.

We draw attention to the fundamental difference between *variables* and *channel names*. A functional variable $x : \tau$ in a context Ψ stands for a value of type τ. A channel name in $\Delta, c : A$ is a symbol: it stands not for a value, but for a channel of typed bidirectional communications. In particular, channel names can only be renamed; unlike functional variables, nothing can be substituted for a channel name.

5.2. Overview of Dynamics

The operational behaviour of processes in Polarized SILL is defined by a substructural operational semantics [Sim12] in the form of a multiset rewriting system. This multiset rewriting system uses three different kinds of facts. The two most commonly encountered facts involve processes and messages. The fact $\mathsf{proc}(c, P)$ means that the closed process P provides a channel c. The fact $\mathsf{msg}(c, m)$ means that the message process m provides a channel c. Message processes represent single messages or pieces of data sent on a channel, and they are closed processes written in a restricted fragment of the process language. Process communication is asynchronous: processes send messages without synchronizing with recipients. Messages sent on a given channel are received in order. However, there is no global ordering on sent messages: messages sent on different channels can be received out of order.

The behaviour of the functional layer is specified by the set $\mathbf{F}$ of persistent facts $\mathbf{eval}(M, v)$, where $\mathbf{eval}(M, v)$ if and only if the closed term M evaluates to the value v under the standard call-by-value semantics. Explicitly, $\mathbf{eval}(M, v)$ if and only if $M \Downarrow v$, where $M \Downarrow v$ is the usual evaluation semantics. It is inductively defined in fig. 5.1. We write $v\ \mathsf{val}$ if v is a value, i.e., if $v \Downarrow v$. The fact $\mathbf{eval}(M, v)$ captures an evaluation relation instead of a transition relation because we only ever observe the process layer, and we never need to observe individual steps in the functional layer. For conciseness, we do not mention this set of facts in our multisets and instead treat it implicitly.

For consistency with the literature, we call a multiset-in-context in a process trace a **configuration**. A **process trace** is a trace from the initial configuration of a process. The **initial configuration** of $\cdot\ ;\ c_1 : A_1, \ldots, c_n : A_n \vdash P :: c_o : A_o$ is the multiset-in-context

$$c_o, \ldots, c_n\ ;\ \mathsf{proc}(c_o, P),$$

where the multiset $\mathbf{F}$ of $\mathbf{eval}(M, v)$ facts is implicitly present. A **fair execution** of $\cdot\,;\,\Delta \vdash P :: c : A$ is a weakly fair execution from its initial configuration. By propositions 3.3.10 and 5.9.9, all weakly fair executions are also strongly fair and über fair.

The substructural operational semantics maintains several invariants that we consider in detail in section 5.9. Chief among these is a preservation-style property (proposition 5.9.1), where each configuration appearing in a trace is well-typed, and where each multiset rewrite rule preserves the type of the configuration. Concretely, we introduce a type system for configurations inspired by one due to Gommerstadt, Jia, and Pfenning [GJP18, § 4.4]. It assigns a session type to each free channel appearing in a configuration. The preservation property states that the types of channels not free in the active multiset of a rule remain unchanged in the result, and that the types of external channels remain unchanged. This approach is in contrast to the one taken by Kavanagh [Kav20a], which conservatively extended the underlying substructural operational semantics to track typing information at runtime. Advantageously, our approach preserves the distinction between operational rules and typing concerns, and it requires no changes to the original substructural operational semantics.

The typing judgment $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ means that the configuration $\Sigma\,;\mathcal{C}$ *uses* the channels in Γ, *provides* the channels in Δ, and has internal channels I. Here, $\Gamma = \gamma_1 : A_1, \ldots, \gamma_n : A_n$, $\Delta = \delta_1 : B_1, \ldots, \delta_m : B_m$, and $\mathrm{I} = \iota_1 : C_1, \ldots, \iota_k : C_k$ are linear contexts of session-typed channel names, with $n, k \geq 0$ and $m \geq 1$. Write $\check{\Gamma}$ for the list $\gamma_1, \ldots, \gamma_n$ of channel names appearing in Γ. The judgment $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ is well-formed only if the channel names in $\check{\Gamma}, \check{\Delta}, \check{\mathrm{I}}$ are pairwise distinct and $\check{\Gamma}, \check{\Delta}, \check{\mathrm{I}} \subseteq \Sigma$. This judgment is parametric in Σ, i.e., it enjoys the "proliferation" and "renaming" structural properties for channel names in Σ (see section 2.5.6 for details). For brevity, we usually leave Σ implicit and write $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ for $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$. We also often write $\Gamma \vdash \mathcal{C} :: \Delta$ if $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ for some I. We call the pair (Γ, Δ) the **interface** of $\mathcal{C}$. In contrast to processes, configurations can provide multiple channels. This is to allow for applications like run-time monitoring [GJP18]. We remark that the multiset $\mathbf{F}$ of $\mathbf{eval}(M, v)$ facts is implicitly contained in $\mathcal{C}$ in every judgment $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$.

The above judgment is inductively defined by the rules (Conf-M), (Conf-P), and (Conf-C):

$$\frac{\cdot\,;\,\Delta \vdash m :: c : A}{\Sigma \parallel \Delta \mid \cdot \vdash \mathsf{msg}(c, m) :: (c : A)}\ (\textsc{Conf-M}) \qquad \frac{\cdot\,;\,\Delta \vdash P :: c : A}{\Sigma \parallel \Delta \mid \cdot \vdash \mathsf{proc}(c, P) :: (c : A)}\ (\textsc{Conf-P})$$

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \qquad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\textsc{Conf-C})$$

The rules (Conf-M) and (Conf-P) lift closed messages and processes to message and process facts, while preserving their used and provided channels. In (Conf-M), we assume that m ranges over "message processes" m^+ and $m^-_{b,c}$. These message processes are a restricted class of processes defined in eqs. (62) and (63). The composition rule (Conf-C) is a "parallel composition plus hiding" operation (cf. [Mil80, pp. 20f.]). It composes two configurations $\mathcal{C}$ and $\mathcal{D}$ so that they communicate along some common (but potentially empty) collection of channels Π. These channels are then hidden from external view: they do not appear in the interface $(\Gamma\Lambda, \Phi\Xi)$, but instead appear in the composition's context $\mathrm{I}_1 \Pi \mathrm{I}_2$ of internal channels. Without loss of generality, we assume that $\Sigma \cap \Sigma' = \varnothing$ (so $\check{I}_1 \cap \check{I}_2 = \varnothing$) when composing $\mathcal{C}$ and $\mathcal{D}$. This requirement ensures that the internal channels in $\mathcal{C}$ do not interfere with those in $\mathcal{D}$, and vice-versa. Because the hypotheses are parametric in Σ and Σ', we can always rename those channels to ensure that this is the case.

We can recognize (Conf-C) as a composition operator in a pluricategory.[1] Indeed, consider the pluricategory whose objects are session-typed channels, and think of a typed configuration $\Gamma \vdash \mathcal{C} :: \Delta$ as a morphism $\Gamma \to \Delta$. Then the rule (Conf-C) is a special case of the corresponding

[1] Pluricategories were defined in definition 2.1.11.

pluricategorical composition rule

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi\Psi \quad \check{\Pi}, \Sigma' \parallel \Delta\Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Delta\Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi\Psi}$$

where Ψ and Δ are both empty. The rule (Conf-C) determines an associative and partially commutative partial composition operator. We study these properties and others in section 5.9.

5.3. Typing and Multiset-Rewriting Rules

In this section, we give the typing rules that inductively define well-typed functional terms and processes. In each case, we also give the associated multiset rewriting rules. With a few exceptions and for brevity, we only give the rules for processes that provide positive session types. All rules can be found in section 5.B.

5.3.1. Manipulating Channels. The forwarding process $b \to a$ forwards all messages between channels a and b of the same positive type. The process $b \leftarrow a$ is the dual for channels of negative type. We remark that the syntax reflects the direction in which messages flow. Though some presentations use a single forwarding process for both polarities, it is useful for practical and semantic concerns to syntactically differentiate between forwarding positive communications and forwarding negative communications.

$$\frac{\cdot \vdash A\ \mathsf{type}_s^+}{\Psi\ ;\ a : A \vdash a \to b :: b : A}\ (\textsc{Fwd}^+) \qquad \frac{\cdot \vdash A\ \mathsf{type}_s^-}{\Psi\ ;\ a : A \vdash a \leftarrow b :: b : A}\ (\textsc{Fwd}^-)$$

These processes act on messages m^+ and $m^-_{b,c}$ travelling in the positive and negative directions. These messages are respectively **message processes** given by eqs. (62) and (63). Their meaning will be explained below. The letters a, b, c, and d range over channel names, l ranges over labels, and v ranges over functional values.

$$\begin{aligned} m^+ ::= {} & \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a \mid \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a \mid a.l;\ d \to a \mid \mathsf{send}\ a\ b;\ d \to a \\ & \mid \mathsf{send}\ a\ \mathsf{unfold};\ d \to a \mid \mathsf{close}\ a \end{aligned} \tag{62}$$

$$\begin{aligned} m^-_{b,c} ::= {} & \_ \leftarrow \mathsf{output}\ b\ v;\ b \leftarrow c \mid \mathsf{send}\ b\ \mathsf{shift};\ b \to c \mid a.l;\ a \leftarrow c \mid \mathsf{send}\ a\ b;\ a \leftarrow c \\ & \mid \mathsf{send}\ a\ \mathsf{unfold};\ a \leftarrow c \end{aligned} \tag{63}$$

The subscripts on $m^-_{b,c}$ serve to indicate which channel names appear in the message fact, and they ensure that rule (65) is only applicable when the process fact and the message fact have a common channel. The operational behaviour is given by rules (64) and (65). Properly speaking, there is an instance of these rules for each different kind of message m^+ and $m^-_{b,c}$. We implicitly universally quantify on the channel names appearing in these multiset rewriting rules:

$$\mathsf{msg}(a, m^+), \mathsf{proc}(b, a \to b) \to \mathsf{msg}(b, [b/a]m^+) \tag{64}$$

$$\mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m^-_{b,c}) \to \mathsf{msg}(c, [a/b]m^-_{b,c}) \tag{65}$$

Process composition $a \leftarrow P;\ Q$ captures Milner's "parallel composition plus hiding" operation [Mil80, pp. 20f.]. It spawns processes P and Q that communicate over a shared private channel a of type A. Those familiar with the π-calculus may like to think of this syntax as analogous to the π-calculus process $(\nu a)(P \mid Q)$.

$$\frac{\Psi\ ;\ \Delta_1 \vdash P :: a : A \quad \Psi\ ;\ a : A, \Delta_2 \vdash Q :: c : C}{\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C}\ (\textsc{Cut})$$

$$\forall \Delta_1, \Delta_2, c.\mathsf{proc}(c, a \leftarrow P;\ Q) \to \exists b.\mathsf{proc}(b, [b/a]P), \mathsf{proc}(c, [b/a]Q) \tag{66}$$

We remark that in (66) we do not quantify over the channel name a. This is because the channel name a is a bound in processes P and Q, and processes denote *general binding trees* (see section 2.4) whose bound names can freely be varied.

Processes can close channels of type $\mathbf{1}$. To do so, the process close a sends a "close message" over the channel a and terminates. The process wait a; P blocks on a until it receives the close message and then continues as P.

$$\frac{}{\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+}\ (\mathrm{C}\mathbf{1}) \qquad \frac{}{\Psi\ ;\ \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}}\ (\mathbf{1}\mathrm{R}) \qquad \frac{\Psi\ ;\ \Delta \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C}\ (\mathbf{1}\mathrm{L})$$

$$\forall \Delta, a, c.\mathsf{msg}(a, \mathsf{close}\ a), \mathsf{proc}(c, \mathsf{wait}\ a;\ P) \to \mathsf{proc}(c, P) \tag{67}$$

$$\forall a.\mathsf{proc}(a, \mathsf{close}\ a) \to \mathsf{msg}(a, \mathsf{close}\ a) \tag{68}$$

The positive type $\mathbf{1}$ does not have a negative dual. It would require a detached process with no client.

Example 5.3.1. The following process is, informally speaking, equivalent to the forwarding process $\cdot\ ;\ a : \mathbf{1} \vdash a \to b :: b : \mathbf{1}$ for channels of type $\mathbf{1}$:

$$\cdot\ ;\ a : \mathbf{1} \vdash \mathsf{wait}\ a;\ \mathsf{close}\ b :: b : \mathbf{1}.$$

Indeed, if no close message arrives on a, then neither process does anything. If a close message arrives on a, then both processes send a close message on b and terminate:

$$\mathsf{msg}(a, \mathsf{close}\ a), \mathsf{proc}(b, a \to b) \to \mathsf{msg}(b, \mathsf{close}\ b)$$

$$\mathsf{msg}(a, \mathsf{close}\ a), \mathsf{proc}(b, \mathsf{wait}\ a;\ \mathsf{close}\ b) \to^* \mathsf{msg}(b, \mathsf{close}\ b)$$

We will revisit this example when we discuss the computational interpretation of the identity expansion theorem of intuitionistic linear logic in section 9.4. ◀

Processes can send and receive channels over channels. The protocol $B \otimes A$ prescribes transmitting a channel of type B followed by communication of type A. The process send $a\ b$; P sends the channel b over the channel a and continues as P. The process $b \leftarrow \mathsf{recv}\ a$; P receives a channel over a, binds it to the name b, and continues as P.

$$\frac{\Xi \vdash A\ \mathsf{type}_s^+ \quad \Xi \vdash B\ \mathsf{type}_s^+}{\Xi \vdash A \otimes B\ \mathsf{type}_s^+}\ (\mathrm{C}\otimes)$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A}\ (\otimes\mathrm{R}) \qquad \frac{\Psi\ ;\ \Delta, a : A, b : B \vdash P :: c : C}{\Psi\ ;\ \Delta, a : B \otimes A \vdash b \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\otimes\mathrm{L})$$

$$\forall \Delta, b, a.\mathsf{proc}(a, \mathsf{send}\ a\ b;\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a) \tag{69}$$

$$\forall a, e, d, \Delta, c.\mathsf{msg}(a, \mathsf{send}\ a\ e;\ d \to a), \mathsf{proc}(c, b \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [e, d/b, a]P) \tag{70}$$

Example 5.3.2. The following closed process P sends a channel $a : \mathbf{1}$ over $b : \mathbf{1} \otimes \mathbf{1}$, and closes b:

$$\cdot\ ;\ a : \mathbf{1} \vdash \mathsf{send}\ b\ a;\ \mathsf{close}\ b :: b : \mathbf{1} \otimes \mathbf{1}$$

The following closed process Q receives a channel of type $\mathbf{1}$ on b and binds it to the name d. Then it receives a close message on b and forwards d over c:

$$\cdot\ ;\ b : \mathbf{1} \otimes \mathbf{1} \vdash d \leftarrow \mathsf{recv}\ b;\ \mathsf{wait}\ b;\ d \to c :: c : \mathbf{1}.$$

Their composition

$$\cdot\ ;\ a : \mathbf{1} \vdash b \leftarrow (\mathsf{send}\ b\ a;\ \mathsf{close}\ b)\ ;\ (d \leftarrow \mathsf{recv}\ b;\ \mathsf{wait}\ b;\ d \to c) :: c : \mathbf{1}$$

spawns P and Q and eventually forwards the channel a over c:

$$\begin{aligned}
&\mathsf{proc}(c, b \leftarrow P;\ Q) \\
&\to \mathsf{proc}(b_1, \mathsf{send}\ b_1\ a;\ \mathsf{close}\ b_1), \mathsf{proc}(c, d \leftarrow \mathsf{recv}\ b_1;\ \mathsf{wait}\ b_1;\ d \to c) \\
&\to^* \mathsf{proc}(b_2, \mathsf{close}\ b_2), \mathsf{proc}(c, \mathsf{wait}\ b_2;\ a \to c) \\
&\to^* \mathsf{proc}(c, a \to c).
\end{aligned}$$

◀

The negative dual $B \multimap A$ of $B \otimes A$ is subtle: for linear-logical reasons, the polarities of A and B differ. Indeed, though $B \multimap A$ and A are negative, B must be positive:

$$\frac{\Xi \vdash B\ \mathsf{type}_\mathsf{s}^+ \quad \Xi \vdash A\ \mathsf{type}_\mathsf{s}^-}{\Xi \vdash B \multimap A\ \mathsf{type}_\mathsf{s}^-}\ (\mathrm{C}{\multimap})$$

$$\frac{\Psi\ ;\ \Delta, b : B \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A}\ (\multimap\mathrm{R}) \quad \frac{\Psi\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, b : B, a : B \multimap A \vdash \mathsf{send}\ a\ b;\ P :: c : C}\ (\multimap\mathrm{L})$$

$$\forall a, e, d, \Delta, c.\mathsf{proc}(a, b \leftarrow \mathsf{recv}\ a;\ P), \mathsf{msg}(d, \mathsf{send}\ a\ e;\ a \leftarrow d) \rightarrow \mathsf{proc}(d, [e, d/b, a]P) \quad (71)$$

$$\forall \Delta, b, a, c.\mathsf{proc}(c, \mathsf{send}\ a\ b;\ P) \rightarrow \exists d.\mathsf{msg}(d, \mathsf{send}\ a\ b;\ a \leftarrow d), \mathsf{proc}(c, [d/a]P) \quad (72)$$

5.3.2. Functional Programming and Value Transmission. The only base types in the functional layer are the types $\{a_0 : A_0 \leftarrow a_1 : A_1, \ldots, a_n : A_n\}$ of quoted processes.[2] These types are formed by the rule (T{}). The functional layer also supports function types $\tau \rightarrow \sigma$. These are formed by the rule (T→). We assume that types are closed whenever they appear in a typing judgment for terms or processes.

$$\frac{\Xi \vdash A_i\ \mathsf{type}_\mathsf{s} \quad (0 \le i \le n)}{\Xi \vdash \{a_0 : A_0 \leftarrow a_1 : A_1, \ldots, a_n : A_n\}\ \mathsf{type}_\mathsf{f}}\ (\mathrm{T}\{\}) \quad \frac{\Xi \vdash \tau\ \mathsf{type}_\mathsf{f} \quad \Xi \vdash \sigma\ \mathsf{type}_\mathsf{f}}{\Xi \vdash \tau \rightarrow \sigma\ \mathsf{type}_\mathsf{f}}\ (\mathrm{T}{\rightarrow})$$

Most of the introduction and elimination rules for functional terms are standard. New is the introduction rule (I-{}) for quoted processes. It encapsulates a process P as a value $a \leftarrow \{P\} \leftarrow \overline{a_i}$ of quoted process type, where we abbreviate ordered lists using an overline. Again, functional terms are not associated with any multiset rewriting rules: their operational behaviour is captured by the relation $\mathbf{eval}(M, v)$.

$$\frac{}{\Psi, x : \tau \Vdash x : \tau}\ (\text{F-Var}) \quad \frac{\Psi, x : \tau \Vdash M : \tau}{\Psi \Vdash \mathsf{fix}\ x.M : \tau}\ (\text{F-Fix})$$

$$\frac{\Psi, x : \tau \Vdash M : \sigma}{\Psi \Vdash \lambda x : \tau.M : \tau \rightarrow \sigma}\ (\text{F-Fun}) \quad \frac{\Psi \Vdash M : \tau \rightarrow \sigma \quad \Psi \Vdash N : \tau}{\Psi \Vdash MN : \sigma}\ (\text{F-App})$$

$$\frac{\Psi\ ;\ \overline{a_i : A_i} \vdash P :: a : A}{\Psi \Vdash a \leftarrow \{P\} \leftarrow \overline{a_i} : \{a : A \leftarrow \overline{a_i : A_i}\}}\ (\mathrm{I}\{\})$$

The elimination form for quoted process terms M is the process $a \leftarrow \{M\} \leftarrow \overline{a_i}$. Operationally, this form evaluates the quoted process term M to a value v, and then spawns the associated quoted process. By the canonical forms lemma (proposition 5.8.2), v will always be a value of the form $a \leftarrow \{P\} \leftarrow \overline{a_i}$, i.e., a quoted process.

$$\frac{\Psi \Vdash M : \{a : A \leftarrow \overline{a_i : A_i}\}}{\Psi\ ;\ \overline{a_i : A_i} \vdash a \leftarrow \{M\} \leftarrow \overline{a}_i :: a : A}\ (\mathrm{E}\{\})$$

$$\forall a, \overline{a}_i.\mathbf{eval}(M, a \leftarrow \{P\} \leftarrow \overline{a_i}), \mathsf{proc}(a, a \leftarrow \{M\} \leftarrow \overline{a_i}) \rightarrow \mathsf{proc}(a, P) \quad (73)$$

The elimination rule (E-{}) differs from the original elimination rule given by Toninho, Caires, and Pfenning [TCP13, p. 354]. There, the elimination rule was a monadic bind similar to the (Cut) rule below, where a quoted process could only be unquoted if it was composed with a continuation process. Though the original rule has the advantage of enforcing a monadic discipline on the interaction between the functional and process layers, it complicates writing and reasoning about recursive processes. Indeed, one cannot directly make a recursive tail call, but must instead always compose the recursive call with a continuation process. We lose nothing by not requiring a continuation process: the original rule is a derived rule in our setting, and our rule can be defined as syntactic sugar in the original version of SILL.

[2] Polarized SILL can straightforwardly be extended to support other base types. We extend it with natural numbers in chapter 8.

Example 5.3.3. Recursive processes are implemented using the functional layer's fixed point operator. An important recursive process is the **divergent process** $\Psi\ ;\ \Delta \vdash \Omega :: c : A$, which exists for all Ψ and $\Delta, c : A$. Let Ω' be the term given by $\Psi \Vdash \mathsf{fix}\ \omega.c \leftarrow \{c \leftarrow \{\omega\} \leftarrow \check{\Delta}\} \leftarrow \check{\Delta} : \{c : A \leftarrow \Delta\}$. Observe that $\mathbf{eval}(\Omega', c \leftarrow \{c \leftarrow \{\Omega'\} \leftarrow \check{\Delta}\} \leftarrow \check{\Delta})$. The process Ω is given by:

$$\Psi\ ;\ \Delta \vdash c \leftarrow \{\Omega'\} \leftarrow \check{\Delta} :: c : A.$$

By rule (73), $\mathsf{proc}(c, \Omega) \to \mathsf{proc}(c, \Omega)$. ◂

Remark 5.3.4. Example 5.3.3 illustrates the subtle interplay between recursion and linearity. Though the linearity of channel contexts ensures that no channels are discarded, linearity cannot guarantee that all channels are used in the presence of recursion. Indeed, the divergent process Ω communicates on none of its channels.

Functional values can be sent over channels of type $\tau \wedge A$. This positive protocol, formed by the rule (C$\wedge$), specifies that the sent value has type τ and that subsequent communication has type A. The process $\_ \leftarrow \mathsf{output}\ a\ M;\ P$ evaluates the term M to a value v, sends v over the channel a, and continues as P. The process $x \leftarrow \mathsf{input}\ a;\ Q$ receives a value v on a, binds it to the variable x, and continues as Q. These behaviours are captured by multiset rewrite rules (74) and (75). In these rules, we use Δ as a shorthand for the list of channel names that appears in the context Δ. To ensure a queue-like structure for messages on a, we generate a fresh channel name d for the "continuation channel" that will carry subsequent communications. Operationally, we rename a in P to the continuation channel d carrying the remainder of the communications.

$$\frac{\Xi \vdash \tau\ \mathsf{type}_\mathsf{f} \quad \Xi \vdash A\ \mathsf{type}^+_\mathsf{s}}{\Xi \vdash \tau \wedge A\ \mathsf{type}^+_\mathsf{s}}\ (\mathrm{C}\wedge)$$

$$\frac{\Psi \Vdash M : \tau \quad \Psi\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A}\ (\wedge\mathrm{R}) \qquad \frac{\Psi, x : \tau\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ P :: c : C}\ (\wedge\mathrm{L})$$

$$\forall a, \Delta.\mathbf{eval}(M, v), \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) \to \\ \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a) \tag{74}$$

$$\forall \Delta, a, d, c.\mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a), \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) \to \\ \to \mathsf{proc}(c, [d, v/a, x]P) \tag{75}$$

Example 5.3.5. The following process sends values $v_1 : \tau_1$, $v_2 : \tau_2$, and $v_3 : \tau_3$ on the channel a, before divergently providing a service of type A:

$$\cdot\ ;\ \cdot \vdash \_ \leftarrow \mathsf{output}\ a\ v_1;\ \_ \leftarrow \mathsf{output}\ a\ v_2;\ \_ \leftarrow \mathsf{output}\ a\ v_3;\ \Omega :: a : \tau_1 \wedge (\tau_2 \wedge (\tau_3 \wedge A)).$$

The queue-like structure of message processes $\_ \leftarrow \mathsf{output}\ a\ v;\ d \to a$ is analogous to the queues to example 3.1.6. Combined with rule (75), it ensures that messages are received in order. Indeed, consider the following trace of the above process:

$$\begin{aligned}
&\mathsf{proc}(a, _ \leftarrow \mathsf{output}\ a\ v_1;\ _ \leftarrow \mathsf{output}\ a\ v_2;\ _ \leftarrow \mathsf{output}\ a\ v_3;\ \Omega) \\
&\to \mathsf{proc}(b, _ \leftarrow \mathsf{output}\ a\ v_2;\ _ \leftarrow \mathsf{output}\ a\ v_3;\ \Omega), \mathsf{msg}(a, _ \leftarrow \mathsf{output}\ a\ v_1;\ b \to a) \\
&\to (\mathsf{proc}(c, _ \leftarrow \mathsf{output}\ c\ v_3;\ \Omega), \mathsf{msg}(b, _ \leftarrow \mathsf{output}\ b\ v_2;\ c \to b), \\
&\qquad\qquad \mathsf{msg}(a, _ \leftarrow \mathsf{output}\ a\ v_1;\ b \to a)) \\
&\to (\mathsf{proc}(d, \Omega), \mathsf{msg}(c, _ \leftarrow \mathsf{output}\ c\ v_3;\ d \to c), \mathsf{msg}(b, _ \leftarrow \mathsf{output}\ b\ v_2;\ c \to b), \\
&\qquad\qquad \mathsf{msg}(a, _ \leftarrow \mathsf{output}\ a\ v_1;\ b \to a))\,.
\end{aligned}$$

Though multisets do not impose an order on their elements, any process P using the channel a will receive the values v_1, v_2, and v_3 in that order. This is because P cannot know the name of the continuation channel b carrying v_2 until it has received the value v_1. Similarly, it cannot know the name of the continuation channel c carrying v_3 until it has received the value v_2. ◂

Example 5.3.6. The following closed process P receives a function f of type $\tau \to \sigma$ and a value x of type τ on the channel a. It evaluates $f(x)$ and sends the corresponding value of type σ on b, before forwarding a to b.

$$\cdot\,;\, a : (\tau \to \sigma) \wedge (\tau \wedge A) \vdash f \leftarrow \mathsf{input}\ a;\ x \leftarrow \mathsf{input}\ a;\ \_ \leftarrow \mathsf{output}\ b\ f(x);\ a \to b :: b : \sigma \wedge A$$

If we take $\tau = \sigma = (\delta \to \delta)$ for some type δ, then the following execution shows that sending P the process the values $\lambda x : \tau.x$ and $\lambda x : \delta.x$ on a causes it to send the value $\lambda x : \delta.x$ on b:

$$\begin{aligned}
&(\mathsf{msg}(c, _ \leftarrow \mathsf{output}\ b\ (\lambda x : \delta.x);\ d \to b),\\
&\qquad \mathsf{msg}(a, _ \leftarrow \mathsf{output}\ a\ (\lambda x : \tau.x);\ c \to a), \mathsf{proc}(b, P))\\
&\to (\mathsf{msg}(c, _ \leftarrow \mathsf{output}\ b\ (\lambda x : \delta.x);\ d \to b),\\
&\qquad \mathsf{proc}(b, x \leftarrow \mathsf{input}\ c;\ _ \leftarrow \mathsf{output}\ b\ (\lambda x : \tau.x)x;\ c \to b))\\
&\to \mathsf{proc}(b, _ \leftarrow \mathsf{output}\ b\ (\lambda x : \tau.x)(\lambda x : \delta.x);\ d \to b)\\
&\to \mathsf{proc}(e, d \to e), \mathsf{msg}(b, _ \leftarrow \mathsf{output}\ c\ (\lambda x : \delta.x);\ e \to c)
\end{aligned}$$

◂

Example 5.3.7. The following process receives a quoted process p of type $\{c : C \leftarrow a : B\}$ over a channel a. After receipt of p, the channel a has type B. The continuation process unquotes p to provide a channel c of type C using the channel a:

$$\Psi\,;\, a : \{c : C \leftarrow a : B\} \wedge B \vdash p \leftarrow \mathsf{input}\ a;\ c \leftarrow \{p\} \leftarrow a :: c : C.$$

◂

The protocol $\tau \supset A$ is the negative dual of $\tau \wedge A$. Recall that polar-dual session types prescribe the same kind of communications, but in opposite directions. In this case, where a provider of type $\tau \wedge A$ *sends* a value of type τ, a provider of type $\tau \supset A$ *receives* a value of type τ. The session type $\tau \supset A$ is formed by the rule (C$\supset$). The inference rules forming processes and the multiset rewrite rules describing their behaviour are the obvious duals of those for $\tau \wedge A$:

$$\frac{\Xi \vdash \tau\ \mathsf{type}_\mathsf{f} \quad \Xi \vdash A\ \mathsf{type}_\mathsf{s}^-}{\Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^-}\ (\mathrm{C}\supset)$$

$$\frac{\Psi, x : \tau\,;\, \Delta \vdash P :: a : A}{\Psi\,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A}\ (\supset\mathrm{R}) \qquad \frac{\Psi \Vdash M : \tau \quad \Psi\,;\, \Delta, a : A \vdash P :: c : C}{\Psi\,;\, \Delta, a : \tau \supset A \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: c : C}\ (\supset\mathrm{L})$$

$$\forall \Delta, a, d, c.\mathsf{proc}(a, x \leftarrow \mathsf{input}\ a;\ P), \mathsf{msg}(d, \_ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d) \to \to \mathsf{proc}(d, [d, v/a, x]P) \tag{76}$$

$$\forall a, \Delta.\mathbf{eval}(M, v), \mathsf{proc}(c, \_ \leftarrow \mathsf{output}\ a\ M;\ P) \to \to \exists d.\mathsf{msg}(d, \_ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d), \mathsf{proc}(c, [d/a]P) \tag{77}$$

5.3.3. Choices. Processes can choose between services. An internal choice type $\oplus\{l : A_l\}_{l \in L}$ prescribes a choice between session types $\{A_l\}_{l \in L}$ (L finite). The process $a.k;\ P$ chooses to provide the service A_k by sending the label k on a, and then continues as P. The process $\mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L}$ blocks until it receives a label k on a and then continues as P_k.

$$\frac{\Xi \vdash A_l\ \mathsf{type}_\mathsf{s}^+ \quad (\forall l \in L)}{\Xi \vdash \oplus\{l : A_l\}_{l \in L}\ \mathsf{type}_\mathsf{s}^+}\ (\mathrm{C}\oplus) \qquad \frac{\Psi\,;\, \Delta \vdash P :: a : A_k \quad (k \in L)}{\Psi\,;\, \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l \in L}}\ (\oplus\mathrm{R})$$

$$\frac{\Psi\,;\, \Delta, a : A_l \vdash P_l :: c : C \quad (\forall l \in L)}{\Psi\,;\, \Delta, a : \oplus\{l : A_l\}_{l \in L} \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L} :: c : C}\ (\oplus\mathrm{L})$$

$$\forall \Delta, a.\mathsf{proc}(a, a.k;\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, a.k;\ d \to a) \tag{78}$$

$$\forall a, d, \Delta, c.\mathsf{msg}(a, a.k;\ d \to a), \mathsf{proc}(c, \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L}) \to \mathsf{proc}(c, [d/a]P_k) \tag{79}$$

The polar dual of the internal choice type $\oplus\{l : A_l\}_{l \in L}$ is the external choice type $\&\{l : A_l\}_{l \in L}$.

Example 5.3.8. The following process P diverges if it receives the label `div` on a, and it forwards a to b if it receives the label `fwd`:

$$\cdot\,;\, a : \oplus\{\mathtt{div} : A, \mathtt{fwd} : B\} \vdash \mathsf{case}\ a\ \{\mathtt{div} \Rightarrow \Omega \mid \mathtt{fwd} \Rightarrow a \to b\} :: b : B.$$

Indeed, if a carries the label `div`, then:

$$\mathsf{msg}(a, a.\mathtt{div};\ c), \mathsf{proc}(b, P) \to \mathsf{proc}(b, \Omega).$$

If a carries the label `fwd`, then:

$$\mathsf{msg}(a, a.\mathtt{fwd};\ c), \mathsf{proc}(b, P) \to \mathsf{proc}(b, c \to b). \qquad \blacktriangleleft$$

5.3.4. Shifts in Polarity. Process communication is asynchronous. Synchronization on a channel is encoded using "polarity shifts" [PG15]. The positive protocol $\downarrow A$ prescribes a synchronization (a shift message) followed by communication satisfying the negative type A. The process $\mathsf{send}\ a\ \mathsf{shift};\ P$ signals that it is ready to receive on a by sending a "shift message" on a, and continues as P. The process $\mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P$ blocks until it receives the shift message and continues as P.

$$\frac{\Xi \vdash A\ \mathsf{type}_\mathsf{s}^-}{\Xi \vdash \downarrow A\ \mathsf{type}_\mathsf{s}^+}\ (\mathrm{C}\downarrow)$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: a : \downarrow A}\ (\downarrow\mathrm{R}) \qquad \frac{\Psi\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \downarrow A \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\downarrow\mathrm{L})$$

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{shift};\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a) \tag{80}$$

$$\forall \Delta, a, d, c.\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a), \mathsf{proc}(c, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [d/a]P) \tag{81}$$

The dual of the positive type $\downarrow A$ is the negative type $\uparrow A$.

At first glance, polarity shifts may appear to be special cases of choice types. The key difference is that the direction of communication does not change with choice types: the label and subsequent communications travel in the same direction. In contrast, the shift message and subsequent communications travel in opposite directions.

5.3.5. Recursive Types. The recursive type $\rho\alpha.A$ prescribes an "unfold" message followed by communication of type $[\rho\alpha.A/\alpha]A$. To ensure that unfolding a recursive type is well-defined, we require that the type variable α have the same polarity as the recursive type. The process $\mathsf{send}\ a\ \mathsf{unfold};\ P$ sends an unfold message and continues as P. The process $\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P$ receives an unfold message and continues as P.

$$\frac{}{\Xi, \alpha\ \mathsf{type}_\mathsf{s}^p \vdash \alpha\ \mathsf{type}_\mathsf{s}^p}\ (\mathrm{CV}\textsc{ar}) \qquad \frac{\Xi, \alpha\ \mathsf{type}_\mathsf{s}^+ \vdash A\ \mathsf{type}_\mathsf{s}^+}{\Xi \vdash \rho\alpha.A\ \mathsf{type}_\mathsf{s}^+}\ (\mathrm{C}\rho^+)$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_\mathsf{s}^+}{\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A}\ (\rho^+\mathrm{R})$$

$$\frac{\Psi\ ;\ \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_\mathsf{s}^+}{\Psi\ ;\ \Delta, a : \rho\alpha.A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\rho^+\mathrm{L})$$

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{unfold};\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a) \tag{82}$$

$$\forall \Delta, a, d.\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a), \mathsf{proc}(c, \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [d/a]P) \tag{83}$$

Example 5.3.9. The set of conatural numbers is given by $\mathbb{N} \cup \{\omega\}$, where $\mathbb{N}$ is the usual set of natural numbers, and ω corresponds to a countably infinite stack of successors $s(s(s(\cdots)))$. The protocol $\mathsf{conat} = \rho\alpha.\ \oplus\{\mathsf{z} : \mathbf{1}, \mathsf{s} : \alpha\}$ encodes conatural numbers. Indeed, a communication is either an infinite sequence of successor labels s, or some finite number of s labels followed by the zero label z and termination. The following recursive process outputs ω on o:

$$\cdot\ ;\ \cdot \vdash \mathsf{fix}\ \omega.\mathsf{send}\ o\ \mathsf{unfold};\ s.o;\ \omega :: o : \mathsf{conat}.$$

It has an infinite fair execution where for $n \geq 1$, the $(3n-2)$-th, $(3n-1)$-th, and $3n$-th rules are respectively instantiations of rules (70), (82), and (69). $\blacktriangleleft$

Example 5.3.10. Consider the type $\texttt{bits} = \rho\beta.\, \oplus\{0 : \beta, 1 : \beta\}$ of bit streams. Its communications consist of potentially infinite sequences of unfold messages interleaved with labels 0 and 1. The following process receives a bit stream on i, flips its bits, and outputs the result on o:

$$\begin{aligned}\Psi\ ;\ i : \texttt{bits} \vdash o \leftarrow \{\mathsf{fix}\ f.o \leftarrow \{&\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\\ &\mathsf{send}\ o\ \mathsf{unfold};\\ &\mathsf{case}\ i\ \{\ 0 \Rightarrow o.1;\ o \leftarrow \{f\} \leftarrow i\\ &\qquad\ \mid 1 \Rightarrow o.0;\ o \leftarrow \{f\} \leftarrow i\}\\ &\}\leftarrow i\} \leftarrow x :: o : \texttt{bits}\end{aligned}$$ ◀

We refer the reader to [TCP13] for further example processes.

5.4. Static Properties of Session Types

Session types are closed under substitution. Substitutions are context morphisms (definition 2.5.7). Concretely, a context morphism $\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \alpha_1\ \mathsf{type}^{p_1}_{\mathsf{s}}, \ldots, \alpha_n\ \mathsf{type}^{p_n}_{\mathsf{s}}$ for session types is a list σ of session types $A_1, \ldots, A_n$ such that $\Theta \vdash A_i\ \mathsf{type}^{p_i}_{\mathsf{s}}$ for $1 \le i \le n$.

Proposition 5.4.1 (Syntactic Substitution of Session Types). *Let $\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi$ be an arbitrary context morphism. If $\Xi \vdash A\ \mathsf{type}^{p}_{\mathsf{s}}$, then $\Theta \vdash [\sigma]A\ \mathsf{type}^{p}_{\mathsf{s}}$.*

5.5. Static Properties of Terms and Processes

We review several static properties about terms and processes. Though mundane, they will be used frequently and implicitly. The key ideas from this section are that terms and processes are closed under substitution, and that we can partition the free channels of a process as *input channels* and *output channels*.

5.5.1. Substitution. Typing for SILL terms and processes is closed under substitution, and the typing judgment is parametric. A context morphism $\sigma :_{\mathsf{f}} \Phi \rightsquigarrow x_1 : \tau_1, \ldots, x_n : \tau_n$ is a list σ of terms $N_1, \ldots, N_n$ satisfying $\Phi \Vdash N_i : \tau_i$ for all $1 \le i \le n$.

Proposition 5.5.1 (Syntactic Substitution of Terms). *Let $\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi$ be arbitrary.*

(1) If $\Psi \Vdash N : \tau$, then $\Phi \Vdash [\sigma]N : \tau$.

(2) If $\Psi\ ;\ \Delta \vdash P :: c : C$, then $\Phi\ ;\ \Delta \vdash [\sigma]P :: c : C$.

Proof. By induction on the derivation of $\Psi \Vdash M : \tau$ and $\Psi\ ;\ \Delta \vdash P :: a : A$. □

5.5.2. Free and Bound Channel Names. We begin by defining the free and bound channel names in a process. Given a well-typed process P, let the set $\mathsf{fc}(P)$ of **free channel names** in P be inductively defined on the structure of P by the following collection of equations:

$$\begin{aligned}
\mathsf{fc}(a \leftarrow P;\ Q) &= (\mathsf{fc}(P) \cup \mathsf{fc}(Q)) \setminus \{a\} & \mathsf{fc}(a \leftarrow \{M\} \leftarrow \overline{a_i}) &= \{a, \overline{a_i}\}\\
\mathsf{fc}(a \rightarrow b) &= \{a, b\} & \mathsf{fc}(a \leftarrow b) &= \{a, b\}\\
\mathsf{fc}(\mathsf{close}\ a) &= \{a\} & \mathsf{fc}(\mathsf{wait}\ a;\ P) &= \{a\} \cup \mathsf{fc}(P)\\
\mathsf{fc}(\mathsf{send}\ a\ b;\ P) &= \{a, b\} \cup \mathsf{fc}(P) & \mathsf{fc}(b \leftarrow \mathsf{recv}\ a;\ P) &= \{a\} \cup \mathsf{fc}(P) \setminus \{b\}\\
\mathsf{fc}(a.k;\ P) &= \{a\} \cup \mathsf{fc}(P) & \mathsf{fc}(\mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L}) &= a \cup \bigcup_{l \in L} \mathsf{fc}(P_l)\\
\mathsf{fc}(_ \leftarrow \mathsf{output}\ a\ M;\ P) &= \{a\} \cup \mathsf{fc}(P) & \mathsf{fc}(x \leftarrow \mathsf{input}\ a;\ P) &= \{a\} \cup \mathsf{fc}(P)\\
\mathsf{fc}(\mathsf{send}\ a\ \mathsf{unfold};\ P) &= \{a\} \cup \mathsf{fc}(P) & \mathsf{fc}(\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) &= \{a\} \cup \mathsf{fc}(P)
\end{aligned}$$

All other channel names in P are **bound** and can freely be α-varied.

Proposition 5.5.2. *If $\Psi\ ;\ c_1 : A_1, \ldots, c_n : A_n \vdash P :: c_0 : A_0$, then $\mathsf{fc}(P) = \{c_0, \ldots, c_n\}$.*

Proof. By induction on the derivation of $\Psi\ ;\ c_1 : A_1, \ldots, c_n : A_n \vdash P :: c_0 : A_0$. □

5.5.3. Input and Output. We partition SILL processes as **sending**, **receiving**, and **structural**. Structural processes are processes of the form $a \leftarrow P;\ Q$, $a \rightarrow b$, $a \leftarrow b$, and $a \leftarrow \{M\} \leftarrow \overline{a_i}$. Sending and receiving processes are respectively those in the left and right columns:

$\mathsf{close}\ a$	$\mathsf{wait}\ a;\ P$
$\mathsf{send}\ a\ b;\ P$	$b \leftarrow \mathsf{recv}\ a;\ P$
$a.k;\ P$	$\mathsf{case}\ a\ (l \Rightarrow P_l)_{l \in L}$
$\_ \leftarrow \mathsf{output}\ a\ M;\ P$	$x \leftarrow \mathsf{input}\ a;\ P$
$\mathsf{send}\ a\ \mathsf{unfold};\ P$	$\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P$

We say that the processes in the left column **send** on a, while the processes in the right column **block** or **receive** on a.

Using polarity, we can statically partition free channels into the sets $\mathsf{oc}(P)$ of **output channel names** and $\mathsf{ic}(P)$ of **input channel names**. Intuitively, $c \in \mathsf{oc}(P)$ if the next time P communicates on c, it sends a message on c; the meaning of $c \in \mathsf{ic}(P)$ is symmetric. Explicitly:

Definition 5.5.3. Assume $\Psi\ ;\ c_1 : A_1, \ldots, c_n : A_n \vdash P :: c_0 : A_0$. Let $\mathsf{oc}(P)$ and $\mathsf{ic}(P)$ be the least subsets of $\mathsf{fc}(P)$ such that:

(1) $c_0 \in \mathsf{oc}(P)$ if and only if A_0 is positive;
(2) $c_0 \in \mathsf{ic}(P)$ if and only if A_0 is negative;
(3) for $1 \le i \le n$, $c_i \in \mathsf{oc}(P)$ if and only if A_i is negative;
(4) for $1 \le i \le n$, $c_i \in \mathsf{ic}(P)$ if and only if A_i is positive. ◀

Remark 5.5.4. That $c \in \mathsf{oc}(P)$ does not imply that P will necessarily send a message on c. It only implies that if P eventually communicates on c, then the first such communication will be output. We will show for every channel name in a configuration appears as the output channel of at most one process or message, and as the input channel of at most one process or message. This fact will be useful in defining the observed communication semantics of chapter 6, for it will let us show that there is at most one message judgment associated to each channel.

If we took an extrinsic view of type theory, then we might want to statically determine whether a channel $c \in \mathsf{fc}(P)$ is an output channel or an output channel, without reference to typing rules or to the dynamics. Unfortunately, this is impossible due to the interplay between the functional and the process layers. Indeed, there is no way of determining on which channels a quoted process $a \leftarrow \{M\} \leftarrow \overline{a_i}$ will send or receive, short of evaluating M to a value $a \leftarrow \{P\} \leftarrow \overline{a_i}$, or looking at the polarities of the types appearing in the typing judgment. Despite this, we can in all other cases statically determine the input and output channel names appearing in a process purely from its syntax. Let $\mathsf{oc_s}(P) \subseteq \mathsf{fc}(P)$ be the subset of *static output channel names* inductively defined by:

$$\mathsf{oc_s}(a \leftarrow P;\ Q) = (\mathsf{oc_s}(P) \cup \mathsf{oc_s}(Q)) \setminus \{a\}$$
$$\mathsf{oc_s}(a \leftarrow \{P\} \leftarrow \overline{a_i}) = \varnothing$$

$\mathsf{oc_s}(a \rightarrow b) = \{b\}$	$\mathsf{oc_s}(a \leftarrow b) = \{a\}$
$\mathsf{oc_s}(\mathsf{close}\ a) = \{a\}$	$\mathsf{oc_s}(\mathsf{wait}\ a;\ P) = \mathsf{oc_s}(P)$
$\mathsf{oc_s}(\mathsf{send}\ a\ b;\ P) = \{a\} \cup \mathsf{oc_s}(P)$	$\mathsf{oc_s}(b \leftarrow \mathsf{recv}\ a;\ P) = \mathsf{oc_s}(P) \setminus \{a, b\}$
$\mathsf{oc_s}(a.k;\ P) = \{a\} \cup \mathsf{oc_s}(P)$	$\mathsf{oc_s}(\mathsf{case}\ a\ (l \Rightarrow P_l)_{l \in L}) = \left(\bigcup_{l \in L} \mathsf{oc_s}(P_l)\right) \setminus \{a\}$
$\mathsf{oc_s}(\_ \leftarrow \mathsf{output}\ a\ M;\ P) = \{a\} \cup \mathsf{oc_s}(P)$	$\mathsf{oc_s}(x \leftarrow \mathsf{input}\ a;\ P) = \mathsf{oc_s}(P) \setminus \{a\}$
$\mathsf{oc_s}(\mathsf{send}\ a\ \mathsf{unfold};\ P) = \{a\} \cup \mathsf{oc_s}(P)$	$\mathsf{oc_s}(\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) = \mathsf{oc_s}(P) \setminus \{a\}$

Symmetrically, the subset $\mathsf{ic_s}(P) \subseteq \mathsf{fc}(P)$ of *static input channel names* is inductively defined by:

$$\mathsf{ic_s}(a \leftarrow P;\ Q) = (\mathsf{ic_s}(P) \cup \mathsf{ic_s}(Q)) \setminus \{a\}$$
$$\mathsf{ic_s}(a \leftarrow \{P\} \leftarrow \overline{a_i}) = \varnothing$$
$$\mathsf{ic_s}(a \to b) = \{a\} \qquad \mathsf{ic_s}(a \leftarrow b) = \{b\}$$
$$\mathsf{ic_s}(\mathsf{close}\ a) = \varnothing \qquad \mathsf{ic_s}(\mathsf{wait}\ a;\ P) = \{a\} \cup \mathsf{ic_s}(P)$$
$$\mathsf{ic_s}(\mathsf{send}\ a\ b;\ P) = \mathsf{ic_s}(P) \setminus \{a\} \qquad \mathsf{ic_s}(b \leftarrow \mathsf{recv}\ a;\ P) = \{a\} \cup \mathsf{ic_s}(P) \setminus \{b\}$$
$$\mathsf{ic_s}(a.k;\ P) = \mathsf{ic_s}(P) \setminus \{a\} \qquad \mathsf{ic_s}(\mathsf{case}\ a\ (l \Rightarrow P_l)_{l \in L}) = \{a\} \cup \left(\bigcup_{l \in L} \mathsf{ic_s}(P_l) \right)$$
$$\mathsf{ic_s}(\_ \leftarrow \mathsf{output}\ a\ M;\ P) = \mathsf{ic_s}(P) \setminus \{a\} \qquad \mathsf{ic_s}(x \leftarrow \mathsf{input}\ a;\ P) = \{a\} \cup \mathsf{ic_s}(P)$$
$$\mathsf{ic_s}(\mathsf{send}\ a\ \mathsf{unfold};\ P) = \mathsf{ic_s}(P) \setminus \{a\} \qquad \mathsf{ic_s}(\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) = \{a\} \cup \mathsf{ic_s}(P)$$

This static view of input and output is consistent with the one given by polarity:

Proposition 5.5.5. *If* $\Psi\ ;\ \Delta \vdash P :: c : A$, *then* $\mathsf{oc_s}(P) \subseteq \mathsf{oc}(P)$ *and* $\mathsf{ic_s}(P) \subseteq \mathsf{ic}(P)$. *If (*E-{}*) does not appear the derivation of* $\Psi\ ;\ \Delta \vdash P :: c : A$, *then the above inclusions are equalities.*

Proof. By induction on the derivation of $\Psi\ ;\ \Delta \vdash P :: c : A$. □

Proposition 5.5.6. *Free channels of well-typed processes partition as input and output channels, i.e., if* $\Psi\ ;\ \Delta \vdash P :: c : A$, *then* $\mathsf{fc}(P) = \mathsf{ic}(P) \cup \mathsf{oc}(P)$ *and* $\mathsf{ic}(P) \cap \mathsf{oc}(P) = \varnothing$.

Proof. By induction on the derivation of $\Psi\ ;\ \Delta \vdash P :: c : A$. □

Using definition 5.5.3, we can deduce that composed processes do not both send or both receive on the private channel linking them, but instead one sends while the other receives:

Corollary 5.5.7. *If* $\Pi\ ;\ \Delta \vdash b \leftarrow P;\ Q :: c : A$, *then* $\mathsf{fc}(P) \cap \mathsf{fc}(Q) = \{b\}$, $\mathsf{oc}(P) \cap \mathsf{oc}(Q) = \varnothing$ *and* $\mathsf{ic}(P) \cap \mathsf{ic}(Q) = \varnothing$.

Proof. The last rule used to form $\Pi\ ;\ \Delta \vdash b \leftarrow P;\ Q :: c : A$ must have been (Cut). By well-formedness of (Cut), $\mathsf{fc}(P) \cap \mathsf{fc}(Q) = \{b\}$. By proposition 5.5.6, both of the intersections in the statement must be subsets of $\{b\}$. If $b \in \mathsf{oc}(P) \cap \mathsf{oc}(Q)$ or $b \in \mathsf{ic}(P) \cap \mathsf{ic}(Q)$, then it is simultaneously positive and negative by definition 5.5.3, a contradiction. So the two intersections are empty. □

5.6. Static Properties of Typed Configurations

In this section, we study various static properties of facts $\mathsf{proc}(c, P)$ and $\mathsf{msg}(c, m)$ and of the typing judgment $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Phi$. In section 5.6.1, we define various sets of channel names and show how they interact with the typing judgment. We explain how the rule (Conf-C) determines an associative and partially commutative composition operator in section 5.6.2. In section 5.6.3 we study various structural properties for the typing judgment. We finish by proving various technical lemmas that will be useful for proving the preservation property in section 5.9.

5.6.1. Sets of Channel Names. We lift the definitions of free and bound channel names to configurations in the obvious way:

$$\mathsf{fc}(\mathcal{C}) = \left(\bigcup_{\mathsf{proc}(c,P) \in \mathcal{C}} \mathsf{fc}(P) \right) \cup \left(\bigcup_{\mathsf{msg}(c,m) \in \mathcal{C}} \mathsf{fc}(m) \right).$$

Lifting input and output channels to configurations is a direct adaptation of definition 5.5.3:

Definition 5.6.1. Assume $c_1 : C_1, \ldots, c_n : C_n \mid \mathrm{I} \vdash \mathcal{C} :: a_0 : A_0, \ldots, a_m : A_m$. Let the **output channels** $\mathsf{oc}(\mathcal{C})$ and **input channels** $\mathsf{ic}(\mathcal{C})$ of $\mathcal{C}$ be the least subsets of $\mathsf{fc}(\mathcal{C})$ such that:

(1) for $0 \le i \le m$, $a_i \in \mathsf{oc}(\mathcal{C})$ if and only if A_i is positive;
(2) for $0 \le i \le m$, $a_i \in \mathsf{ic}(\mathcal{C})$ if and only if A_i is negative;

(3) for $1 \leq i \leq n$, $c_i \in \mathsf{oc}(\mathcal{C})$ if and only if C_i is negative;
(4) for $1 \leq i \leq n$, $c_i \in \mathsf{ic}(\mathcal{C})$ if and only if C_i is positive. ◀

Remark 5.6.2. Unlike the free channels of processes, the free channels of typed configurations are not partitioned as input and output channels, but as input, output, and internal channels. Recall that the internal channels of $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ are those in I.

It is also useful to specify the channel on which a message was sent—its "carrier channel"—as well as its continuation channel.

Definition 5.6.3. The **carrier channel** of a message fact is:

$$\begin{aligned}
\mathsf{cc}(\mathsf{msg}(a, \mathsf{close}\ a)) &= a \\
\mathsf{cc}(\mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a)) &= a \\
\mathsf{cc}(\mathsf{msg}(d, \mathsf{send}\ a\ b;\ a \leftarrow d)) &= a \\
\mathsf{cc}(\mathsf{msg}(a, a.k;\ d \to a)) &= a \\
\mathsf{cc}(\mathsf{msg}(d, a.k;\ a \leftarrow d)) &= a \\
\mathsf{cc}(\mathsf{msg}(a, _ \leftarrow \mathsf{output}\ a\ v;\ d \to a)) &= a \\
\mathsf{cc}(\mathsf{msg}(d, _ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d)) &= a \\
\mathsf{cc}(\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a)) &= a \\
\mathsf{cc}(\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \to d)) &= a \\
\mathsf{cc}(\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a)) &= a \\
\mathsf{cc}(\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{unfold};\ a \leftarrow d)) &= a
\end{aligned}$$ ◀

Definition 5.6.4. The **continuation channel** of a message fact, if defined, is:

$$\begin{aligned}
\mathsf{kc}(\mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a)) &= d \\
\mathsf{kc}(\mathsf{msg}(d, \mathsf{send}\ a\ b;\ a \leftarrow d)) &= d \\
\mathsf{kc}(\mathsf{msg}(a, a.k;\ d \to a)) &= d \\
\mathsf{kc}(\mathsf{msg}(d, a.k;\ a \leftarrow d)) &= d \\
\mathsf{kc}(\mathsf{msg}(a, _ \leftarrow \mathsf{output}\ a\ v;\ d \to a)) &= d \\
\mathsf{kc}(\mathsf{msg}(d, _ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d)) &= d \\
\mathsf{kc}(\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a)) &= d \\
\mathsf{kc}(\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \to d)) &= d \\
\mathsf{kc}(\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a)) &= d \\
\mathsf{kc}(\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{unfold};\ a \leftarrow d)) &= d
\end{aligned}$$

In particular, we remark that $\mathsf{kc}(\mathsf{msg}(a, \mathsf{close}\ a))$ is undefined. ◀

Remark 5.6.5. The carrier channel of a message fact is always an output channel, but the converse need not be true. Consider for example the message fact $\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a)$. Its carrier channel is a, but its output channels are a and its continuation channel d.

5.6.2. Associativity and Partial Commutativity. The composition notation $\mathcal{C}, \mathcal{D}$ for configurations does not specify along which channels two configurations $\mathcal{C}$ and $\mathcal{D}$ were composed. Indeed, given configurations $\Gamma \vdash \mathcal{C} :: \Phi, \Pi_1, \Pi_2$ and $\Gamma', \Pi_1, \Pi_2 \vdash \mathcal{D} :: \Xi$, one could conceivably compose $\mathcal{C}$ and $\mathcal{D}$ along Π_1 or along Π_2, giving composites $\mathcal{C}, \mathcal{D}$ with different interfaces and internal channels. Proposition 5.6.6 states that no such choice exists, and that if the composition $\mathcal{C}, \mathcal{D}$ exists, then its type is uniquely determined by the types of $\mathcal{C}$ and $\mathcal{D}$. Put differently, the rule (Conf-C) determines a partial function from pairs of configuration typing judgments to configuration judgments:

PROPOSITION 5.6.6. *Consider two judgments* $\Gamma_i \mid \mathrm{I}_i \vdash \mathcal{C}_i :: \Phi_i$ *for* $i = 1, 2$*. If they can be composed in a certain order using (*CONF-C*), then the type of their composition is uniquely determined. Explicitly, if the two following instantiations of (*CONF-C*) are valid, then their conclusions are equal:*

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Phi_1 \quad \Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Phi_2}{\Gamma \mid \mathrm{I} \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Phi} \qquad \frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Phi_1 \quad \Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Phi_2}{\Gamma' \mid \mathrm{I}' \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Phi'} \tag{84}$$

If $\Phi_1 \cap \Gamma_2$ *is non-empty and the composition on the left is valid, then the one on the right is not:*

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Phi_1 \quad \Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Phi_2}{\Gamma \mid \mathrm{I} \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Phi} \qquad \frac{\Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Phi_2 \quad \Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Phi_1}{\Gamma' \mid \mathrm{I}' \vdash \mathcal{C}_2, \mathcal{C}_1 :: \Phi'} \tag{85}$$

If $\Phi_1 \cap \Gamma_2$ *and* $\Phi_2 \cap \Gamma_1$ *are both empty, then the left composition in* (85) *is valid if and only if the right one is.*

Proof. Consider the two compositions in (84), and let Π_1 and Π_2 respectively be the channels along which $\mathcal{C}_1$ and $\mathcal{C}_2$ are composed. It is sufficient to show that $\Pi_1 = \Pi_2$. Let $c : A \in \Pi_1$ be arbitrary. Because it appears in Φ_1 and Γ_2, it must also appear in Π_2. Otherwise, $c : A$ would appear in both Γ' and Φ', resulting in a judgment that is not well-formed. So $\Pi_1 \subseteq \Pi_2$. A symmetric argument gives the opposite inclusion, whence the desired equality.

Assume $\Pi = \Phi_1 \cap \Gamma_2$ is non-empty and that the composition on the left of (85) is valid. Let $c : A \in \Pi$ be arbitrary. Suppose to the contrary that the composition on the right is valid. Then $c : A$ appears in both Γ' and Φ', resulting in a judgment that is not well-formed. So the composition on the right is not valid.

Assume $\Phi_1 \cap \Gamma_2$ and $\Phi_2 \cap \Gamma_1$ are both empty and that the left composition in (85) is valid. Then the channel names appearing in the left conclusion are all pairwise distinct. So the right composition is a valid instance of (CONF-C), where Π is empty. A symmetric argument gives that if the right composition is valid, then so is the left composition. □

As in a pluricategory, composition of configurations is associative, that is to say, for all derivations $\mathcal{D}_i$ for $1 \leq i \leq 3$, we identify the compositions

$$\frac{\mathcal{D}_1 \qquad \dfrac{\mathcal{D}_2 \qquad \mathcal{D}_3}{\Sigma_2, \check{\Pi}_{23} \parallel \Pi_{12}\Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2\Pi_{23} \quad \check{\Pi}_{23}, \Sigma_3 \parallel \Pi_{23}\Gamma_3 \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}}{\Sigma_1, \check{\Pi}_{12} \parallel \Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1\Pi_{12} \qquad \check{\Pi}_{12}, \Sigma_2, \check{\Pi}_{23}, \Sigma_3 \parallel \Pi_{12}\Gamma_2\Gamma_3 \mid \mathrm{I}_2\,\mathrm{I}_3 \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_2\Xi_3}}{\Sigma_1, \check{\Pi}_{12}, \Sigma_2, \check{\Pi}_{23}, \Sigma_3 \parallel \Gamma_1\Gamma_2\Gamma_3 \mid \mathrm{I}_1\Pi_{12}\,\mathrm{I}_2\Pi_{23}\,\mathrm{I}_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1\Xi_2\Xi_3}$$

and

$$\frac{\dfrac{\mathcal{D}_1 \qquad \mathcal{D}_2}{\Sigma_1, \check{\Pi}_{12} \parallel \Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1\Pi_{12} \quad \Sigma_2, \check{\Pi}_{23} \parallel \Pi_{12}\Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2\Pi_{23}} \qquad \mathcal{D}_3}{\Sigma_1, \check{\Pi}_{12}, \Sigma_2, \check{\Pi}_{23} \parallel \Gamma_1\Pi_{12}\Gamma_2\Pi_{23} \mid \mathrm{I}_1\,\mathrm{I}_2 \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_1\Xi_2\Pi_{23} \qquad \check{\Pi}_{23}, \Sigma_3 \parallel \Pi_{23}\Gamma_3 \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}}{\Sigma_1, \check{\Pi}_{12}, \Sigma_2, \check{\Pi}_{23}, \Sigma_3 \parallel \Gamma_1\Gamma_2\Gamma_3 \mid \mathrm{I}_1\Pi_{12}\,\mathrm{I}_2\Pi_{23}\,\mathrm{I}_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1\Xi_2\Xi_3}$$

It is not immediately obvious that we can always reassociate compositions. Indeed, reassociating a composition could plausibly result in a different conclusion, or result in conclusions that are not well-formed. Fortunately, compositions can always be reassociated. We rely on proposition 5.6.6 to construct the intermediary conclusions in the following statement:

PROPOSITION 5.6.7. *The composition*

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1 \qquad \dfrac{\Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2 \quad \Gamma_3 \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}{\Gamma_{23} \mid \mathrm{I}_{23} \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_{23}}}{\Gamma \mid \mathrm{I} \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi}$$

is valid if and only if

$$\frac{\dfrac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1 \quad \Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2}{\Gamma_{12} \mid \mathrm{I}_{12} \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_{12}} \qquad \Gamma_3 \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}{\Gamma \mid \mathrm{I} \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi}$$

is valid.

Proof. Assume that the first composition is valid. Then we recognize the right branch of the derivation as

$$\frac{\Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2' \Pi_{23} \quad \Pi_{23} \Gamma_3' \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}{\Gamma_2 \Gamma_3' \mid \mathrm{I}_2 \Pi_{23} \mathrm{I}_3 \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_2' \Xi_3}$$

for some Π_{23}. This Π_{23} and the conclusion are unique by proposition 5.6.6. Set $\Pi_2 = \Xi_1 \cap \Gamma_2$ and $\Pi_3 = \Xi_1 \cap \Gamma_3'$. Let Ξ_1' be such that $\Xi_1 = \Xi_1', \Pi_2, \Pi_3$, and let $\Gamma_2' = \Gamma_2 \setminus \Pi_2$, and $\Gamma_3'' = \Gamma_3' \setminus \Pi_3$. Then we recognize the first composition as

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1' \Pi_2 \Pi_3 \quad \Gamma_2' \Pi_2 \Gamma_3'' \Pi_3 \mid \mathrm{I}_2 \Pi_{23} \mathrm{I}_3 \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_2' \Xi_3}{\Gamma_1 \Gamma_2' \Gamma_3'' \mid \mathrm{I}_1 \Pi_2 \Pi_3 \mathrm{I}_2 \Pi_{23} \mathrm{I}_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1' \Xi_2' \Xi_3}$$

In particular, we remark that the channel names appearing in the contexts in the conclusion are all pairwise distinct. We show that the second composition is valid. We begin by showing that

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1 \quad \Gamma_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2}{\Gamma_{12} \mid \mathrm{I}_{12} \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_{12}}$$

is valid. Relying on proposition 5.6.6, we recognize it as the composition:

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1' \Pi_2 \Pi_3 \quad \Gamma_2' \Pi_2 \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2' \Pi_{23}}{\Gamma_1 \Gamma_2' \mid \mathrm{I}_1 \Pi_2 \mathrm{I}_2 \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_1' \Xi_2' \Pi_{23} \Pi_3}$$

By the above remark that channel names are all pairwise distinct, the conclusion is well-formed. Next, we recognize the second composition as the composition:

$$\frac{\Gamma_1 \Gamma_2' \mid \mathrm{I}_1 \Pi_2 \mathrm{I}_2 \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_1' \Xi_2' \Pi_{23} \Pi_3 \quad \Pi_{23} \Pi_3 \Gamma_3'' \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}{\Gamma_1 \Gamma_2' \Gamma_3'' \mid \mathrm{I}_1 \Pi_2 \mathrm{I}_2 \Pi_{23} \Pi_3 \mathrm{I}_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1' \Xi_2' \Xi_3}$$

Again, the conclusion is well-formed, and it is identical to the conclusion of the first derivation.

Conversely, assume the second composition is valid. We repeat an analogous argument to show that the first composition is valid. We recognize the left branch of the derivation as

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1' \Pi_{12} \quad \Pi_{12} \Gamma_2' \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2}{\Gamma_1 \Gamma_2' \mid \mathrm{I}_1 \Pi_{12} \mathrm{I}_2 \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_1' \Xi_2}$$

where Π_{12} is the interface along which $\mathcal{C}_1$ and $\mathcal{C}_2$ are composed, and Γ_2' and Ξ_1' are its complement in Γ_2 and Ξ_1, respectively. Again, this Π_{12} and the conclusion are unique. Set $\Pi_1 = \Xi_1' \cap \Gamma_3$ and $\Pi_2 = \Xi_2 \cap \Gamma_3$. Let $\Xi_1'' = \Xi_1' \setminus \Pi_1$ and $\Xi_2' = \Xi_2 \setminus \Pi_2$, and let Γ_3' be such that $\Gamma_3 = \Gamma_3', \Pi_1, \Pi_2$. Then we recognize the second composition as

$$\frac{\Gamma_1 \Gamma_2' \mid \mathrm{I}_1 \Pi_{12} \mathrm{I}_2 \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_1'' \Pi_1 \Xi_2' \Pi_2 \quad \Pi_1 \Pi_2 \Gamma_3 \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}{\Gamma_1 \Gamma_2' \Gamma_3 \mid \mathrm{I}_1 \Pi_{12} \mathrm{I}_2 \Pi_1 \Pi_2 \mathrm{I}_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1'' \Xi_2 \Xi_3}$$

Because it is well-formed, the channel names appearing in its conclusion are all pairwise distinct. We show that the first composition is valid. Its right branch is the following uniquely determined composition:

$$\frac{\Pi_{12} \Gamma_2' \mid \mathrm{I}_2 \vdash \mathcal{C}_2 :: \Xi_2' \Pi_2 \quad \Pi_1 \Pi_2 \Gamma_3 \mid \mathrm{I}_3 \vdash \mathcal{C}_3 :: \Xi_3}{\Pi_{12} \Gamma_2' \Pi_1 \Gamma_3 \mid \mathrm{I}_2 \Pi_2 \mathrm{I}_3 \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_2' \Xi_3}$$

Remark that $\Xi_1 = \Xi_1'' \Pi_1 \Pi_{12}$. Then we recognize the first composition as

$$\frac{\Gamma_1 \mid \mathrm{I}_1 \vdash \mathcal{C}_1 :: \Xi_1'' \Pi_1 \Pi_{12} \quad \Pi_{12} \Gamma_2' \Pi_1 \Gamma_3 \mid \mathrm{I}_2 \Pi_2 \mathrm{I}_3 \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_2' \Xi_3}{\Gamma_1 \Gamma_2' \Gamma_3 \mid \mathrm{I}_1 \Pi_1 \Pi_{12} \mathrm{I}_2 \Pi_2 \mathrm{I}_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1'' \Xi_2' \Xi_3}$$

Its conclusion is well-formed because the channel names are known to be pairwise distinct, so the first derivation is valid. We recognize its conclusion as the conclusion of the second composition. We conclude the result. □

Composition is also, as in a pluricategory, partially commutative[3]. Given respective derivations $\mathcal{D}_i$ for the judgments

(1) $\Sigma_1, \check{\Pi}_1 \parallel \Gamma_1 \mid I_1 \vdash \mathcal{C}_1 :: \Xi_1 \Pi_1$,
(2) $\Sigma_2, \check{\Pi}_2 \parallel \Gamma_2 \mid I_2 \vdash \mathcal{C}_2 :: \Xi_2 \Pi_2$,
(3) $\Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Pi_1 \Pi_2 \Gamma_3 \mid I_3 \vdash \mathcal{C}_3 :: \Xi_3$,

we identify the compositions

$$\dfrac{\overset{\mathcal{D}_1}{\Sigma_1, \check{\Pi}_1 \parallel \Gamma_1 \mid I_1 \vdash \mathcal{C}_1 :: \Xi_1 \Pi_1} \quad \dfrac{\overset{\mathcal{D}_2}{\Sigma_2, \check{\Pi}_2 \parallel \Gamma_2 \mid I_2 \vdash \mathcal{C}_2 :: \Xi_2 \Pi_2} \quad \overset{\mathcal{D}_3}{\Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Pi_1 \Pi_2 \Gamma_3 \mid I_3 \vdash \mathcal{C}_3 :: \Xi_3}}{\Sigma_2, \Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Pi_1 \Gamma_2 \Gamma_3 \mid I_2 \Pi_2 I_3 \vdash \mathcal{C}_2, \mathcal{C}_3 :: \Xi_2 \Xi_3}}{\Sigma_1, \Sigma_2, \Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Gamma_1 \Gamma_2 \Gamma_3 \mid I_1 \Pi_1 I_2 \Pi_2 I_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1 \Xi_2 \Xi_3}$$

and

$$\dfrac{\overset{\mathcal{D}_2}{\Sigma_2, \check{\Pi}_2 \parallel \Gamma_2 \mid I_2 \vdash \mathcal{C}_2 :: \Xi_2 \Pi_2} \quad \dfrac{\overset{\mathcal{D}_1}{\Sigma_1, \check{\Pi}_1 \parallel \Gamma_1 \mid I_1 \vdash \mathcal{C}_1 :: \Xi_1 \Pi_1} \quad \overset{\mathcal{D}_3}{\Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Pi_1 \Pi_2 \Gamma_3 \mid I_3 \vdash \mathcal{C}_3 :: \Xi_3}}{\Sigma_1, \Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Gamma_1 \Pi_2 \Gamma_3 \mid I_1 \Pi_1 I_3 \vdash \mathcal{C}_1, \mathcal{C}_3 :: \Xi_1 \Xi_3}}{\Sigma_1, \Sigma_2, \Sigma_3, \check{\Pi}_1, \check{\Pi}_2 \parallel \Gamma_1 \Gamma_2 \Gamma_3 \mid I_1 \Pi_1 I_2 \Pi_2 I_3 \vdash \mathcal{C}_2, \mathcal{C}_1, \mathcal{C}_3 :: \Xi_1 \Xi_2 \Xi_3}$$

Given respective derivations $\mathcal{D}_i$ for the judgments

(1) $\Sigma_1, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \mid I_1 \vdash \mathcal{C}_1 :: \Xi_1 \Pi_2 \Pi_3$,
(2) $\Sigma_2, \check{\Pi}_2 \parallel \Gamma_2 \Pi_2 \mid I_2 \vdash \mathcal{C}_2 :: \Xi_2$,
(3) $\Sigma_3, \check{\Pi}_3 \parallel \Gamma_3 \Pi_3 \mid I_3 \vdash \mathcal{C}_3 :: \Xi_3$,

we identify the compositions

$$\dfrac{\dfrac{\overset{\mathcal{D}_1}{\Sigma_1, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \mid I_1 \vdash \mathcal{C}_1 :: \Xi_1 \Pi_2 \Pi_3} \quad \overset{\mathcal{D}_2}{\Sigma_2, \check{\Pi}_2 \parallel \Gamma_2 \Pi_2 \mid I_2 \vdash \mathcal{C}_2 :: \Xi_2}}{\Sigma_1, \Sigma_2, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \Gamma_2 \mid I_1 \Pi_2 I_2 \vdash \mathcal{C}_1, \mathcal{C}_2 :: \Xi_1 \Xi_2 \Pi_3} \quad \overset{\mathcal{D}_3}{\Sigma_3, \check{\Pi}_3 \parallel \Gamma_3 \Pi_3 \mid I_3 \vdash \mathcal{C}_3 :: \Xi_3}}{\Sigma_1, \Sigma_2, \Sigma_3, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \Gamma_2 \Gamma_3 \mid I_1 \Pi_2 I_2 \Pi_3 I_3 \vdash \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 :: \Xi_1 \Xi_2 \Xi_3}$$

and

$$\dfrac{\dfrac{\overset{\mathcal{D}_1}{\Sigma_1, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \mid I_1 \vdash \mathcal{C}_1 :: \Xi_1 \Pi_2 \Pi_3} \quad \overset{\mathcal{D}_3}{\Sigma_3, \check{\Pi}_3 \parallel \Gamma_3 \Pi_3 \mid I_3 \vdash \mathcal{C}_3 :: \Xi_3}}{\Sigma_1, \Sigma_3, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \Gamma_3 \mid I_1 \Pi_3 I_3 \vdash \mathcal{C}_1, \mathcal{C}_3 :: \Xi_1 \Pi_2 \Xi_3} \quad \overset{\mathcal{D}_2}{\Sigma_2, \check{\Pi}_2 \parallel \Gamma_2 \Pi_2 \mid I_2 \vdash \mathcal{C}_2 :: \Xi_2}}{\Sigma_1, \Sigma_2, \Sigma_3, \check{\Pi}_2, \check{\Pi}_3 \parallel \Gamma_1 \Gamma_2 \Gamma_3 \mid I_1 \Pi_2 I_2 \Pi_3 I_3 \vdash \mathcal{C}_1, \mathcal{C}_3, \mathcal{C}_2 :: \Xi_1 \Xi_2 \Xi_3}$$

We remark that the types associated to channels remain identical after reassociation and after commutation. This obvious fact will be repeatedly used without mention.

We conjecture that a graphical language similar to ones for monoidal categories (see section 2.1.2) could be adapted to pluricategories: conceivably, the sole change required would be to allow multiple wires to be joined between boxes. Such a graphical language would significantly simplify reasoning about our composition operator. Indeed, many proofs below involve tediously reassociating and commuting compositions, while graphically, this simply corresponds to continuously deforming diagrams. Unfortunately, developing sound graphical languages is a non-trivial task whose subtleties have ensnared many who have attempted it (cf. [JS91, p. 57]) and we leave such a graphical language for future work.

[3]The following statement of partial commutativity is simpler than in a general pluricategory because we are considering objects drawn from a free commutative monoid, instead of the usual case of a free monoid.

5.6.3. Structural Properties. We prove several properties about the typing judgments for multisets-in-context. Though largely technical, these structural properties will be indispensable to showing our preservation theorem, and to relating congruence relations on configurations to congruence relations on processes. We start by showing the subformula property and an inversion-style principle.

Proposition 5.6.8 (Subformula Property). *If* $\Gamma' \mid \mathrm{I}' \vdash \mathcal{E} :: \Delta'$ *appears in the derivation of* $\Gamma \mid \mathrm{I} \vdash \mathcal{F} :: \Delta$*, then* $\Gamma' \subseteq \Gamma\mathrm{I}$*,* $\mathrm{I}' \subseteq \mathrm{I}$*, and* $\Delta' \subseteq \mathrm{I}\Delta$*.*

Proof. By induction on the derivation of $\Gamma \mid \mathrm{I} \vdash \mathcal{F} :: \Delta$.

Case (Conf-M)**:** Then $\Gamma' \mid \mathrm{I}' \vdash \mathcal{E} :: \Delta'$ is the conclusion of the rule and the result is immediate.
Case (Conf-P)**:** Then $\Gamma' \mid \mathrm{I}' \vdash \mathcal{E} :: \Delta'$ is the conclusion of the rule and the result is immediate.
Case (Conf-C)**:**

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \qquad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1\Pi\mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\textsc{Conf-C})$$

We consider three subcases:
(1) if $\Gamma' \mid \mathrm{I}' \vdash \mathcal{E} :: \Delta'$ is the conclusion of the rule, the result is immediate.
(2) if $\Gamma' \mid \mathrm{I}' \vdash \mathcal{E} :: \Delta'$ appears in the derivation of the left premise, then the result follows by the induction hypothesis.
(3) if $\Gamma' \mid \mathrm{I}' \vdash \mathcal{E} :: \Delta'$ appears in the derivation of the right premise, then the result follows by the induction hypothesis. □

The following proposition specifies an inversion principle (cf. [Har16, Lemma 8.2]) for message facts. It also states the relationship between free channels in configuration typing judgments and process typing judgments.

Proposition 5.6.9 (Inversion Principle). *If* $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$*, and* $\mathsf{msg}(c_0, P) \in \mathcal{C}$ *or* $\mathsf{proc}(c_0, P) \in \mathcal{C}$*, then* $\cdot\ ;\ \Lambda \vdash P :: c_0 : A$ *for some* $\Lambda \subseteq \Gamma\mathrm{I}$ *and* $c_0 : A \in \mathrm{I}\Delta$*. Concretely, if* $\mathsf{proc}(c_0, P) \in \mathcal{C}$ *or* $\mathsf{msg}(c_0, P) \in \mathcal{C}$*, then*

(1) $c_0 \in \mathsf{fc}(P)$*;*
(2) for all $c_i \in \mathsf{fc}(P)$*, then* $c_i : A_i \in \Gamma\mathrm{I}\Delta$ *for some* A_i*; and*
(3) where $\mathsf{fc}(P) = \{c_0, \ldots, c_m\}$*, we have* $\cdot\ ; c_1 : A_1, \ldots, c_m : A_m \vdash P :: c_0 : A_0$*.*

If $\mathsf{msg}(c_0, m) \in \mathcal{C}$*, then* $\cdot\ ; \Lambda \vdash m :: c_0 : A$ *is given by:*

- *if* $m = \mathsf{close}\ c$*, then* $\cdot\ ; \cdot \vdash \mathsf{close}\ c :: c : \mathbf{1}$*;*
- *if* $m = c.l_j$*;* $d \to c$*, then* $\cdot\ ; d : A_j \vdash c.l_j$*;* $d \to c :: c : \oplus\{l_i : A_i\}_{i \in I}$ *for some* A_i *with* $i, j \in I$*;*
- *if* $m = c.l_j$*;* $c \leftarrow d$*, then* $\cdot\ ; c : \&\{l_i : A_i\}_{i \in I} \vdash c.l_j$*;* $c \leftarrow d :: d : A_j$ *for some* A_i *with* $i, j \in I$*;*
- *if* $m = \mathsf{send}\ c\ a$*;* $b \to c$*, then* $\cdot\ ; a : A, b : B \vdash \mathsf{send}\ c\ a$*;* $b \to c :: c : A \otimes B$ *for some* A *and* B*;*
- *if* $m = \mathsf{send}\ c\ a$*;* $c \leftarrow b$*, then* $\cdot\ ; a : A, c : A \multimap B \vdash \mathsf{send}\ c\ a$*;* $c \leftarrow b :: b : B$ *for some* A *and* B*;*
- *if* $m = \_ \leftarrow \mathsf{output}\ c\ v$*;* $d \to c$*, then* $\cdot\ ; d : A \vdash \_ \leftarrow \mathsf{output}\ c\ v$*;* $d \to c :: c : \tau \wedge A$ *for some* A *and* τ *such that* $\cdot \Vdash v : \tau$*;*
- *if* $m = \_ \leftarrow \mathsf{output}\ c\ v$*;* $c \leftarrow d$*, then* $\cdot\ ; c : \tau \supset A \vdash \_ \leftarrow \mathsf{output}\ c\ v$*;* $c \leftarrow d :: d : A$ *for some* A *and* τ *such that* $\cdot \Vdash v : \tau$*;*
- *if* $m = \mathsf{send}\ c\ \mathsf{unfold}$*;* $d \to c$*, then* $\cdot\ ; d : [\rho\alpha.A/\alpha]A \vdash \mathsf{send}\ c\ \mathsf{unfold}$*;* $d \to c :: c : \rho\alpha.A$ *for some* $\alpha \vdash A\ \mathsf{type}_s^+$*; and*
- *if* $m = \mathsf{send}\ c\ \mathsf{unfold}$*;* $c \leftarrow d$*, then* $\cdot\ ; c : \rho\alpha.A \vdash \mathsf{send}\ c\ \mathsf{unfold}$*;* $c \leftarrow d :: d : [\rho\alpha.A/\alpha]A$ *for some* $\alpha \vdash A\ \mathsf{type}_s^-$*.*

Proof. By induction on $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$. In the cases (Conf-M) and (Conf-P), the result follows by inversion on the typing judgment for the process that is the rule hypothesis. Proposition 5.6.8 gives the result in the case of (Conf-C). □

Typing judgments assign a type to every free channel appearing in a configuration:

Proposition 5.6.10. *If* $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$, *then* $\mathsf{fc}(\mathcal{C}) \subseteq \check{\Gamma}, \check{\mathrm{I}}, \check{\Delta} \subseteq \Sigma$.

Proof. Immediate by propositions 5.5.2 and 5.6.9. □

The following lemma states that shared channels in well-typed multisets are always internal channels:

Lemma 5.6.11. *If* $\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{F} :: \Theta$ *and* $c \in \mathsf{fc}(\mathcal{E}) \cap \mathsf{fc}(\mathcal{F})$, *then* $c \in \check{\mathrm{I}}$.

Proof. By induction on the derivation of $\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{F} :: \Theta$.

Case (Conf-M)**:** This case is impossible, for there are at least two elements in the multiset, but the conclusion of the rule only has one element.

Case (Conf-P)**:** Analogous to case (conf-m).

Case (Conf-C)**:**

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\text{Conf-C})$$

We consider three subcases:

(1) if $c \in \mathsf{fc}(\mathcal{C}) \cap \mathsf{fc}(\mathcal{D})$, then by the side-condition $\Sigma \cap \Sigma' = \varnothing$ on (Conf-C), it must be that $c \in \check{\Pi}$. But $\mathrm{I} = \mathrm{I}_1 \Pi \mathrm{I}_2$, so $c \in \check{\mathrm{I}}$ as desired.
(2) if $c \in \mathsf{fc}(\mathcal{C})$ but $c \notin \mathsf{fc}(\mathcal{D})$, then $\mathcal{E}$ and $\mathcal{F}$ must both intersect with $\mathcal{C}$. Applying the induction hypothesis to the left premise of the rule, we get that $c \in \check{\mathrm{I}}_1$. It follows that $c \in \check{\mathrm{I}}$.
(3) if $c \in \mathsf{fc}(\mathcal{D})$ but $c \notin \mathsf{fc}(\mathcal{C})$: this subcase is symmetric to the previous one. □

The following lemma specifies that message or process facts in multisets do not have shared input channels, and that they do not have shared output channels. It will repeatedly be used in the proof that Polarized SILL's multiset rewriting system is non-overlapping on initial process configurations.

Lemma 5.6.12. *Let* $J \in \{\mathsf{msg}(c, P), \mathsf{proc}(c, P)\}$ *and* $K \in \{\mathsf{msg}(d, Q), \mathsf{proc}(d, Q)\}$ *be arbitrary, and assume* $J \neq K$. *If* $\Gamma \mid \mathrm{I} \vdash \mathcal{E}, J, K :: \Phi$ *with* $\mathcal{E}$ *potentially empty, then* $\mathsf{oc}(J) \cap \mathsf{oc}(K) = \varnothing$ *and* $\mathsf{ic}(J) \cap \mathsf{ic}(K) = \varnothing$.

Proof. By induction on the derivation of $\Gamma \mid \mathrm{I} \vdash \mathcal{C}, J, K :: \Phi$.

Case (Conf-M)**:** This case is impossible, for its conclusion contains a single fact.

Case (Conf-P)**:** This case is impossible, for its conclusion contains a single fact.

Case (Conf-C)**:** Recall the rule:

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\text{Conf-C})$$

We proceed by case analysis on where J and K are located. If they are both located in the same branch, then we are done by the induction hypothesis. Assume without loss of generality that J is in the left branch and that K is in the right branch. Suppose to the contrary that there exists some $c' \in (\mathsf{oc}(J) \cap \mathsf{oc}(K)) \cup (\mathsf{ic}(J) \cap \mathsf{ic}(K))$. By lemma 5.6.11, we have $c' : A \in \Pi$ for some A. By proposition 5.6.9 we have $\cdot\,; \Delta \vdash P :: c' : A$ and $\cdot\,; \Delta', c' : A \vdash Q :: d : B$ for some Δ, Δ', and B. By definition 5.5.3, this implies that A is simultaneously positive and negative, a contradiction. So the intersections are empty. □

In particular, lemma 5.6.12 implies that if a channel c is already carrying a message, then no other process in the configuration will output on c.

Lemma 5.6.13. *If* $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ *and* $\mathsf{msg}(c, m) \in \mathcal{C}$, *then* $\mathsf{cc}(\mathsf{msg}(c, m)) \notin \mathsf{oc}(K)$ *for all other* $K \in \mathcal{C}$.

Proof. Immediate by the observation that $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathsf{oc}(\mathsf{msg}(c, m))$ and lemma 5.6.12. □

Because processes are not uniquely typed, configurations do not in general have unique types. The following lemma shows that if a subset of a well-typed configuration can be assigned a type, then it can be assigned a type that agrees with the type of the configuration that contains it.

LEMMA 5.6.14. *If* $\Psi \mid \mathrm{I} \vdash \mathcal{E} :: \Theta$ *and* $\mathcal{F} \subseteq \mathcal{E}$ *is such that* $\Gamma' \mid \mathrm{I}' \vdash \mathcal{F} :: \Delta'$ *for some* Γ', I', *and* Δ', *then there exist* $\Gamma'' \subseteq \Psi\,\mathrm{I}$, $\mathrm{I}'' \subseteq \mathrm{I}$, *and* $\Delta'' \subseteq \mathrm{I}\Theta$ *such that* $\Gamma'' \mid \mathrm{I}'' \vdash \mathcal{F} :: \Delta''$.

Proof. By induction on the derivation of $\Gamma' \mid \mathrm{I}' \vdash \mathcal{F} :: \Delta'$.

CASE (CONF-M)**:**

$$\frac{\cdot\,;\,\Delta \vdash m :: c : A}{\Sigma \parallel \Delta \mid \cdot \vdash \mathsf{msg}(c, m) :: (c : A)}\ (\textsc{Conf-M})$$

Then $\mathcal{F}$ is $\mathsf{msg}(c, m) \in \mathcal{E}$. Apply proposition 5.6.9 and (CONF-M) to get the result.

CASE (CONF-P)**:** Analogous to (CONF-M).

CASE (CONF-C)**:**

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\textsc{Conf-C})$$

By the induction hypotheses, $\Gamma' \mid \mathrm{I}'_1 \vdash \mathcal{C} :: \Phi'\Pi'$ and $\Pi''\Lambda' \mid \mathrm{I}'_2 \vdash \mathcal{D} :: \Xi'$ with

$$\begin{aligned} \Gamma' &\subseteq \Psi\,\mathrm{I}, & \Pi''\Lambda' &\subseteq \Psi\,\mathrm{I}, \\ \mathrm{I}'_1 &\subseteq \mathrm{I}, & \mathrm{I}'_2 &\subseteq \mathrm{I}, \\ \Phi'\Pi' &\subseteq \mathrm{I}\Theta, & \Xi' &\subseteq \mathrm{I}\Theta. \end{aligned}$$

By lemma 5.6.11, $\check{\Pi}' = \check{\Pi}'' \subseteq \check{\mathrm{I}}$. Because I uniquely assigns types to channel names, it must be that $\Pi' = \Pi''$. So by (CONF-C), $\Gamma'\Lambda' \mid \mathrm{I}'_1\Pi'\mathrm{I}'_2 \vdash \mathcal{F} :: \Phi'\Xi'$. The above inclusions imply $\Gamma'\Lambda' \subseteq \Psi\,\mathrm{I}$, $\mathrm{I}'_1\mathrm{I}'_2 \subseteq \mathrm{I}$, and $\Phi'\Xi' \subseteq \mathrm{I}\Theta$, as desired. □

To show preservation in proposition 5.9.1, we will need to show that replacing a subset matched by a rewriting rule with a multiset having the same interface does not affect the interface of the whole multiset. To do so, we will need to reason about intersecting multisets, and reassociate and commute compositions so that all elements in an intersection appear together. The following proposition shows us that we can group elements in the intersection of two consistent multisets together in the typing derivation. In its proof, we use three dots "⋯" to elide the unique conclusion given by functionality of composition (proposition 5.6.6).

PROPOSITION 5.6.15 (Intersection Property). *Assume that* $\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L$ *and that* $\Gamma_R \mid \mathrm{I}_R \vdash \mathcal{R} :: \Phi_R$. *Also assume that* $\mathcal{M} = \mathcal{L} \cap \mathcal{R}$ *is non-empty. Assume that the two typing judgments agree on their intersection, i.e., that for all* $c \in \mathsf{fc}(\mathcal{M})$, $c : A \in \Gamma_L \mathrm{I}_L \Phi_L$ *if and only if* $c : A \in \Gamma_R \mathrm{I}_R \Phi_R$. *Set*

$$\begin{aligned} \Gamma_M &= \Gamma_R \cap (\Gamma_L \cup \mathrm{I}_L) & \Gamma'_L &= \Gamma_L \setminus \Gamma_R & \Gamma'_R &= (\Gamma_R \setminus (\Gamma_L \cup \mathrm{I}_L)) \cup (\Phi_L \cap \mathrm{I}_R) \\ \mathrm{I}_M &= \mathrm{I}_L \cap \mathrm{I}_R & \mathrm{I}'_L &= \mathrm{I}_L \setminus (\Gamma_R \cup \mathrm{I}_R) & \mathrm{I}'_R &= \mathrm{I}_R \setminus (\mathrm{I}_L \cup \Phi_L) \\ \mathcal{M} &= \mathcal{L} \cap \mathcal{R}, & \mathcal{L}' &= \mathcal{L} \setminus \mathcal{M}, & \mathcal{R}' &= \mathcal{R} \setminus \mathcal{M}, \\ \Phi_M &= \Phi_L \cap (\mathrm{I}_R \cup \Phi_R) & \Phi'_L &= (\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cup (\mathrm{I}_L \cap \Gamma_R) & \Phi'_R &= \Phi_R \setminus \Phi_L \end{aligned}$$

Then each of the following rules is a valid instance of (CONF-C) *if its premisses are both non-empty:*

$$\frac{\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L}' :: \Phi'_L \quad \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L} \qquad \frac{\Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M \quad \Gamma'_R \mid \mathrm{I}'_R \vdash \mathcal{R}' :: \Phi'_R}{\Gamma_R \mid \mathrm{I}_R \vdash \mathcal{R} :: \Phi_R}$$

If there exist derivations $\mathcal{D}_L$ *and* $\mathcal{D}_R$ *for* $\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L$ *and* $\Gamma_R \mid \mathrm{I}_R \vdash \mathcal{R} :: \Phi_R$, *respectively, then there exist derivations* $\mathcal{D}'_L$, $\mathcal{D}_M$, *and* $\mathcal{D}'_R$ *for* $\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L}' :: \Phi'_L$, $\Gamma_R \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_R$, *and* $\Gamma_R \mid \mathrm{I}'_R \vdash \mathcal{R}' :: \Phi_R$, *respectively, whenever the respective multiset is non-empty.*

Proof. We first show the result for $\mathcal{L}$. If $\mathcal{M} = \mathcal{L}$, then the result is immediate. Assume now that $\mathcal{M} \neq \mathcal{L}$, or equivalently, that $\mathcal{L}'$ is non-empty. We begin by checking that

$$\frac{\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L}' :: \Phi'_L \quad \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M}{\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Phi}$$

is a valid instance of (Conf-C). In particular, we observe that the channel names appearing in $\Gamma'_L \mathrm{I}'_L \Phi'_L$ are all pairwise-distinct, as are those appearing in $\Gamma_M \mathrm{I}_M \Phi_M$. We also observe that the channel names in $\Phi'_L \Gamma_M$ not in the intersection $\Phi'_L \cap \Gamma_M$ are all pairwise distinct. Indeed, if $c : A$ is not in the intersection but $c : A \in \Phi'_L$, then if $c : B \in \Gamma_M$ for some B, then by hypothesis we have both $c : A \in \Gamma_L \mathrm{I}_L \Phi_L$ and $c : B \in \Gamma_L \mathrm{I}_L \Phi_L$. Then $A = B$ because the judgment for $\mathcal{L}$ is well formed. So $c : A$ is in the intersection $\Phi'_L \cap \Gamma_M$, a contradiction. An identical argument covers the case when $c : A$ is not in the intersection but $c : A \in \Gamma_M$.

Next, we check that the conclusion is $\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L$. We compute:

$$\begin{aligned}
\Gamma_M \setminus \Phi'_L &= (\Gamma_R \cap (\Gamma_L \cup \mathrm{I}_L)) \setminus \Phi'_L \\
&= ((\Gamma_R \cap \Gamma_L) \setminus \Phi'_L) \cup ((\Gamma_R \cap \mathrm{I}_L) \setminus \Phi'_L)
\end{aligned}$$

because $\Gamma_R \cap \mathrm{I}_L \subseteq \Phi'_L$:

$$= (\Gamma_R \cap \Gamma_L) \setminus \Phi'_L$$

because $\Gamma_L \cap \mathrm{I}_L = \Gamma_L \cap \Phi_L = \varnothing$:

$$\begin{aligned}
&= (\Gamma_R \cap \Gamma_L). \\
\Gamma &= \Gamma'_L \cup (\Gamma_M \setminus \Phi'_L) \\
&= (\Gamma_L \setminus \Gamma_R) \cup (\Gamma_R \cap \Gamma_L) \\
&= \Gamma_L.
\end{aligned}$$

$$\begin{aligned}
\Phi'_L \cap \Gamma_M &= ((\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cup (\mathrm{I}_L \cap \Gamma_R)) \cap (\Gamma_R \cap (\Gamma_L \cup \mathrm{I}_L)) \\
&= ((\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cup (\mathrm{I}_L \cap \Gamma_R)) \cap ((\Gamma_R \cap \Gamma_L) \cup (\Gamma_R \cap \mathrm{I}_L)) \\
&= ((\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cap (\Gamma_R \cap \Gamma_L)) \cup ((\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cap (\Gamma_R \cap \mathrm{I}_L)) \cup \\
&\quad \cup ((\mathrm{I}_L \cap \Gamma_R) \cap (\Gamma_R \cap \Gamma_L)) \cup ((\mathrm{I}_L \cap \Gamma_R) \cap (\Gamma_R \cap \mathrm{I}_L))
\end{aligned}$$

because $\Phi_L \cap \mathrm{I}_L = \mathrm{I}_L \cap \Gamma_L = \varnothing$:

$$= ((\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cap (\Gamma_R \cap \Gamma_L)) \cup (\mathrm{I}_L \cap \Gamma_R)$$

by De Morgan's law:

$$= ((\Phi_L \setminus \mathrm{I}_R) \cap (\Phi_L \setminus \Phi_R) \cap \Gamma_R \cap \Gamma_L) \cup (\mathrm{I}_L \cap \Gamma_R)$$

because $\Phi_L \cap \Gamma_L = \varnothing$:

$$\begin{aligned}
&= \mathrm{I}_L \cap \Gamma_R. \\
\mathrm{I} &= \mathrm{I}'_L \cup (\Phi'_L \cap \Gamma_M) \cup \mathrm{I}_M \\
&= \mathrm{I}_L \setminus (\Gamma_R \cup \mathrm{I}_R) \cup (\mathrm{I}_L \cap \Gamma_R) \cup (\mathrm{I}_L \cap \mathrm{I}_R) \\
&= \mathrm{I}_L \setminus (\Gamma_R \cup \mathrm{I}_R) \cup (\mathrm{I}_L \cap (\Gamma_R \cup \mathrm{I}_R)) \\
&= \mathrm{I}_L.
\end{aligned}$$

$$\begin{aligned}
\mathcal{C} &= \mathcal{L}', \mathcal{M} \\
&= \mathcal{L} \setminus \mathcal{M}, \mathcal{M} \\
&= \mathcal{L}.
\end{aligned}$$

$$\begin{aligned}
\Phi'_L \setminus \Gamma_M &= ((\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \cup (\mathrm{I}_L \cap \Gamma_R)) \setminus (\Gamma_R \cap (\Gamma_L \cup \mathrm{I}_L)) \\
&= (\Phi_L \setminus (\mathrm{I}_R \cup \Phi_R)) \setminus (\Gamma_R \cap (\Gamma_L \cup \mathrm{I}_L))
\end{aligned}$$

because $\Phi_L \cap (\Gamma_L \cup \mathrm{I}_L) = \varnothing$:

$$\begin{aligned} &= \Phi_L \setminus (\mathrm{I}_R \cup \Phi_R). \\ \Phi &= \Phi_M \cup (\Phi'_L \setminus \Gamma_M) \\ &= (\Phi_L \cap (\mathrm{I}_R \cup \Phi_R)) \cup \Phi_L \setminus (\mathrm{I}_R \cup \Phi_R) \\ &= \Phi_L. \end{aligned}$$

We conclude that the conclusion is indeed $\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L$.

Next, assume that there exists a derivation $\mathcal{D}_L$ for $\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L$. We show that there exist derivations $\mathcal{D}'_L$ and $\mathcal{D}_M$ such that

$$\frac{\begin{array}{cc}\mathcal{D}'_L & \mathcal{D}_M \\ \Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L}' :: \Phi'_L & \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M\end{array}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}\ (\textsc{Conf-C})$$

is a valid derivation. To do so, we proceed by induction on $\mathcal{D}_L$. Because we assumed that $\mathcal{M} \neq \mathcal{L}$, the last rule in $\mathcal{D}_L$ is an instance of (Conf-C):

$$\frac{\begin{array}{cc}\mathcal{D}_F & \mathcal{D}_G \\ \Gamma_F \mid \mathrm{I}_F \vdash \mathcal{F} :: \Phi_F & \Gamma_G \mid \mathrm{I}_G \vdash \mathcal{G} :: \Phi_G\end{array}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}\ (\textsc{Conf-C})$$

We proceed by case analysis on the relationship between its hypotheses and $\mathcal{M}$:

(1) If $\mathcal{F} = \mathcal{L}'$ and $\mathcal{G} = \mathcal{M}$, then we are done by the subformula property: take $\mathcal{D}'_L = \mathcal{D}_F$ and $\mathcal{D}_M = \mathcal{D}_G$.

(2) If $\mathcal{M} \subsetneq \mathcal{G}$, then by the induction hypothesis applied to $\mathcal{D}_G$ we have derivations $\mathcal{D}'_G$ and $\mathcal{D}_M$ such that

$$\frac{\begin{array}{cc}\mathcal{D}'_G & \mathcal{D}_M \\ \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G & \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M\end{array}}{\Gamma_G \mid \mathrm{I}_G \vdash \mathcal{G} :: \Phi_G}\ (\textsc{Conf-C})$$

Replacing $\mathcal{D}_G$ in $\mathcal{D}$ by this derivation and reassociating gives the derivation

$$\frac{\begin{array}{cc}\dfrac{\begin{array}{cc}\mathcal{D}_F & \mathcal{D}'_G \\ \Gamma_F \mid \mathrm{I}_F \vdash \mathcal{F} :: \Phi_F & \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G\end{array}}{\Gamma''_L \mid \mathrm{I}''_L \vdash \mathcal{L}' :: \Phi''_L} & \mathcal{D}_M \\ & \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M\end{array}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}\ (\textsc{Conf-C})$$

By the subformula property, we deduce that $\Gamma''_L \mid \mathrm{I}''_L \vdash \mathcal{L}' :: \Phi''_L$ is $\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L}' :: \Phi'_L$. Let $\mathcal{D}'_L$ be the derivation for the left branch of this derivation tree.

(3) If $\mathcal{M} \subsetneq \mathcal{F}$, then by the induction hypothesis on $\mathcal{D}_F$, we get a derivation

$$\frac{\begin{array}{cc}\dfrac{\begin{array}{cc}\mathcal{D}'_F & \mathcal{D}_M \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F & \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M\end{array}}{\Gamma_F \mid \mathrm{I}_F \vdash \mathcal{F} :: \Phi_F} & \mathcal{D}_G \\ & \Gamma_G \mid \mathrm{I}_G \vdash \mathcal{G} :: \Phi_G\end{array}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

We reassociate to get the derivation

$$\frac{\begin{array}{cc}\mathcal{D}'_F & \dfrac{\begin{array}{cc}\mathcal{D}_M & \mathcal{D}_G \\ \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M & \Gamma_G \mid \mathrm{I}_G \vdash \mathcal{G} :: \Phi_G\end{array}}{\cdots \mathcal{M}, \mathcal{G} \cdots} \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F & \end{array}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

By proposition 5.6.6 we can swap the two premisses in the right tree because $\Phi_M \cap \Gamma_G$ and $\Gamma_M \cap \Phi_G$ are both empty. Indeed, they are disjoint because they have no channel

names in common. Reassociating gives the derivation

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}'_F \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F\end{matrix} \quad \begin{matrix}\mathcal{D}_G \\ \Gamma_G \mid \mathrm{I}_G \vdash \mathcal{G} :: \Phi_G\end{matrix}}{\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L} :: \Phi'_L} \quad \begin{matrix}\mathcal{D}_M \\ \Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M\end{matrix}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

We recognize that $\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L} :: \Phi'_L$ is $\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L$ by the subformula property. Let $\mathcal{D}'_L$ be the derivation of the left branch, and $\mathcal{D}_M$ be as given.

(4) If $\mathcal{M}_F = \mathcal{M} \cap \mathcal{F} = \mathcal{R} \cap \mathcal{F}$ and $\mathcal{M}_G = \mathcal{M} \cap \mathcal{G} = \mathcal{R} \cap \mathcal{G}$ has a non-empty intersection with both branches of the derivation, then we assume first $\mathcal{F} \setminus \mathcal{M}_F$ and $\mathcal{G} \setminus \mathcal{M}_G$ are both non-empty. We apply the induction hypothesis to both branches to get a derivation

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}'_F \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F\end{matrix} \quad \begin{matrix}\mathcal{D}_{MF} \\ \Gamma_{MF} \mid \mathrm{I}_{MF} \vdash \mathcal{M}_F :: \Phi_{MF}\end{matrix}}{\Gamma_F \mid \mathrm{I}_F \vdash \mathcal{F} :: \Phi_F} \quad \dfrac{\begin{matrix}\mathcal{D}'_G \\ \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G\end{matrix} \quad \begin{matrix}\mathcal{D}_{MG} \\ \Gamma_{MG} \mid \mathrm{I}_{MG} \vdash \mathcal{M}_G :: \Phi_{MG}\end{matrix}}{\Gamma_G \mid \mathrm{I}_G \vdash \mathcal{G} :: \Phi_G}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

Reassociating twice gives the derivation:

$$\dfrac{\begin{matrix}\mathcal{D}'_F \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F\end{matrix} \quad \dfrac{\dfrac{\begin{matrix}\mathcal{D}_{MF} \\ \Gamma_{MF} \mid \mathrm{I}_{MF} \vdash \mathcal{M}_F :: \Phi_{MF}\end{matrix} \quad \begin{matrix}\mathcal{D}'_G \\ \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G\end{matrix}}{\cdots \mathcal{M}_F, \mathcal{G}' \cdots} \quad \begin{matrix}\mathcal{D}_{MG} \\ \Gamma_{MG} \mid \mathrm{I}_{MG} \vdash \mathcal{M}_G :: \Phi_{MG}\end{matrix}}{\cdots \mathcal{G}, \mathcal{M}_F \cdots}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

By proposition 5.6.6 we can swap the two premisses in the middle tree because $\Phi_{MF} \cap \Gamma'_G$ and $\Gamma_{MF} \cap \Phi'_G$ are both empty. Indeed,

$$\begin{aligned}\Phi_{MF} &= \Phi_F \cap (\mathrm{I}_R \cup \Phi_R) \\ \Gamma'_G &= \Gamma_G \setminus \Gamma_R \\ \Gamma_{MF} &= \Gamma_R \cap (\Gamma_F \cup \mathrm{I}_F) \\ \Phi'_G &= (\Phi_G \setminus (\mathrm{I}_R \cup \Phi_R)) \cup (\mathrm{I}_G \cap \Gamma_R)\end{aligned}$$

and the two intersections are empty because

$$\Phi_F \cap \Gamma_G = \Gamma_F \cap \Phi_G = \mathrm{I}_F \cap \Phi_G = \mathrm{I}_G \cap \mathrm{I}_F = \varnothing.$$

Swapping the premisses gives:

$$\dfrac{\begin{matrix}\mathcal{D}'_F \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F\end{matrix} \quad \dfrac{\dfrac{\begin{matrix}\mathcal{D}'_G \\ \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G\end{matrix} \quad \begin{matrix}\mathcal{D}_{MF} \\ \Gamma_{MF} \mid \mathrm{I}_{MF} \vdash \mathcal{M}_F :: \Phi_{MF}\end{matrix}}{\cdots \mathcal{M}_F, \mathcal{G}' \cdots} \quad \begin{matrix}\mathcal{D}_{MG} \\ \Gamma_{MG} \mid \mathrm{I}_{MG} \vdash \mathcal{M}_G :: \Phi_{MG}\end{matrix}}{\cdots \mathcal{G}, \mathcal{M}_F \cdots}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

Finally, reassociating twice gives the derivation:

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}'_F \\ \Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F\end{matrix} \quad \begin{matrix}\mathcal{D}'_G \\ \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G\end{matrix}}{\cdots \mathcal{F}', \mathcal{G}' \cdots} \quad \dfrac{\begin{matrix}\mathcal{D}_{MF} \\ \Gamma_{MF} \mid \mathrm{I}_{MF} \vdash \mathcal{M}_F :: \Phi_{MF}\end{matrix} \quad \begin{matrix}\mathcal{D}_{MG} \\ \Gamma_{MG} \mid \mathrm{I}_{MG} \vdash \mathcal{M}_G :: \Phi_{MG}\end{matrix}}{\cdots \mathcal{M}_F, \mathcal{M}_G \cdots}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

Applying the subformula property and observing that $\mathcal{M} = \mathcal{M}_F, \mathcal{M}_G$ and $\mathcal{L}' = \mathcal{F}', \mathcal{G}'$, we recognize the above derivation as:

$$\dfrac{\dfrac{\mathcal{D}'_F \qquad \mathcal{D}'_G}{\Gamma'_F \mid \mathrm{I}'_F \vdash \mathcal{F}' :: \Phi'_F \quad \Gamma'_G \mid \mathrm{I}'_G \vdash \mathcal{G}' :: \Phi'_G} \qquad \dfrac{\mathcal{D}_{MF} \qquad \mathcal{D}_{MG}}{\Gamma_{MF} \mid \mathrm{I}_{MF} \vdash \mathcal{M}_F :: \Phi_{MF} \quad \Gamma_{MG} \mid \mathrm{I}_{MG} \vdash \mathcal{M}_G :: \Phi_{MG}}}{\Gamma'_L \mid \mathrm{I}'_L \vdash \mathcal{L}' :: \Phi'_L} \qquad \dfrac{}{\Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M}}{\Gamma_L \mid \mathrm{I}_L \vdash \mathcal{L} :: \Phi_L}$$

Let $\mathcal{D}'_L$ and $\mathcal{D}_M$ respectively be the left and right derivations. If one of $\mathcal{F} \setminus \mathcal{M}_F$ or $\mathcal{G} \setminus \mathcal{M}_G$ is empty, then applying the induction hypothesis to the other branch and reassociating gives the result. If both are empty, then $\mathcal{M} = \mathcal{L}$, a contradiction.

This completes the proof for the rule on the left of the statement.

We now consider the rule on the right of the statement. The same argument as for the rule on left gives that

$$\dfrac{\Gamma_M \mid \mathrm{I}_M \vdash \mathcal{M} :: \Phi_M \quad \Gamma'_R \mid \mathrm{I}'_R \vdash \mathcal{R}' :: \Phi'_R}{\Gamma \mid \mathrm{I} \vdash \mathcal{R} :: \Phi}$$

is a valid instance of (Conf-C). Next, we check that its conclusion is $\Gamma_R \mid \mathrm{I}_R \vdash \mathcal{R} :: \Phi_R$. But this is follows by symmetry with the previous rule, noticing that the derivations are identical, except that we exchange every appearance of a Φ and a Γ, and every L and R. For example, the derivation to check that $\Gamma = \Gamma_R$ in this rule is identical to the derivation to check that $\Phi = \Phi_L$ in the previous rule:

$$\begin{aligned}\Gamma'_R \setminus \Phi_M &= ((\Gamma_R \setminus (\Gamma_L \cup \mathrm{I}_L)) \cup (\Phi_L \cap \mathrm{I}_R)) \setminus (\Phi_L \cap (\mathrm{I}_R \cup \Phi_R)) \\ &= (\Gamma_R \setminus (\Gamma_L \cup \mathrm{I}_L)) \setminus (\Phi_L \cap (\mathrm{I}_R \cup \Phi_R))\end{aligned}$$

because $\Gamma_R \cap (\Phi_R \cup \mathrm{I}_R) = \varnothing$:

$$\begin{aligned}&= \Gamma_R \setminus (\mathrm{I}_L \cup \Gamma_L).\\ \Gamma = \Gamma_M \cup (\Gamma'_R \setminus \Phi_M) &= (\Gamma_R \cap (\Gamma_L \cup \mathrm{I}_L)) \cup (\Gamma_R \setminus (\mathrm{I}_L \cup \Gamma_L)) \\ &= \Gamma_R.\end{aligned}$$

The check that $\mathrm{I} = \mathrm{I}_R$ and $\Phi = \Phi_R$ is analogous. The construction of the desired derivations is also an analogously straightforward adaptation of the construction for the previous rule. □

There are two remaining ingredients to showing that we can replace subsets of well-typed configurations without affecting the interface of the whole configuration, provided the replaced multiset and replacement multiset share the same interface. The first involves showing that given a hypothetical derivation, we can freely replace the hypothesis, provided that the new hypothesis has the same interface and that its internal channels do not conflict with those in the conclusion. It is given by lemma 5.6.16. The second involves showing that we can always reassociate a configuration so that a well-typed multiset appears as a conclusion in the configuration's typing derivation, i.e., showing that we can a always reassociate a configuration so that we can apply lemma 5.6.16 to replace the multiset of interest. We do so using "LMR derivations", introduced below.

Lemma 5.6.16. *Assume* $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta' \blacktriangleright \Gamma \mid \mathrm{I}_E \mathrm{I}_F \vdash \mathcal{E}, \mathcal{F} :: \Theta$. *If* $\Gamma' \mid \mathrm{I}_G \vdash \mathcal{G} :: \Theta'$ *is well-formed and* $\check{\mathrm{I}}_G$ *is disjoint from* $\check{\Gamma}, \check{\mathrm{I}}_E, \check{\Theta}$, *then* $\Gamma' \mid \mathrm{I}_G \vdash \mathcal{G} :: \Theta' \blacktriangleright \Gamma \mid \mathrm{I}_E \mathrm{I}_G \vdash \mathcal{E}, \mathcal{G} :: \Theta$.

Proof. By induction on the hypothetical derivation $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta' \blacktriangleright \Gamma \mid \mathrm{I}_E \mathrm{I}_F \vdash \mathcal{E}, \mathcal{F} :: \Theta$, where we replace the hypothesis $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta'$ with the hypothesis $\Gamma' \mid \mathrm{I}_G \vdash \mathcal{G} :: \Theta'$. □

It is often useful to reassociate a typing derivation such that a given subset appears as a conclusion in the derivation. Assume $\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi$ and $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$ with $\Gamma \subseteq \Psi$, $\Gamma', \mathrm{I}_M, \Delta' \subseteq \mathrm{I}$, and $\Delta \subseteq \Xi$. An **LMR derivation** (left-middle-right derivation) of $\Psi \mid \mathrm{I}_E \mathrm{I}_M \vdash \mathcal{E}, \mathcal{M} :: \Xi$ for $\mathcal{M}$ is a derivation of $\Psi \mid \mathrm{I}_E \mathrm{I}_M \vdash \mathcal{E}, \mathcal{M} :: \Xi$ that decomposes $\mathcal{E}$ into (potentially empty) multisets

$\mathcal{L}$ and $\mathcal{R}$ such that $\mathcal{L}$ and $\mathcal{R}$ do not share any channels, and all channels that $\mathcal{L}$ provides are used by $\mathcal{M}$. Explicitly,

(1) if Γ' is non-empty and $\Delta \neq \Xi$, then $\mathcal{L}$ and $\mathcal{R}$ are both non-empty and the LMR derivation is of the form:

$$\dfrac{\dfrac{\Lambda \vdash \mathcal{L} :: \Gamma' \quad \Gamma'\Gamma \vdash \mathcal{M} :: \Delta\Delta'}{\Lambda\Gamma \vdash \mathcal{L}, \mathcal{M} :: \Delta\Delta'}\ (\textsc{Conf-C}) \quad \mathrm{P}\Delta' \vdash \mathcal{R} :: \Phi}{\Psi \mid \mathrm{I}_E\, \mathrm{I}_M \vdash \mathcal{E}, \mathcal{M} :: \Xi}\ (\textsc{Conf-C})$$

(2) if Γ' is non-empty and $\Delta = \Xi$, then $\mathcal{L}$ is non-empty, $\mathcal{R}$ is empty, and the LMR derivation is of the form:

$$\dfrac{\Lambda \vdash \mathcal{L} :: \Gamma' \quad \Gamma\Gamma' \vdash \mathcal{M} :: \Delta}{\Psi \mid \mathrm{I}_E\, \mathrm{I}_M \vdash \mathcal{E}, \mathcal{M} :: \Xi}\ (\textsc{Conf-C})$$

(3) if Γ' is empty and $\Delta \neq \Xi$, then $\mathcal{L}$ is empty, $\mathcal{R}$ is non-empty, and the LMR derivation is of the form:

$$\dfrac{\Gamma \vdash \mathcal{M} :: \Delta\Delta' \quad \mathrm{P}\Delta' \vdash \mathcal{R} :: \Phi}{\Psi \mid \mathrm{I}_E\, \mathrm{I}_M \vdash \mathcal{E}, \mathcal{M} :: \Xi}\ (\textsc{Conf-C})$$

(4) If Γ' is empty and $\Delta = \Xi$, then $\mathcal{E}$ is empty, so so are $\mathcal{L}$ and $\mathcal{R}$, and the LMR derivation is $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$.

Proposition 5.6.17. *Assume* $\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi$ *and* $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$ *with* $\Gamma \subseteq \Psi$, $\Gamma', \mathrm{I}_M, \Delta' \subseteq \mathrm{I}$, *and* $\Delta \subseteq \Xi$. *There exists an LMR derivation of* $\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi$ *for* $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$.

Proof. We proceed by induction on the derivation of $\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi$.

Case (Conf-M)**:** Then $\mathcal{E}$ must be empty, and the LMR derivation is $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$.

Case (Conf-P)**:** Analogous to the case (Conf-M).

Case (Conf-C)**:** Recall the rule schema:

$$\dfrac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi\, \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\textsc{Conf-C})$$

Assume first that $\mathcal{M}$ is contained in the left premise. By the induction hypothesis, there is an LMR derivation of that premise for $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$. If $\mathcal{L}$ and $\mathcal{R}$ are both empty, then we are done, for the left premise is $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$. If $\mathcal{L}$ is non-empty but $\mathcal{R}$ is empty, then we are done, for we have an LMR derivation. If $\mathcal{L}$ is empty and $\mathcal{R}$ is non-empty, then we are in the situation

$$\dfrac{\dfrac{\cdots\mathcal{M}\cdots \quad \cdots\mathcal{R}\cdots}{\cdots\mathcal{M}, \mathcal{R}\cdots} \quad \cdots\mathcal{R}'\cdots}{\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi}$$

where we use three dots "$\cdots$" to elide the unique conclusion given by functionality of composition (proposition 5.6.6). We can rotate this derivation counter-clockwise to get the desired LMR derivation:

$$\dfrac{\cdots\mathcal{M}\cdots \quad \dfrac{\cdots\mathcal{R}\cdots \quad \cdots\mathcal{R}'\cdots}{\cdots\mathcal{R}, \mathcal{R}'\cdots}}{\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi}$$

If $\mathcal{L}$ and $\mathcal{R}$ are both non-empty, then we are in the situation

$$\dfrac{\dfrac{\dfrac{\cdots\mathcal{L}\cdots \quad \cdots\mathcal{M}\cdots}{\cdots\mathcal{L}, \mathcal{M}\cdots} \quad \cdots\mathcal{R}\cdots}{\cdots\mathcal{L}, \mathcal{M}, \mathcal{R}\cdots} \quad \cdots\mathcal{R}'\cdots}{\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi}$$

Then we can rotate this derivation counter-clockwise to get the desired LMR derivation:

$$\dfrac{\dfrac{\cdots\mathcal{L}\cdots \quad \cdots\mathcal{M}\cdots}{\cdots\mathcal{L}, \mathcal{M}\cdots} \quad \dfrac{\cdots\mathcal{R}\cdots \quad \cdots\mathcal{R}'\cdots}{\cdots\mathcal{R}, \mathcal{R}'\cdots}}{\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi}$$

Assume next that $\mathcal{M}$ is contained in the right premise. By the induction hypothesis, there is an LMR derivation of that premise for $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$. The analysis is analogous, except in the case when $\mathcal{L}$ and $\mathcal{R}$ given by the induction hypothesis are both non-empty: in this case, we will need two clockwise rotations.

Composing that hypothetical derivation with (CONF-C) and the other premise gives the desired hypothetical derivation.

Assume first that $\mathcal{M}$ has a non-empty intersection with both premises. Call the intersections with the left and right premises $\mathcal{M}_L$ and $\mathcal{M}_R$ respectively (so $\mathcal{M} = \mathcal{M}_L, \mathcal{M}_R$), and let their respective complements in $\mathcal{C}$ and $\mathcal{D}$ be $\mathcal{L}$ and $\mathcal{R}$. Assume that $\mathcal{L}$ and $\mathcal{R}$ are both non-empty. Then by proposition 5.6.15 there exists a hypothetical derivation

$$\dfrac{\dfrac{\cdots\mathcal{L}\cdots \quad \cdots\mathcal{M}_L\cdots}{\cdots\mathcal{C}\cdots} \quad \dfrac{\cdots\mathcal{M}_R\cdots \quad \cdots\mathcal{R}\cdots}{\cdots\mathcal{D}\cdots}}{\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi}$$

By proposition 5.6.7, we can twice rotate this derivation to get the following derivation.

$$\dfrac{\dfrac{\cdots\mathcal{L}\cdots \quad \dfrac{\cdots\mathcal{M}_L\cdots \quad \cdots\mathcal{M}_R\cdots}{\cdots\mathcal{M}\cdots}}{\cdots\mathcal{L},\mathcal{M}\cdots} \quad \cdots\mathcal{R}\cdots}{\Psi \mid \mathrm{I} \vdash \mathcal{E}, \mathcal{M} :: \Xi}$$

We know that the interface for $\mathcal{M}$ in the derivation remains $\Gamma\Gamma' \mid \mathrm{I}_M \vdash \mathcal{M} :: \Delta'\Delta$ by the subformula property (proposition 5.6.8). This gives the desired LMR derivation. If either $\mathcal{L}$ or $\mathcal{R}$ is empty, then applying the induction hypothesis to the other branch and rotating the tree will give the result. If both are empty, then the LMR derivation is just the derivation of $\mathcal{M}$. □

The following proposition, combined with lemma 5.6.16, shows that we can replace any subset of a multiset with one that has the same interface:

PROPOSITION 5.6.18 (Replacement Property). *Assume $\mathcal{F}$ is non-empty. If $\Gamma \mid \mathrm{I}_E\,\mathrm{I}_F \vdash \mathcal{E}, \mathcal{F} :: \Theta$ and $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta'$ with $\Gamma' \subseteq \Gamma\,\mathrm{I}_E$ and $\Theta' \subseteq \mathrm{I}_E\Theta$, then there exists a hypothetical derivation $\Gamma' \mid \mathrm{I}_3 \vdash \mathcal{G} :: \Theta' \blacktriangleright \Gamma \mid \mathrm{I}_E\,\mathrm{I}_3 \vdash \mathcal{E}, \mathcal{G} :: \Theta$ for all $\mathcal{G}$ and I_3 for which the conclusion is well-formed.*

Proof. If $\mathcal{E}$ is empty, then $\Gamma \mid \mathrm{I}_E\,\mathrm{I}_F \vdash \mathcal{E}, \mathcal{F} :: \Theta$ is $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta'$ and I_3 is empty, and we are done by the fact that hypothetical derivability is reflexive. Assume now that $\mathcal{E}$ is not empty.

By proposition 5.6.17, there exists an LMR derivation for $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta'$ in $\Gamma \mid \mathrm{I}_E\,\mathrm{I}_F \vdash \mathcal{E}, \mathcal{F} :: \Theta$. Pruning the LMR derivation at $\Gamma' \mid \mathrm{I}_F \vdash \mathcal{F} :: \Theta'$ gives the hypothetical derivation. The result follows by lemma 5.6.16. □

Every configuration can be decomposed as the composition of independent "simply branched" configurations with no common channels. The absence of common channels implies that these simply branched subconfigurations do not interact during executions. We will use this fact to reduce proofs about arbitrary configurations to proofs about simply branched configurations.

Definition 5.6.19. A configuration $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ is **simply branched** if it has a derivation in which every instance of the rule

$$\dfrac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \qquad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1\Pi\,\mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\textsc{Conf-C})$$

has exactly one channel in the context Π. ◂

PROPOSITION 5.6.20. *Every configuration $\Gamma \vdash \mathcal{C} :: d_0 : D_0, \ldots, d_n : D_n$ is the composition $\Gamma_0, \ldots, \Gamma_n \mid \mathrm{I}_0, \ldots, \mathrm{I}_n \vdash \mathcal{C}_0, \ldots, \mathcal{C}_n :: d_0 : D_0, \ldots, d_n : D_n$ of simply-branched configurations $\Gamma_i \mid \mathrm{I}_i \vdash \mathcal{C}_i :: d_i : D_i$ for $0 \le i \le n$.*

Proof. We proceed by induction on the derivation of $\Gamma \vdash \mathcal{C} :: d_0 : D_0, \ldots, d_n : D_n$.

CASE (CONF-M)**:** Immediate.

Case (Conf-P): Immediate.

Case (Conf-C): By assumption, both branches of the rule

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi} \text{ (Conf-C)}$$

can be decomposed into the composition of simply-branched configurations. Iterating the composition using an induction on the number of channels in the intersection Π gives the result. Indeed, if there are no channels in the intersection Π, then we are done. Assume the result for some n, and assume Π has $n+1$ channels. Leave one of the configurations of the left branch's decomposition out. By the induction hypothesis, the composition of the remainder can be given the desired decomposition. Applying (Conf-C) to the branch left-out and the decomposition given by the induction hypothesis gives the result. □

Finally, we can characterize simply branched configurations by looking at their provided channels:

Proposition 5.6.21. *A configuration $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ is simply branched if and only if Δ contains exactly one channel.*

Proof. Assume first that $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ is simply branched. We proceed by induction on one of its simply-branched derivations to show that Δ contains exactly one channel.

Case (Conf-M): Immediate.

Case (Conf-P): Immediate.

Case (Conf-C): By assumption, both branches of the rule

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi} \text{ (Conf-C)}$$

are simply branched. By the induction hypothesis, this means that $\Phi\Pi$ contains exactly one channel, and by assumption, that channel must be contained in Π. So Φ is empty. By the induction hypothesis, we also have that Ξ contains exactly one channel. Because $\Delta = \Phi\Xi$, we conclude that Δ also contains exactly one channel.

The converse is given by proposition 5.6.20. □

5.7. Type-Indexed Relations

Our ultimate goal is to relate programs that are equivalent or that somehow approximate each other. We define various desirable properties for relations on programs and configurations.

Polarized SILL and its configurations do not have unicity of typing, and processes could be equivalent at one type but not at another. Accordingly, we would like our relations to be type-indexed:

Definition 5.7.1. **Type-indexed (binary) relations** are families of relations indexed by typing sequents. Explicitly:

(1) A **type-indexed relation $\mathfrak{R}$ on configurations** is a family of relations $(\mathfrak{R}_{\Delta \vdash \Phi})_{\Delta,\Phi}$ where $(\mathcal{C}, \mathcal{D}) \in \mathfrak{R}_{\Delta \vdash \Phi}$ only if $\Delta \vdash \mathcal{C} :: \Phi$ and $\Delta \vdash \mathcal{D} :: \Phi$. In this case, we write $\Delta \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{D} :: \Phi$.

(2) A **type-indexed relation $\mathfrak{R}$ on processes** is a family of relations $(\mathfrak{R}_{\Psi;\Delta \vdash c:A})_{\Psi,\Delta,c:A}$ where $(P, Q) \in \mathfrak{R}_{\Psi,\Delta \vdash c:A}$ only if $\Psi\ ;\ \Delta \vdash P :: c : A$ and $\Psi\ ;\ \Delta \vdash Q :: c : A$. In this case, we write $\Psi\ ;\ \Delta \vdash P\ \mathfrak{R}\ Q :: c : A$.

(3) A **type-indexed relation $\mathfrak{R}$ on terms** is a family of relations $(\mathfrak{R}_{\Psi \vdash \tau})_{\Psi,\tau}$ where $(M, N) \in \mathfrak{R}_{\Psi \vdash \tau}$ only if $\Psi \Vdash M : \tau$ and $\Psi \Vdash N : \tau$. In this case, we write $\Psi \Vdash M\ \mathfrak{R}\ N : \tau$.

Type-indexed relations are assumed to satisfy the exchange, renaming, and weakening structural properties whenever their underlying judgments do. The **renaming** property for type-indexed relations on configurations is subtle because we elided internal channels. Explicitly, it is the property:

- If $\Gamma \vdash \mathcal{C} \mathrel{\mathfrak{R}} \mathcal{D} :: \Delta$, $\Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Delta$, and $\Gamma \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Delta$, and $\sigma\colon \Gamma\Delta \leftrightarrow \Gamma'\Delta'$, $\sigma_1\colon \mathrm{I}_1 \leftrightarrow \mathrm{I}'_1$, and $\sigma_2\colon \mathrm{I}_2 \leftrightarrow \mathrm{I}'_2$ are renamings, then $\Gamma' \vdash [\sigma, \sigma_1]\mathcal{C} \mathrel{\mathfrak{R}} [\sigma, \sigma_2]\mathcal{D} :: \Delta'$. ◂

We will study the effects of running "equivalent programs" in various program contexts. Contexts are programs with holes:

Definition 5.7.2. A **(typed) term context** $\Psi \Vdash C[\cdot]^{\Gamma}_{\sigma} : \tau$ is a term formed by the rules of section 5.A.1 plus one instance of the axiom (F-Hole):

$$\frac{}{\Gamma \Vdash [\cdot]^{\Gamma}_{\sigma} : \sigma} \text{ (F-Hole)}$$

Given a term context $\Psi \Vdash C[\cdot]^{\Gamma}_{\sigma} : \tau$ and a term $\Gamma \Vdash M : \sigma$, the result of "plugging" M into the hole is the term $\Psi \Vdash C[M]^{\Gamma}_{\sigma} : \tau$ obtained by replacing the axiom (F-Hole) by the derivation of $\Gamma \Vdash M : \sigma$. ◂

Definition 5.7.3. A **(typed) process context** $\Psi \mathbin{;} \Delta \vdash C[\cdot]^{\Gamma;\Lambda}_{b:B} :: a : A$ is a process formed by the rules of section 5.A.2 plus one instance of the axiom (P-Hole):

$$\frac{}{\Gamma \mathbin{;} \Lambda \vdash [\cdot]^{\Gamma;\Lambda}_{b:B} :: b : B} \text{ (P-Hole)}$$

Given a process context $\Psi \mathbin{;} \Delta \vdash C[\cdot]^{\Gamma;\Lambda}_{b:B} :: a : A$ and a process $\Gamma \mathbin{;} \Lambda \vdash P :: b : B$, the result of "plugging" P into the hole is the term $\Psi \mathbin{;} \Delta \vdash C[P]^{\Gamma;\Lambda}_{b:B} :: a : A$ obtained by replacing the axiom (P-Hole) by the derivation of $\Gamma \mathbin{;} \Lambda \vdash P :: b : B$. ◂

We most often work only with closed terms and processes, and listing empty functional contexts becomes tiresome. Consequently, we write $C[\cdot]^{\Lambda}_{b:B}$ and $C[\cdot]_{\sigma}$ for $C[\cdot]^{\cdot;\Lambda}_{b:B}$ and $C[\cdot]^{\cdot}_{\sigma}$, respectively.

Definition 5.7.4. A **(typed) configuration context** $\Gamma \mid \mathrm{I} \vdash \mathcal{C}[\cdot]^{\Lambda}_{\Xi} :: \Delta$ is a configuration formed by the rules of section 5.2 plus one instance of the axiom (Conf-H):

$$\frac{}{\Lambda \mid \cdot \vdash [\cdot]^{\Lambda}_{\Xi} :: \Xi} \text{ (Conf-H)}$$

Consider a configuration context $\Gamma \mid \mathrm{I} \vdash \mathcal{C}[\cdot]^{\Lambda}_{\Xi} :: \Delta$ and a configuration $\Lambda \mid \mathrm{I}' \vdash \mathcal{D} :: \Xi$ such that $\check{\mathrm{I}}'$ is disjoint from $\check{\Gamma}$, $\check{\mathrm{I}}$, $\check{\Delta}$. The result of "plugging" $\mathcal{D}$ into the hole is the configuration $\Gamma \mid \mathrm{I}, \mathrm{I}' \vdash \mathcal{C}[\mathcal{D}]^{\Lambda}_{\Xi} :: \Delta$ given by lemma 5.6.16, where we replace the axiom (Conf-H) by the derivation of $\Lambda \mid \mathrm{I}' \vdash \mathcal{D} :: \Xi$ and thread the added internal channels I' through the derivation. ◂

Remark 5.7.5. We can always plug a configuration in a hole with a matching interface in definition 5.7.4 by suitably renaming the internal channel names in I'.

Definition 5.7.6. A type-indexed relation is **contextual** if it is closed under contexts. Explicitly:

(1) A type-indexed relation $\mathfrak{R}$ on configurations is contextual if $\Lambda \vdash \mathcal{C} \mathrel{\mathfrak{R}} \mathcal{D} :: \Xi$ implies that $\Gamma \vdash \mathcal{E}[\mathcal{C}]^{\Lambda}_{\Xi} \mathrel{\mathfrak{R}} \mathcal{E}[\mathcal{D}]^{\Lambda}_{\Xi} :: \Delta$ for all $\Gamma \vdash \mathcal{E}[\cdot]^{\Lambda}_{\Xi} :: \Delta$.

(2) A type-indexed relation $\mathfrak{R}$ on processes is contextual if $\Psi \mathbin{;} \Delta \vdash P \mathrel{\mathfrak{R}} Q :: c : A$ implies that $\Psi' \mathbin{;} \Delta' \vdash C[P]^{\Psi;\Delta}_{c:A} \mathrel{\mathfrak{R}} C[Q]^{\Psi;\Delta}_{c:A} :: b : B$ for all $\Psi' \mathbin{;} \Delta' \vdash C[\cdot]^{\Psi;\Delta}_{c:A} :: b : B$.

(3) A type-indexed relation $\mathfrak{R}$ on terms is contextual if $\Psi \Vdash M \mathrel{\mathfrak{R}} N : \tau$ implies that $\Psi' \Vdash C[M]^{\Psi}_{\tau} \mathrel{\mathfrak{R}} C[N]^{\Psi}_{\tau} : \tau'$ for all $\Psi' \Vdash C[\cdot]^{\Psi}_{\tau} : \tau'$. ◂

Definition 5.7.7. The **contextual interior** $\mathfrak{R}^c$ of a type-indexed relation $\mathfrak{R}$ is the greatest contextual type-indexed relation contained in $\mathfrak{R}$. ◂

Lemma 5.7.8. *Taking the contextual interior of a relation is a monotone operation, and it preserves arbitrary intersections.*

Simply branched configuration contexts closely mirror the "observation contexts" used to observe processes in section 7.5. The concept of a simply branched context is subtle: given a simply branched configuration context, we would like the result of filling its hole to again be simply branched. However, this need not always be the case: the context $[\cdot]^{\Gamma}_{a:A,b:B}$ satisfies definition 5.6.19, but proposition 5.6.21 implies that for no $\mathcal{C}$ is $[\mathcal{C}]^{\Gamma}_{a:A,b:B}$ simply branched. Instead, we use the characterization of proposition 5.6.21 to define simply branched configuration contexts:

Definition 5.7.9. A configuration context $\Lambda \vdash \mathcal{B}[\cdot]^{\Gamma}_{\Delta} :: \Xi$ is **simply branched** if Ξ contains exactly one channel. ◂

Definition 5.7.10. A typed relation $\mathfrak{R}$ on configurations is **simply branched contextual** if $\Gamma \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{D} :: \Delta$ implies that $\Phi \vdash \mathcal{B}[\mathcal{C}]^{\Gamma}_{\Delta}\ \mathfrak{R}\ \mathcal{B}[\mathcal{D}]^{\Gamma}_{\Delta} :: c : C$ for all simply branched contexts $\Phi \vdash \mathcal{B}[\cdot]^{\Gamma}_{\Delta} :: c : C$. ◂

Definition 5.7.11. The **simply branched contextual interior** $\mathfrak{R}^b$ of a typed relation $\mathfrak{R}$ on configurations is the greatest simply branched contextual typed relation contained in $\mathfrak{R}$. ◂

For most of the relations $\mathfrak{R}$ considered below, $\mathfrak{R}^b$ and $\mathfrak{R}^c$ coincide. This fact, reminiscent of Milner's context lemma [Mil77], reduces our proof burden when showing that configurations are related by contextual preorders: we only need to quantify over simply branched contexts instead of over all contexts. Contextual preorders are called precongruences:

Definition 5.7.12. A typed relation $\mathfrak{R}$ on configurations is a **precongruence** if:

(1) each relation in the family is a preorder; and
(2) the relation respects composition: if $\Gamma \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{C}' :: \Phi\Sigma$ and $\Sigma\Lambda \vdash \mathcal{D}\ \mathfrak{R}\ \mathcal{D}' :: \Xi$, then $\Gamma\Lambda \vdash \mathcal{C}, \mathcal{D}\ \mathfrak{R}\ \mathcal{C}', \mathcal{D}' :: \Phi\Xi$.

It is a **congruence** if it is also an equivalence relation. ◂

Congruence relations are desirable because they let us "replace equals by equals". The following proposition is standard:

Proposition 5.7.13. *A typed equivalence relation $\mathfrak{R}$ on configurations is a precongruence if and only if it is a contextual preorder.*

Proof. It is obvious that every precongruence is contextual. To show that every contextual preorder is a precongruence, assume $\Gamma \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{C}' :: \Phi\Sigma$ and $\Sigma\Lambda \vdash \mathcal{D}\ \mathfrak{R}\ \mathcal{D}' :: \Xi$. By contextuality, $\Gamma\Lambda \vdash \mathcal{C}, \mathcal{D}\ \mathfrak{R}\ \mathcal{C}', \mathcal{D} :: \Phi\Xi$ and $\Gamma\Lambda \vdash \mathcal{C}', \mathcal{D}\ \mathfrak{R}\ \mathcal{C}', \mathcal{D}' :: \Phi\Xi$. By transitivity, $\Gamma\Lambda \vdash \mathcal{C}, \mathcal{D}\ \mathfrak{R}\ \mathcal{C}', \mathcal{D}' :: \Phi\Xi$. □

We can use contextual interiors and proposition 5.7.13 to extract precongruences from preorders (cf. [Mil80, Theorem 7.5]). Proposition 5.7.13 and the definitions of precongruence and congruence translate from configurations to processes and terms in the obvious way.

5.8. Dynamic Properties of Terms

The following preservation result for the functional layer and its proof are standard:

Proposition 5.8.1 (Preservation). *If $\cdot \Vdash M : \tau$ and $M \Downarrow v$, then $\cdot \Vdash M : v$.*

Proposition 5.8.2 (Canonical Forms). *If M val, then*

(1) if $\cdot \Vdash M : \tau \to \tau'$, then M is $\lambda x : \tau.M'$ for some term M';
(2) if $\cdot \Vdash M : \{c_o : A_o \leftarrow \overline{c_i : A_i}\}$, then M is $c_o \leftarrow \{P\} \leftarrow \overline{c}_i$ for some process P.

Proof. By case analysis on M val and inversion on the typing judgment. □

5.9. Dynamic Properties of Typed Configurations

In this section, we prove two important properties about Polarized SILL. The first, in section 5.9.1, is that the substructural operational semantics for Polarized SILL enjoys a type preservation property. The second, in section 5.9.2, is that all well-typed processes and configurations have fair executions.

5.9.1. Preservation. Let $\mathcal{P}$ be MRS for Polarized SILL, i.e., the MRS given by the rules of section 5.B. We prove various invariants maintained by process traces and traces from well-typed configurations. Our first goal is to show that the substructural operational semantics preserves interfaces, and that it does not change the types of internal channel names. To do so, we use the fact that multiset rewriting only makes local changes to a multiset, and these local changes do not affect the type of a multiset. In particular, we show that whenever a rule replaces the active multiset with a new multiset, then the new multiset has the same interface as the active multiset. Moreover, the type of the stationary multiset remains unchanged.

We formulate our preservation result in terms of configuration contexts. In particular, our formulation makes explicit the fact that the type of the stationary multiset (seen as a configuration context) does not change, and the fact that the active multiset and its replacement have the same interface.

Proposition 5.9.1 (Preservation). *Assume* $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$. *If* $\Sigma\,; \mathcal{C} \to \Sigma'\,; \mathcal{C}'$ *by some rule instance* $\mathcal{E} \to \mathcal{E}'$, *then there exist* $\Psi \subseteq \Gamma\,\mathrm{I}$, $\mathrm{I}_L \subseteq \mathrm{I}$, *and* $\Theta \subseteq \mathrm{I}\Delta$ *such that*

(1) $\Psi \mid \mathrm{I}_L \vdash \mathcal{E} :: \Theta$,
(2) $\Psi \mid \mathrm{I}_R \vdash \mathcal{E}' :: \Theta$ *for some* I_R *whose channel names are disjoint from those in* $\Gamma\,\mathrm{I}_L\,\Delta$,
(3) $\mathcal{C}$ *is given by* $\Sigma \parallel \Gamma \mid \mathrm{I}_L\,\mathrm{I}' \vdash \mathcal{D}[\mathcal{E}]^{\Psi}_{\Theta} :: \Delta$ *for some configuration context* $\Gamma \mid \mathrm{I}' \vdash \mathcal{D}[\cdot]^{\Psi}_{\Theta} :: \Delta$ *and some* I', *and*
(4) $\mathcal{C}'$ *is given by* $\Sigma, \check{\mathrm{I}}_R \parallel \Gamma \mid \mathrm{I}_R\,\mathrm{I}' \vdash \mathcal{D}[\mathcal{E}']^{\Psi}_{\Theta} :: \Delta$ *and* $\Sigma' = \Sigma, \check{\mathrm{I}}_R$.

Proof. We will proceed by case analysis on the rule below. By lemma 5.6.14, there exist $\Psi \subseteq \Gamma\,\mathrm{I}$, $\mathrm{I}_L \subseteq \mathrm{I}$, and $\Theta \subseteq \mathrm{I}\Delta$ such that $\Psi \mid \mathrm{I}_L \vdash \mathcal{E} :: \Theta$. By proposition 5.6.18, $\Psi \mid \mathrm{I}_L \vdash \mathcal{E} :: \Theta \triangleright \Gamma \mid \mathrm{I}_L\,\mathrm{I}' \vdash \mathcal{C} :: \Delta$. Replacing the hypothesis $\Psi \mid \mathrm{I}_L \vdash \mathcal{E} :: \Theta$ by the axiom (Conf-H) gives a configuration context $\Gamma \mid \mathrm{I}' \vdash \mathcal{D}[\cdot]^{\Psi}_{\Theta} :: \Delta$ such that $\Sigma \parallel \Gamma \mid \mathrm{I}_L\,\mathrm{I}' \vdash \mathcal{D}[\mathcal{E}]^{\Psi}_{\Theta} :: \Delta$, where $\mathcal{C} = \mathcal{D}[\mathcal{E}]^{\Psi}_{\Theta}$. In each case below, we show that there exists an I_R such that $\Psi \mid \mathrm{I}_R \vdash \mathcal{E}' :: \Theta$, and its channel names are the fresh channel names generated by the rule instance. The action of the multiset rewrite rule $\mathcal{E} \to \mathcal{E}'$ replaces $\mathcal{E}$ by $\mathcal{E}'$ in $\mathcal{C}$ to give the multiset $\mathcal{C}' = \mathcal{D}[\mathcal{E}']^{\Psi}_{\Theta}$. In particular, this implies $\Sigma, \check{\mathrm{I}}_R \parallel \Gamma \mid \mathrm{I}_R\,\mathrm{I}' \vdash \mathcal{D}[\mathcal{E}']^{\Psi}_{\Theta} :: \Delta$ and $\Sigma' = \Sigma, \check{\mathrm{I}}_R$.

We proceed by case analysis on the rule used to make the step. In each case, we freely use the fact that parametric hypothetical derivations are closed under renaming of channel names. We omit cases that follow by analogy with others.

Case (64)**:** The rule is:

$$\mathsf{msg}(a, m^+), \mathsf{proc}(b, a \to b) \to \mathsf{msg}(b, [b/a]m^+)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot\,; \Delta \vdash m^+ :: a : A}{\Delta \mid \cdot \vdash \mathsf{msg}(a, m^+) :: a : A}\ (\textsc{Conf-M}) \quad \dfrac{\dfrac{}{\cdot\,; a : A \vdash a \to b :: b : A}\ (\textsc{Fwd}^+)}{a : A \mid \cdot \vdash a \to b :: b : A}\ (\textsc{Conf-P})}{\Delta \mid a : A \vdash \mathsf{msg}(a, m^+), \mathsf{proc}(b, a \to b) :: b : A}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\dfrac{\cdot\,; \Delta \vdash [b/a]m^+ :: b : A}{\Delta \mid \cdot \vdash \mathsf{msg}(b, [b/a]m^+) :: b : A}\ (\textsc{Conf-M})$$

Both sides share the same interface, so this completes the case.

Case (65)**:** The rule is:

$$\mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m^-_{b,c}) \to \mathsf{msg}(c, [a/b]m^-_{b,c})$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\dfrac{}{\cdot\,; a : A \vdash a \leftarrow b :: b : A}\ (\textsc{Fwd}^-)}{a : A \mid \cdot \vdash \mathsf{proc}(b, a \leftarrow b) :: b : A}\ (\textsc{Conf-P}) \quad \dfrac{\cdot\,; b : A \vdash m^-_{b,c} :: c : C}{b : A \mid \cdot \vdash \mathsf{msg}(c, m^-_{b,c}) :: c : C}\ (\textsc{Conf-M})}{a : A \mid b : A \vdash \mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m^-_{b,c}) :: c : C}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\frac{\cdot\,;\, a : A \vdash [a/b]m^{-}_{b,c} :: c : C}{a : A \mid \cdot \vdash \mathsf{msg}(c, [a/b]m^{-}_{b,c}) :: c : C}\ (\textsc{Conf}\text{-M})$$

Both sides share the same interface, so this completes the case.

Case (66): The rule is:

$$\forall \Delta_1, \Delta_2, c.\mathsf{proc}(c, a \leftarrow P;\ Q) \to \exists b.\mathsf{proc}(b, [b/a]P), \mathsf{proc}(c, [b/a]Q)$$

The typing judgment for the left-hand side is, by inversion:

$$\frac{\dfrac{\cdot\,;\, \Delta_1 \vdash P :: a : A \quad \cdot\,;\, a : A, \Delta_2 \vdash Q :: c : C}{\cdot\,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C}\ (\textsc{Cut})}{\Delta_1\Delta_2 \mid \cdot \vdash \mathsf{proc}(c, a \leftarrow P;\ Q) :: c : C}\ (\textsc{Conf}\text{-P})$$

By substitution, the right-hand side is:

$$\frac{\dfrac{\cdot\,;\, \Delta_1 \vdash [b/a]P :: b : A}{\Delta_1 \mid \cdot \vdash \mathsf{proc}(b, [b/a]P) :: b : A}\ (\textsc{Conf}\text{-P}) \quad \dfrac{\cdot\,;\, b : A, \Delta_2 \vdash [b/a]Q :: c : C}{b : A, \Delta_2 \mid \cdot \vdash \mathsf{proc}(c, [b/a]Q) :: c : C}\ (\textsc{Conf}\text{-P})}{\Delta_1\Delta_2 \mid b : A \vdash \mathsf{proc}(b, [b/a]P), \mathsf{proc}(c, [b/a]Q) :: c : C}\ (\textsc{Conf}\text{-C})$$

Both sides share the same interface, so this completes the case.

Case (73): The rule is:

$$\forall a, \overline{a_i}.\mathbf{eval}(M, a \leftarrow \{P\} \leftarrow \overline{a_i}), \mathsf{proc}(a, a \leftarrow \{M\} \leftarrow \overline{a_i}) \to \mathsf{proc}(a, P)$$

The typing judgment for the left-hand side is, by inversion:

$$\frac{\dfrac{\dfrac{\cdot \Vdash M : \{a : A \leftarrow \overline{a_i : A_i}\}}{\cdot\,;\, \overline{a_i : A_i} \vdash a \leftarrow \{M\} \leftarrow \overline{a_i} :: a : A}\ (\text{E-}\{\})}{\overline{a_i : A_i} \mid \cdot \vdash \mathsf{proc}(a, a \leftarrow \{M\} \leftarrow \overline{a_i}) :: a : A}\ (\textsc{Conf}\text{-P})$$

By proposition 5.8.1, $\cdot \Vdash a \leftarrow \{P\} \leftarrow \overline{a_i} : \{a : A \leftarrow \overline{a_i : A_i}\}$. By inversion, $\cdot\,;\, \overline{a_i : A_i} \vdash P :: a : A$. The right-hand side is:

$$\frac{\cdot\,;\, \overline{a_i : A_i} \vdash P :: a : A}{\overline{a_i : A_i} \mid \cdot \vdash \mathsf{proc}(a, P) :: a : A}\ (\textsc{Conf}\text{-P})$$

Both sides share the same interface, so this completes the case.

Case (68): The rule is:

$$\forall a.\mathsf{proc}(a, \mathsf{close}\ a) \to \mathsf{msg}(a, \mathsf{close}\ a)$$

The typing judgment for the left-hand side is, by inversion:

$$\frac{\dfrac{}{\cdot\,;\, \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}}\ (\mathbf{1}\text{R})}{\cdot \mid \cdot \vdash \mathsf{proc}(a, \mathsf{close}\ a) :: a : \mathbf{1}}\ (\textsc{Conf}\text{-P})$$

The right-hand side is:

$$\frac{\dfrac{}{\cdot\,;\, \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}}\ (\mathbf{1}\text{R})}{\cdot \mid \cdot \vdash \mathsf{msg}(a, \mathsf{close}\ a) :: a : \mathbf{1}}\ (\textsc{Conf}\text{-M})$$

Both sides share the same interface, so this completes the case.

Case (80): The rule is:

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{shift};\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a)$$

The typing judgment for the left-hand side is, by inversion:

$$\frac{\dfrac{\cdot\,;\, \Delta \vdash P :: a : A}{\cdot\,;\, \Delta \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: a : {\downarrow}A}\ ({\downarrow}\text{R})}{\Delta \mid \cdot \vdash \mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{shift};\ P) :: a : {\downarrow}A}\ (\textsc{Conf}\text{-P})$$

The right-hand side is:

$$\dfrac{\dfrac{\cdot\,;\,\Delta \vdash [d/a]P :: d : A}{\Delta \mid \cdot \vdash \mathsf{proc}(d, [d/A]P) :: d : A}\ (\textsc{Conf-P}) \quad \dfrac{\dfrac{\dfrac{}{\cdot\,;\,d : A \vdash d \leftarrow a :: a : A}\ (\textsc{Fwd}^-)}{\cdot\,;\,d : A \vdash \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a :: a : \downarrow A}\ (\downarrow\mathrm{R})}{d : A \mid \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a) :: a : \downarrow A}\ (\textsc{Conf-M})}{\Delta \mid d : A \vdash \mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a) :: a : \downarrow A}\ (\textsc{Conf-C})$$

Both sides share the same interface, so this completes the case.

Case (81): The rule is:

$$\forall \Delta, a, d, c.\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a), \mathsf{proc}(c, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P) \rightarrow \mathsf{proc}(c, [d/a]P)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot\,;\,d : A \vdash d \leftarrow a :: a : A}\ (\textsc{Fwd}^-)}{\cdot\,;\,d : A \vdash \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a :: a : \downarrow A}\ (\downarrow\mathrm{R})}{d : A \mid \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a) :: a : \downarrow A}\ (\textsc{Conf-M}) \quad \dfrac{\dfrac{\cdot\,;\,\Delta, a : A \vdash P :: c : C}{\cdot\,;\,\Delta, a : \downarrow A \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\downarrow\mathrm{L})}{\Delta, a : \downarrow A \mid \cdot \vdash \mathsf{proc}(c, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P) :: c : C}\ (\textsc{Conf-P})}{\Delta, d : A \mid a : \downarrow A \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a), \mathsf{proc}(c, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P) :: a : \downarrow A}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\dfrac{\cdot\,;\,\Delta, d : A \vdash [d/a]P :: c : C}{\Delta, d : A \mid \cdot \vdash \mathsf{proc}(c, [d/a]P) :: c : C}\ (\textsc{Conf-P})$$

Both sides share the same interface, so this completes the case.

Case (5.B): The rule is:

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P), \mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d) \rightarrow \mathsf{proc}(d, [d/a]P)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot\,;\,\Delta \vdash P :: a : A}{\cdot\,;\,\Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A}\ (\uparrow\mathrm{R})}{\Delta \mid \cdot \vdash \mathsf{proc}(a, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P) :: a : \uparrow A}\ (\textsc{Conf-P}) \quad \dfrac{\dfrac{\dfrac{}{\cdot\,;\,a : A \vdash a \rightarrow d :: d : A}\ (\textsc{Fwd}^+)}{\cdot\,;\,a : \uparrow A \vdash \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d :: d : A}\ (\uparrow\mathrm{L})}{a : \uparrow A \mid \cdot \vdash \mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d) :: d : A}\ (\textsc{Conf-M})}{\Delta \mid a : \uparrow A \vdash \mathsf{proc}(a, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P), \mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d) :: d : A}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\dfrac{\cdot\,;\,\Delta \vdash [d/a]P :: d : A}{\Delta \mid \cdot \vdash \mathsf{proc}(d, [d/a]P) :: d : A}\ (\textsc{Conf-P})$$

Both sides share the same interface, so this completes the case.

Case (5.B): The rule is:

$$\forall \Delta, a, c.\mathsf{proc}(c, \mathsf{send}\ a\ \mathsf{shift};\ P) \rightarrow \exists d.\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d), \mathsf{proc}(c, [d/a]P)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot\,;\,\Delta, a : A \vdash P :: c : C}{\cdot\,;\,\Delta, a : \uparrow A \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: c : C}\ (\uparrow\mathrm{L})}{\Delta, a : \uparrow A \mid \cdot \vdash \mathsf{proc}(c, \mathsf{send}\ a\ \mathsf{shift};\ P) :: c : C}\ (\textsc{Conf-P})$$

The right-hand side is:

$$\dfrac{\dfrac{\dfrac{}{\cdot\,;\,a : A \vdash a \rightarrow d :: d : A}\ (\textsc{Fwd}^+)}{\cdot\,;\,a : \uparrow A \vdash \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d :: d : A}\ (\uparrow\mathrm{L})}{a : \uparrow A \mid \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d) :: d : A}\ (\textsc{Conf-M}) \quad \dfrac{\cdot\,;\,\Delta, d : A \vdash [d/a]P :: c : C}{\Delta, d : A \mid \cdot \vdash \mathsf{proc}(d, [d/A]P) :: c : C}\ (\textsc{Conf-P})}{\Delta, a : \uparrow A \mid d : A \vdash \mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \rightarrow d), \mathsf{proc}(c, [d/a]P) :: c : C}\ (\textsc{Conf-C})$$

Both sides share the same interface, so this completes the case.

Case (78): The rule is:

$$\forall \Delta, a.\mathsf{proc}(a, a.k;\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, a.k;\ d \to a)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot\ ;\ \Delta \vdash P :: a : A_k}{\cdot\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l \in L}}\ (\oplus\mathrm{R})}{\Delta\ \vdash\ \cdot \vdash \mathsf{proc}(a, a.k;\ P) :: a : \oplus\{l : A_l\}_{l \in L}}\ (\textsc{Conf-P})$$

The right-hand side is:

$$\dfrac{\dfrac{\cdot\ ;\ \Delta \vdash [d/a]P :: d : A_k}{\Delta\ \vdash\ \cdot \vdash \mathsf{proc}(d, [d/a]P) :: d : A_k}\ (\textsc{Conf-P}) \quad \dfrac{\dfrac{\dfrac{}{\cdot\ ;\ d : A_k \vdash d \to a :: a : A_k}\ (\textsc{Fwd}^+)}{\cdot\ ;\ d : A_k \vdash a.k;\ d \to a :: a : \oplus\{l : A_l\}_{l \in L}}\ (\oplus\mathrm{R})}{d : A_k\ \vdash\ \cdot \vdash \mathsf{msg}(a, a.k;\ d \to a) :: a : \oplus\{l : A_l\}_{l \in L}}\ (\textsc{Conf-M})}{\Delta\ \vdash\ d : A_k \vdash \mathsf{proc}(d, [d/a]P), \mathsf{msg}(d, a.k;\ d \to a) :: a : \oplus\{l : A_l\}_{l \in L}}\ (\textsc{Conf-C})$$

Both sides share the same interface, so this completes the case.

Case (79): The rule is:

$$\forall a, d, \Delta, c.\mathsf{msg}(a, a.k;\ d \to a), \mathsf{proc}(c, \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L}) \to \mathsf{proc}(c, [d/a]P_k)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot\ ;\ d : A_k \vdash d \to a :: a : A_k}\ (\textsc{Fwd}^+)}{\cdot\ ;\ d : A_k \vdash a.k;\ d \to a :: a : \oplus\{l : A_l\}_{l \in L}}\ (\oplus\mathrm{R})}{d : A_k\ \vdash\ \cdot \vdash \mathsf{msg}(a, a.k;\ d \to a) :: a : \oplus\{l : A_l\}_{l \in L}}\ (\textsc{Conf-M}) \quad \dfrac{\dfrac{\cdot\ ;\ \Delta, a : A_l \vdash P_l :: c : C \quad (\forall l \in L)}{\cdot\ ;\ \Delta, a : \oplus\{l : A_l\}_{l \in L} \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L} :: c : C}\ (\oplus\mathrm{L})}{\Delta, a : \oplus\{l : A_l\}_{l \in L}\ \vdash\ \cdot \vdash \mathsf{proc}(c, \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L}) :: c : C}\ (\textsc{Conf-P})}{\Delta, d : A_k\ \vdash\ a : \oplus\{l : A_l\}_{l \in L} \vdash \mathsf{msg}(a, a.k;\ d \to a), \mathsf{proc}(c, \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L}) :: c : C}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\dfrac{\cdot\ ;\ \Delta, d : A_k \vdash [d/a]P_k :: c : C}{\Delta, d : A_k\ \vdash\ \cdot \vdash \mathsf{proc}(c, [d/a]P_k) :: c : C}\ (\textsc{Conf-P})$$

Both sides share the same interface, so this completes the case.

Case (69): The rule is:

$$\forall \Delta, b, a.\mathsf{proc}(a, \mathsf{send}\ a\ b;\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot\ ;\ \Delta \vdash P :: a : A}{\cdot\ ;\ \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A}\ (\otimes\mathrm{R})}{\Delta, b : B\ \vdash\ \cdot \vdash \mathsf{proc}(a, \mathsf{send}\ a\ b;\ P) :: a : B \otimes A}\ (\textsc{Conf-P})$$

The right-hand side is:

$$\dfrac{\dfrac{\cdot\ ;\ \Delta \vdash [d/a]P :: d : A}{\Delta\ \vdash\ \cdot \vdash \mathsf{proc}(d, [d/a]P) :: d : A}\ (\textsc{Conf-P}) \quad \dfrac{\dfrac{\dfrac{}{\cdot\ ;\ d : A \vdash d \to a :: a : A}\ (\textsc{Fwd}^+)}{\cdot\ ;\ b : B, d : A \vdash \mathsf{send}\ a\ b;\ d \to a :: a : B \otimes A}\ (\otimes\mathrm{R})}{b : B, d : A\ \vdash\ \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a) :: a : B \otimes A}\ (\textsc{Conf-M})}{\Delta, b : B\ \vdash\ d : A \vdash \mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a) :: a : B \otimes A}\ (\textsc{Conf-C})$$

Both sides share the same interface, so this completes the case.

Case (70): The rule is:

$$\forall a, e, d, \Delta, c.\mathsf{msg}(a, \mathsf{send}\ a\ e;\ d \to a), \mathsf{proc}(c, b \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [e, d/b, a]P)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot\ ;\ d : A \vdash d \to a :: a : A}\ (\textsc{Fwd}^+)}{\cdot\ ;\ e : B, d : A \vdash \mathsf{send}\ a\ e;\ d \to a :: a : B \otimes A}\ (\otimes\mathrm{R})}{e : B, d : A\ \vdash\ \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ e;\ d \to a) :: a : B \otimes A}\ (\textsc{Conf-M}) \quad \dfrac{\dfrac{\cdot\ ;\ \Delta, a : A, b : B \vdash b \leftarrow \mathsf{recv}\ a;\ P :: c : C}{\cdot\ ;\ \Delta, a : B \otimes A \vdash b \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\otimes\mathrm{L})}{\Delta, a : B \otimes A\ \vdash\ \cdot \vdash \mathsf{proc}(c, b \leftarrow \mathsf{recv}\ a;\ P) :: c : C}\ (\textsc{Conf-P})}{\Delta, d : A, e : B\ \vdash\ a : B \otimes A \vdash \mathsf{msg}(a, \mathsf{send}\ a\ e;\ d \to a), \mathsf{proc}(c, b \leftarrow \mathsf{recv}\ a;\ P) :: c : C}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\frac{\cdot\,;\,\Delta, d : A, e : B \vdash [e, d/b, a]P :: c : C}{\Delta, d : A, e : B \;\vdash\; \cdot \vdash \mathsf{proc}(c, b \leftarrow \mathsf{recv}\ a;\ P) :: c : C}\ (\textsc{Conf-P})$$

Both sides share the same interface, so this completes the case.

Case (74): The rule is:

$$\forall a, \Delta.\mathbf{eval}(M, v), \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) \rightarrow \\ \rightarrow \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot \Vdash M : \tau \quad \cdot\,;\,\Delta \vdash P :: a : A}{\cdot\,;\,\Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A}\ (\wedge\mathrm{R})}{\Delta \;\vdash\; \cdot \vdash \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) :: a : \tau \wedge A}\ (\textsc{Conf-P})$$

By proposition 5.8.1, $\cdot \Vdash v : \tau$. The right-hand side is:

$$\dfrac{\dfrac{\cdot\,;\,\Delta \vdash [d/a]P :: d : A}{\Delta \;\vdash\; \cdot \vdash \mathsf{proc}(d, [d/a]P) :: d : A}\ (\textsc{Conf-P}) \quad \dfrac{\dfrac{\cdot \Vdash v : \tau \quad \dfrac{}{\cdot\,;\,d : A \vdash d \rightarrow a :: a : A}\ (\textsc{Fwd}^+)}{\cdot\,;\,d : A \vdash \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a :: a : \tau \wedge A}\ (\wedge\mathrm{R})}{d : A \;\vdash\; \cdot \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a) :: a : \tau \wedge A}\ (\textsc{Conf-M})}{\Delta \;\vdash\; d : A \vdash \mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a) :: a : \tau \wedge A}\ (\textsc{Conf-C})$$

Both sides share the same interface, so this completes the case.

Case (75): The rule is:

$$\forall \Delta, a, d, c.\mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a), \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) \rightarrow \\ \rightarrow \mathsf{proc}(c, [d, v/a, x]P)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\dfrac{\cdot \Vdash v : \tau \quad \dfrac{}{\cdot\,;\,d : A \vdash d \rightarrow a :: a : A}\ (\textsc{Fwd}^+)}{\cdot\,;\,d : A \vdash \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a :: a : \tau \wedge A}\ (\wedge\mathrm{R})}{d : A \;\vdash\; \cdot \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a) :: a : \tau \wedge A}\ (\textsc{Conf-M}) \quad \dfrac{\dfrac{x : \tau\,;\,\Delta, a : A \vdash P :: c : C}{\cdot\,;\,\Delta, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ P :: c : C}\ (\wedge\mathrm{L})}{\Delta, a : \tau \wedge A \;\vdash\; \cdot \vdash \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) :: c : C}\ (\textsc{Conf-P})}{\Delta, d : A \;\vdash\; a : \tau \wedge A \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \rightarrow a), \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) :: c : C}\ (\textsc{Conf-C})$$

The right-hand side is:

$$\frac{\cdot\,;\,\Delta, d : A \vdash [d, v/a, x]P :: c : C}{\Delta, d : A \;\vdash\; \cdot \vdash \mathsf{proc}(c, [d, v/a, x]P) :: c : C}\ (\textsc{Conf-P})$$

Both sides share the same interface, so this completes the case.

Case (82): The rule is:

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{unfold};\ P) \rightarrow \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \rightarrow a)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\cdot\,;\,\Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_s^+}{\cdot\,;\,\Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A}\ (\rho^+\mathrm{R})}{\Delta \;\vdash\; \cdot \vdash \mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{unfold};\ P) :: a : \rho\alpha.A}\ (\textsc{Conf-P})$$

The right-hand side is:

$$\dfrac{\dfrac{\cdot\,;\,\Delta \vdash [d/a]P :: d : [\rho\alpha.A/\alpha]A}{\Delta \;\vdash\; \cdot \vdash \mathsf{proc}(d, [d/a]P) :: d : [\rho\alpha.A/\alpha]A}\ (\textsc{Conf-P}) \quad \dfrac{\dfrac{\dfrac{}{\cdot\,;\,d : [\rho\alpha.A/\alpha]A \vdash d \rightarrow a :: a : [\rho\alpha.A/\alpha]A}\ (\textsc{Fwd}^+) \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_s^+}{\cdot\,;\,d : [\rho\alpha.A/\alpha]A \vdash \mathsf{send}\ a\ \mathsf{unfold};\ d \rightarrow a :: a : \rho\alpha.A}\ (\rho^+\mathrm{R})}{d : [\rho\alpha.A/\alpha]A \;\vdash\; \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \rightarrow a) :: a : \rho\alpha.A}\ (\textsc{Conf-M})}{\Delta \;\vdash\; d : [\rho\alpha.A/\alpha]A \vdash \mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \rightarrow a) :: a : \rho\alpha.A}\ (\textsc{Conf-C})$$

Both sides share the same interface, so this completes the case.

Case (83): The rule is:

$$\forall \Delta, a, d.\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a), \mathsf{proc}(c, \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [d/a]P)$$

The typing judgment for the left-hand side is, by inversion:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot\,;\, d : [\rho\alpha.A/\alpha]A \vdash d \to a :: a : [\rho\alpha.A/\alpha]A}\,(\textsc{Fwd}^+) \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}^+_s}{\cdot\,;\, d : [\rho\alpha.A/\alpha]A \vdash \mathsf{send}\ a\ \mathsf{unfold};\ d \to a :: a : \rho\alpha.A}\,(\rho^+\mathrm{R})}{d : [\rho\alpha.A/\alpha]A \mid \cdot \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a) :: a : \rho\alpha.A}\,(\textsc{Conf-M}) \quad \dfrac{\dfrac{\cdot\,;\, \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}^+_s}{\cdot\,;\, \Delta, a : \rho\alpha.A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: c : C}\,(\rho^+\mathrm{L})}{\Delta, a : \rho\alpha.A \mid \cdot \vdash \mathsf{proc}(c, \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) :: c : C}\,(\textsc{Conf-P})}{\Delta, d : [\rho\alpha.A/\alpha]A \mid a : \rho\alpha.A \vdash \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \to a), \mathsf{proc}(c, \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) :: c : C}\,(\textsc{Conf-C})$$

The right-hand side is:

$$\dfrac{\cdot\,;\, \Delta, d : [\rho\alpha.A/\alpha]A \vdash [d/a]P :: c : C}{\Delta, d : [\rho\alpha.A/\alpha]A \mid \cdot \vdash \mathsf{proc}(c, [d/a]P) :: c : C}\,(\textsc{Conf-P})$$

Both sides share the same interface, so this completes the case. □

As a corollary of proposition 5.9.1, we know that in a trace $T = (M_0, (r_i; \delta_i)_i)$ from a well-typed $\Sigma \parallel \Gamma \mid \mathrm{I}_0 \vdash M_0 :: \Delta$, we have for all i some Σ_i and I_i such that $\Sigma_i \parallel \Gamma \mid \mathrm{I}_i \vdash M_i :: \Delta$. Indeed, Σ_i and I_i are given by induction on n, where each step is given by an application of proposition 5.9.1. We also know that every channel appearing in a process trace has an associated session type, a fact that we will use repeatedly when reasoning about process traces. Definition 5.9.2 captures this relationship between traces, channels, and types:

Definition 5.9.2. Consider a trace $T = (M_0, (r_i; \delta_i)_i)$ from $\Sigma \parallel \Gamma \mid \mathrm{I}_0 \vdash M_0 :: \Delta$. We write $T \vdash c : A$ to mean that $c : A$ appears in Γ, Δ, or I_i for some i. ◂

By proposition 5.6.10, $T \vdash c : A$ is an entire relation from free channel names appearing in $\bigcup_i \mathsf{fc}(M_i)$ to session types. In fact, it is a total function:

Corollary 5.9.3. *Let $T = (M_0, (r_i; \delta_i)_i)$ be a trace from $\Sigma \parallel \Gamma \mid \mathrm{I}_0 \vdash M_0 :: \Delta$. For all c, if $T \vdash c : A$ and $T \vdash c : A'$, then $A = A'$.*

Proof. It is sufficient to show that if $c : A$ is in $\Gamma, \mathrm{I}_n, \Delta$ and $c : A'$ is in $\Gamma, \mathrm{I}_k, \Delta$ for some $k \le n$, then $A = A'$. We do so by induction on n.

Case $n = 0$: Then $0 \le k \le n$ implies that $k = 0$. The result is immediate by well-formedness of $\Sigma \parallel \Gamma \mid \mathrm{I} \vdash M_0 :: \Delta$.

Case $n = n' + 1$: Assume the result for n'. If $k = n$, then $A = A'$ by well-formedness of $\Sigma \parallel \Gamma \mid \mathrm{I}_n \vdash M_n :: \Delta$. Otherwise $k < n$, so in particular, $k \le n'$. By proposition 5.9.1, $c : A$ is in $\Gamma, \mathrm{I}_{n'}, \Delta$ (preservation implies that a channel cannot reappear in a trace after having disappeared). By the induction hypothesis, $A = A'$ as desired. □

The following proposition further confirms that the types assigned by $T \vdash c : A$ are consistent with those for message and process facts appearing in the trace T:

Proposition 5.9.4. *Let $T = (M_0, (r_i; \delta_i)_i)$ be a trace from $\Sigma_0 \parallel \Gamma \mid \mathrm{I}_0 \vdash M_0 :: \Delta_0$, and let Σ_i and I_i be given by recursion on n and proposition 5.9.1. For all n, if $\mathsf{proc}(c_0, P) \in M_n$, then*

1. *$c_0 \in \mathsf{fc}(P)$;*
2. *for all $c_i \in \mathsf{fc}(P)$, then $T \vdash c_i : A_i$ for some A_i; and*
3. *where $\mathsf{fc}(P) = \{c_0, \ldots, c_m\}$, we have $\cdot\,;\, c_1 : A_1, \ldots, c_m : A_m \vdash P :: c_0 : A_0$.*

If $\mathsf{msg}(c_0, m) \in M_n$, then

- *if $m = \mathsf{close}\ c$, then $T \vdash c : \mathbf{1}$;*
- *if $m = c.l_j;\ d \to c$, then $T \vdash c : \oplus\{l_i : A_i\}_{i \in I}$ for some A_i ($i \in I$), and $T \vdash d : A_j$ for some $j \in I$;*
- *if $m = c.l_j;\ c \leftarrow d$, then $T \vdash c : \&\{l_i : A_i\}_{i \in I}$ for some A_i ($i \in I$), and $T \vdash d : A_j$ for some $j \in I$;*
- *if $m = \mathsf{send}\ c\ a;\ b \to c$, then $T \vdash c : A \otimes B$, $T \vdash a : A$, and $T \vdash b : B$ for some A and B;*
- *if $m = \mathsf{send}\ c\ a;\ c \leftarrow b$, then $T \vdash c : A \multimap B$, $T \vdash a : A$, and $T \vdash b : B$ for some A and B;*

- *if* $m = \_ \leftarrow \mathsf{output}\ c\ v;\ d \to c$, *then* $T \vdash c : \tau \wedge A$ *and* $T \vdash d : A$ *for some* A *and* τ *such that* $\cdot \Vdash v : \tau$;
- *if* $m = \_ \leftarrow \mathsf{output}\ c\ v;\ c \leftarrow d$, *then* $T \vdash c : \tau \supset A$ *and* $T \vdash d : A$ *for some* A *and* τ *such that* $\cdot \Vdash v : \tau$;
- *if* $m = \mathsf{send}\ c\ \mathsf{unfold};\ d \to c$, *then* $T \vdash c : \rho\alpha.A$ *and* $T \vdash d : [\rho\alpha.A/\alpha]A$ *for some* $\alpha \vdash A\ \mathsf{type}_s^+$; *and*
- *if* $m = \mathsf{send}\ c\ \mathsf{unfold};\ c \leftarrow d$, *then* $T \vdash c : \rho\alpha.A$ *and* $T \vdash d : [\rho\alpha.A/\alpha]A$ *for some* $\alpha \vdash A\ \mathsf{type}_s^-$.

Proof. By assumption, we have $\Sigma_i \parallel \Delta \mid \mathrm{I}_i \vdash M_i :: (c : A)$ for all i. Assume $\mathsf{proc}(c_0, P) \in M_n$ or $\mathsf{msg}(c, m) \in M_n$. Then the result is immediate by proposition 5.6.9 and inversion on the typing judgment for processes. □

5.9.2. Fairness. We show that every well-typed configuration has a fair execution, and that its fair executions are all union-equivalent (recall definition 3.3.22). These two facts will follow easily from the fact that the MRS $\mathcal{P}$ is non-overlapping on well-typed configurations. The proof of non-overlapping property depends on the following sequence of technical results.

The first technical lemma characterizes the input and output channel names for facts appearing in rules.

LEMMA 5.9.5. *If* $F(\vec{k}) \xrightarrow{(r;(\vec{k},\vec{a}))} G(\vec{k},\vec{a})$ *by a rule* $r \in \mathcal{P}$, *then*

(1) if $\mathsf{msg}(c, m) \in F(\vec{k})$, *then*
  (a) $F(\vec{k}) = \mathsf{msg}(c, m), \mathsf{proc}(d, P)$ *for some* d *and* P,
  (b) $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathsf{ic}_s(\mathsf{proc}(d, P))$, *and*
  (c) $\mathsf{cc}(\mathsf{msg}(c, m)) \notin \mathsf{fc}(G(\vec{k},\vec{a}))$;
(2) if $F(\vec{k}) = \mathsf{proc}(c, P)$, *then*
  (a) $G(\vec{k},\vec{a}) = \mathsf{msg}(d, m), \mathsf{proc}(e, Q)$ *for some* d, m, e, *and* Q, *and*
  (b) $\mathsf{cc}(\mathsf{msg}(d, m)) \in \mathsf{oc}_s(\mathsf{proc}(c, P))$;
(3) if $G(\vec{k},\vec{a}) = \mathsf{msg}(c, m)$, *then*
  (a) $\mathsf{proc}(d, P) \in F(\vec{k})$ *for some* d *and* P, *and*
  (b) $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathsf{oc}_s(\mathsf{proc}(d, P))$;
(4) if $G(\vec{k},\vec{a}) = \mathsf{msg}(c, m), \mathsf{proc}(d, P)$, *then*
  (a) $\mathsf{proc}(e, Q) \in F(\vec{k})$ *for some* e *and* Q,
  (b) $\mathsf{oc}_s(\mathsf{proc}(d, P)) \subseteq \mathsf{oc}_s(\mathsf{proc}(e, Q)) \cup \vec{a}$,
  (c) $\mathsf{kc}(\mathsf{msg}(c, m)) \in \vec{a}$,
  (d) $\mathsf{kc}(\mathsf{msg}(c, m)) \in \mathsf{fc}(\mathsf{proc}(d, P))$,
  (e) $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathsf{oc}_s(\mathsf{proc}(d, P))$, *and*
  (f) $\mathsf{cc}(\mathsf{msg}(c, m)) \notin \mathsf{fc}(\mathsf{proc}(d, P))$;
(5) if $G(\vec{k},\vec{a}) = \mathsf{proc}(c, P), \mathsf{proc}(d, Q)$, *then*
  (a) r *is* (66),
  (b) $F(\vec{k}) = \mathsf{proc}(d, c \leftarrow P;\ Q)$, *where without loss of generality,* $\vec{a} = c$, *and*
  (c) $\mathsf{oc}_s(G(\vec{k},\vec{a})) \subseteq \mathsf{oc}_s(F(\vec{k})) \cup \{c\}$.

The above enumeration of cases for $G(\vec{k},\vec{a})$ *is exhaustive.*

Proof. Immediate by a case analysis on the rules, using proposition 5.5.5 to simplify reasoning. □

Remark 5.9.6. We avoided the question of configurations being well-typed in lemma 5.9.5 by using static input and output channel names, which lift to facts in the obvious way.

Corollary 5.9.7 shows that a configuration never consumes a message that it sends on its interface:

COROLLARY 5.9.7. *If* $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ *and* $\mathcal{C} \to \mathcal{C}'$, *then for all* $\mathsf{msg}(c, m) \in \mathcal{C}$, *if* $\mathsf{cc}(\mathsf{msg}(c, m)) \in \check{\Gamma}, \check{\Delta}$, *then* $\mathsf{msg}(c, m) \in \mathcal{C}'$.

Proof. Let $\mathsf{msg}(c, m) \in \mathcal{C}$ with $\mathsf{cc}(\mathsf{msg}(c, m)) \in \check{\Gamma}, \check{\Delta}$ be arbitrary. The only way for $\mathsf{msg}(c, m) \in \mathcal{C}$ but $\mathsf{msg}(c, m) \notin \mathcal{C}'$ is for the step to be by a rule with $\mathsf{msg}(c, m)$ in its active multiset $F(\vec{k})$. By lemma 5.9.5, this implies that there exists some $\mathsf{proc}(d, P) \in F(\vec{k})$ with $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathsf{ic_s}(\mathsf{proc}(d, P))$. By proposition 5.5.5, this implies that $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathsf{ic}(\mathsf{proc}(d, P))$. But then by lemma 5.6.11, $\mathsf{cc}(\mathsf{msg}(c, m)) \in \mathrm{I}$. This implies that $\mathsf{cc}(\mathsf{msg}(c, m)) \notin \check{\Gamma}, \check{\Delta}$, a contradiction. □

As a second corollary, each channel in a trace appears as the carrier channel of at most one message judgment:

Corollary 5.9.8. *Let $T = (M_0, (r_i; \delta_i)_i)$ be a trace from $\Gamma \mid \mathrm{I}_0 \vdash M_0 :: \Delta_0$. For all $j \leq k$, if $\mathsf{msg}(c_j, m_j) \in M_j$ and $K \in M_k$, then $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc}(K)$ or $K = \mathsf{msg}(c_j, m_j)$.*

Proof. We proceed by induction on k to show that if $K \in M_k$, then for all $0 \leq j \leq k$, if $\mathsf{msg}(c_j, m_j) \in M_j$, then $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc}(K)$ or $K = \mathsf{msg}(c_j, m_j)$.

Case $k = 0$**:** The base case is given by lemma 5.6.13.

Case $k = k' + 1$**:** Assume the result for some k'. Assume that $M_{k'} \to M_k$ by some rule instantiation $F(\vec{h}) \xrightarrow{(r;(\vec{h},\vec{a}))} G(\vec{h}, \vec{a})$. Let $\mathsf{msg}(c_j, m_j) \in M_j$ and $K \in M_k$ be arbitrary, where $0 \leq j \leq k$. If $K = \mathsf{msg}(c_j, m_j)$, then we are done. Now assume that $K \neq \mathsf{msg}(c_j, m_j)$. We proceed by case analysis $j \leq k$:

Subcase $j = k$**:** Then $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc}(K)$ by lemma 5.6.12 and proposition 5.9.1.

Subcase $j < k$**:** If K is in the stationary subset, then $K \in M_{k'}$ and we are done by the induction hypothesis. Otherwise, $K \in G(\vec{h}, \vec{a})$. We proceed by case analysis on the fact K:

Subsubcase $K = \mathsf{proc}(c_k, P_k)$**:** By lemma 5.9.5, $\mathsf{oc}(\mathsf{proc}(c_k, P_k)) \subseteq \mathsf{oc_s}(\mathsf{proc}(d, P)) \cup \vec{a}$ for some $\mathsf{proc}(d, P) \in M_{k'}$. By the induction hypothesis, $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc}(\mathsf{proc}(d, P))$. By proposition 5.5.5, $\mathsf{oc_s}(\mathsf{proc}(d, P)) \subseteq \mathsf{oc}(\mathsf{proc}(d, P))$. So $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc_s}(\mathsf{proc}(d, P))$. By freshness, $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \vec{a}$. So $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc_s}(\mathsf{proc}(d, P)) \cup \vec{a}$. It follows that $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc}(\mathsf{proc}(c_k, P_k))$ as desired.

Subsubcase $K = \mathsf{msg}(c_k, m_k)$**:** By lemma 5.9.5, $\mathsf{cc}(\mathsf{msg}(c_k, m_k)) \in \mathsf{oc_s}(\mathsf{proc}(d, P))$ for some $\mathsf{proc}(d, P) \in M_{k'}$. By the induction hypothesis, $\mathsf{cc}(\mathsf{msg}(c_j, m_j)) \notin \mathsf{oc}(\mathsf{proc}(d, P))$. By proposition 5.5.5, $\mathsf{oc_s}(\mathsf{proc}(d, P)) \subseteq \mathsf{oc}(\mathsf{proc}(d, P))$. It follows that $\mathsf{cc}(\mathsf{msg}(c_k, m_k)) \notin \mathsf{oc}(\mathsf{msg}(c_j, m_j))$ as desired. □

Next, we use our technical results to show the that the multiset rewriting system for Polarized SILL is non-overlapping on well-typed configurations:

Proposition 5.9.9. *If $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$, then the MRS $\mathcal{P}$ is non-overlapping on $\mathcal{C}$.*

Proof. It is sufficient to show that if $s_1(\phi_1)$ and $s_2(\phi_2)$ are distinct instantiations applicable to $\mathcal{C}$, then $F_1(\phi_1)$ and $F_2(\phi_2)$ are disjoint multisets: $F_1(\phi_1) \cap F_2(\phi_2) = \varnothing$. Indeed, if this is the case and $s_1(\phi_1), \ldots, s_k(\phi_k)$ are the distinct rule instantiations applications to M_n, then $F_1(\phi_1), \ldots, F_k(\phi_k)$ are all pairwise-disjoint multisets. It follows that $F_1(\phi_1), \ldots, F_k(\phi_k) \subseteq M_n$, so the overlap in $\mathcal{C}$ is empty: $\Omega_{\mathcal{C}}(F_1(\phi_1), \ldots, F_k(\phi_k)) = \varnothing$.

We proceed by case analysis on the possible judgments in $F_1(\phi_1) \cap F_2(\phi_2)$.

Case $\mathsf{msg}(d, m)$**:** By lemma 5.9.5, there exist $\mathsf{proc}(d_i, P_i) \in F_i(\phi_i)$ with $\mathsf{cc}(\mathsf{msg}(d, m)) \in \mathsf{ic}(P_i)$ for $i = 1, 2$. If $P_1 = P_2$, then an inspection of the rules reveals that $s_1 = s_2$ and $\phi_1 = \phi_2$, so we are done. Suppose to the contrary that $P_1 \neq P_2$. So $\mathsf{cc}(\mathsf{msg}(d, m)) \in \mathsf{ic}(P_1) \cap \mathsf{ic}(P_2)$. This is a contradiction by lemma 5.6.12.

Case $\mathsf{proc}(e, P)$**:** Then $s_1 = s_2$ by a case analysis on the rules. We show that $\phi_1 = \phi_2$. If s_1 is one of (5.B), (66) to (69), (72), (73), (77), (78), (80), (82), (87) and (89), then $\phi_1 = \phi_2$ because all constants matched by ϕ_1 and ϕ_2 appear in $\mathsf{proc}(e, P)$. If s_1 is one of (5.B), (64), (65), (70), (71), (76), (79), (81), (83), (86) and (88), then for $i = 1, 2$, the multiset $F_i(\phi_i)$ contain a fact $\mathsf{msg}(d_i, m_i)$ where there is a channel name $e_i \in m_i$ that appears in ϕ_i, but not in $\mathsf{proc}(c, P)$ (explicitly, e_i is the name of the continuation channel). If $m_1 = m_2$, then we are done, for an inspection of the rules reveals that $\phi_1 = \phi_2$. Suppose to the contrary that $m_1 \neq m_2$. By lemma 5.6.14, $\cdot ; \Delta_i' \vdash m_i :: d_i : D_i$ for some

Δ_i' and $d_i : D_i$. Inspection of the rules reveals that $\mathsf{cc}(\mathsf{msg}(d_1, m_1)) = \mathsf{cc}(\mathsf{msg}(d_2, m_2)) \in \mathsf{ic}(P)$. This is a contradiction by lemma 5.6.12. □

Corollary 5.9.10. *Every configuration* $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ *has a fair execution. Its fair executions are all permutations of each other and they are all union-equivalent.*

Proof. By propositions 5.9.1 and 5.9.9, $\mathcal{P}$ is non-overlapping from $\mathcal{C}$. By proposition 3.3.9, this implies that it commutes from $\mathcal{C}$, so a fair execution exists by proposition 3.3.7. All of its fair executions are permutations of each other by proposition 3.3.21. They are union-equivalent by corollary 3.3.24. □

Corollary 5.9.11. *Every process* $\cdot\,;\Delta \vdash P :: c : A$ *has a fair execution. Its fair executions are all permutations of each other and they are all union-equivalent.*

Proof. Immediate by corollary 5.9.10 with the initial configuration $\Delta \mid \cdot \vdash \mathsf{proc}(c, P) :: c : A$. □

5.10. Related Work

Honda [Hon93] and Takeuchi, Honda, and Kubo [THK94] introduced session types to describe sessions of interaction. Caires and Pfenning [CP10] observed a proofs-as-programs correspondence between the session-typed π-calculus and intuitionistic linear logic, where the (Cut) rule captures process communication. Toninho, Caires, and Pfenning [TCP13] built on this correspondence and introduced SILL's monadic integration between functional and synchronous message-passing programming. They specified SILL's operational behaviour using a substructural operational semantics (SSOS). Gay and Vasconcelos [GV10] introduced asynchronous communication for session-typed languages. They used an operational semantics and buffers to model asynchronicity. Pfenning and Griffith [PG15] observed that the polarity of a type determines the direction of communication along a channel. They observed that synchronous communication can be encoded in an asynchronous setting using explicit shift operators. They gave a computational interpretation to polarized adjoint logic. In this interpretation, linear propositions, affine propositions, and unrestricted propositions correspond to different modes in which resources can be used.

There are several process calculi and session-typed programming languages that are closely related to Polarized SILL, and to which we conjecture our techniques could be extended. Wadler [Wad14] introduced "Classical Processes" (CP), a proofs-as-programs interpretation of classical linear logic that builds on the ideas of Caires and Pfenning [CP10]. CP supports replication but not recursion. Though CP does not natively support functional programming, Wadler gives a translation for GV, a linear functional language with pairs but no recursion, into CP. In contrast, Polarized SILL uniformly integrates functional programming and message-passing concurrency. CP has a synchronous communication semantics and does not have an explicit treatment of polarities. Polarized SILL has an asynchronous communication semantics, and synchronous communication is encoded using polarity shifts, even though we do not detail this construction here.

Kokke, Montesi, and Peressotti [KMP19] introduced "hypersequent classical processes" (HCP). HCP is a revised proofs-as-processes interpretation between classical linear logic and the π-calculus. Building on Atkey's [Atk17] semantics for CP, they gave HCP a denotational semantics using Brzozowski derivatives [Brz64]. HCP does not include recursion, shifts, or functional value transmission.

Gommerstadt, Jia, and Pfenning [GJP18] introduced run-time monitors for a dependent version of Polarized SILL. Our type system for configurations is inspired by theirs [GJP18, p. 786].

Pruiksma and Pfenning [PP21] gave a message passing interpretation to adjoint logic. It supports richer communication topologies than Polarized SILL. For example, it supports *multicast*, *replicable services*, and *cancellation*. Its operational semantics is specified by a multiset rewriting system. It enjoys session-fidelity and deadlock-freedom.

5.A. Complete Listing of Typing Rules for Polarized SILL

For ease of reference, we collect all of the rules for Polarized SILL in this appendix.

5.A.1. Rules for Term Formation.

$$\frac{\Psi\ ;\ \overline{a_i : A_i} \vdash P :: a : A}{\Psi \Vdash a \leftarrow \{P\} \leftarrow \overline{a_i} : \{a : A \leftarrow \overline{a_i : A_i}\}}\ (\text{I-}\{\})$$

$$\frac{}{\Psi, x : \tau \Vdash x : \tau}\ (\text{F-Var}) \quad \frac{\Psi, x : \tau \Vdash M : \tau}{\Psi \Vdash \mathsf{fix}\ x.M : \tau}\ (\text{F-Fix})$$

$$\frac{\Psi, x : \tau \Vdash M : \sigma}{\Psi \Vdash \lambda x : \tau.M : \tau \to \sigma}\ (\text{F-Fun}) \quad \frac{\Psi \Vdash M : \tau \to \sigma \quad \Psi \Vdash N : \tau}{\Psi \Vdash MN : \sigma}\ (\text{F-App})$$

5.A.2. Rules for Process Formation.

$$\frac{\cdot \vdash A\ \mathsf{type}_s^+}{\Psi\ ;\ a : A \vdash a \to b :: b : A}\ (\text{Fwd}^+) \quad \frac{\cdot \vdash A\ \mathsf{type}_s^-}{\Psi\ ;\ a : A \vdash a \leftarrow b :: b : A}\ (\text{Fwd}^-)$$

$$\frac{\Psi\ ;\ \Delta_1 \vdash P :: a : A \quad \Psi\ ;\ a : A, \Delta_2 \vdash Q :: c : C}{\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C}\ (\text{Cut})$$

$$\frac{\Psi \Vdash M : \{a : A \leftarrow \overline{a_i : A_i}\}}{\Psi\ ;\ \overline{a_i : A_i} \vdash a \leftarrow \{M\} \leftarrow \overline{a}_i :: a : A}\ (\text{E-}\{\})$$

$$\frac{}{\Psi\ ;\ \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}}\ (\mathbf{1}\text{R}) \quad \frac{\Psi\ ;\ \Delta \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C}\ (\mathbf{1}\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: a : \downarrow A}\ (\downarrow\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \downarrow A \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\downarrow\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a :}{\Psi\ ;\ \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A}\ (\uparrow\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \uparrow A \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: c : C}\ (\uparrow\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : A_k \quad (k \in L)}{\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l \in L}}\ (\oplus\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : A_l \vdash P_l :: c : C \quad (\forall l \in L)}{\Psi\ ;\ \Delta, a : \oplus\{l : A_l\}_{l \in L} \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L} :: c : C}\ (\oplus\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P_l :: a : A_l \quad (\forall l \in L)}{\Psi\ ;\ \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l \in L} :: a : \&\{l : A_l\}_{l \in L}}\ (\&\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : A_k \vdash P :: c : C \quad (k \in L)}{\Psi\ ;\ \Delta, a : \&\{l : A_l\}_{l \in L} \vdash a.k;\ P :: c : C}\ (\&\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A}\ (\otimes\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : A, b : B \vdash P :: c : C}{\Psi\ ;\ \Delta, a : B \otimes A \vdash b \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\otimes\text{L})$$

$$\frac{\Psi\ ;\ \Delta, b : B \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A}\ (\multimap\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, b : B, a : B \multimap A \vdash \mathsf{send}\ a\ b;\ P :: c : C}\ (\multimap\text{L})$$

$$\frac{\Psi \Vdash M : \tau \quad \Psi\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A}\ (\wedge\text{R}) \quad \frac{\Psi, x : \tau\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ P :: c : C}\ (\wedge\text{L})$$

$$\frac{\Psi, x : \tau\ ;\ \Delta \vdash P :: a : A}{\Psi\ ;\ \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A}\ (\supset\text{R}) \quad \frac{\Psi \Vdash M : \tau \quad \Psi\ ;\ \Delta, a : A \vdash P :: c : C}{\Psi\ ;\ \Delta, a : \tau \supset A \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: c : C}\ (\supset\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_s^+}{\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A}\ (\rho^+\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_s^+}{\Psi\ ;\ \Delta, a : \rho\alpha.A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: c : C}\ (\rho^+\text{L})$$

$$\frac{\Psi\ ;\ \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_s^-}{\Psi\ ;\ \Delta \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: a : \rho\alpha.A}\ (\rho^-\text{R}) \quad \frac{\Psi\ ;\ \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C \quad \cdot \vdash \rho\alpha.A\ \mathsf{type}_s^-}{\Psi\ ;\ \Delta, a : \rho\alpha.A \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: c : C}\ (\rho^-\text{L})$$

5.A.3. Rules for Type Formation.

$$\frac{}{\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+}\ (\text{C}\mathbf{1}) \quad \frac{}{\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p}\ (\text{CVar})$$

$$\frac{\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+}{\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+}\ (\text{C}\rho^+) \quad \frac{\Xi, \alpha\ \mathsf{type}_s^- \vdash A\ \mathsf{type}_s^-}{\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^-}\ (\text{C}\rho^-)$$

$$\frac{\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}^{-}}{\Xi \vdash {\downarrow}A\ \mathsf{type}_{\mathsf{s}}^{+}}\ (\mathrm{C}{\downarrow}) \qquad \frac{\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}}{\Xi \vdash {\uparrow}A\ \mathsf{type}_{\mathsf{s}}^{-}}\ (\mathrm{C}{\uparrow})$$

$$\frac{\Xi \vdash A_l\ \mathsf{type}_{\mathsf{s}}^{+} \quad (\forall l \in L)}{\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_{\mathsf{s}}^{+}}\ (\mathrm{C}\oplus) \qquad \frac{\Xi \vdash A_l\ \mathsf{type}_{\mathsf{s}}^{-} \quad (\forall l \in L)}{\Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_{\mathsf{s}}^{-}}\ (\mathrm{C}\&)$$

$$\frac{\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}^{+} \quad \Xi \vdash B\ \mathsf{type}_{\mathsf{s}}^{+}}{\Xi \vdash A \otimes B\ \mathsf{type}_{\mathsf{s}}^{+}}\ (\mathrm{C}\otimes) \qquad \frac{\Xi \vdash B\ \mathsf{type}_{\mathsf{s}}^{+} \quad \Xi \vdash A\ \mathsf{type}_{\mathsf{s}}^{-}}{\Xi \vdash B \multimap A\ \mathsf{type}_{\mathsf{s}}^{-}}\ (\mathrm{C}\multimap)$$

$$\frac{\Xi \vdash \tau\ \mathsf{type}_{\mathsf{f}} \quad \Xi \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}}{\Xi \vdash \tau \wedge A\ \mathsf{type}_{\mathsf{s}}^{+}}\ (\mathrm{C}\wedge) \qquad \frac{\Xi \vdash \tau\ \mathsf{type}_{\mathsf{f}} \quad \Xi \vdash A\ \mathsf{type}_{\mathsf{s}}^{-}}{\Xi \vdash \tau \supset A\ \mathsf{type}_{\mathsf{s}}^{-}}\ (\mathrm{C}\supset)$$

$$\frac{\Xi \vdash A_i\ \mathsf{type}_{\mathsf{s}} \quad (0 \le i \le n)}{\Xi \vdash \{a_0 : A_0 \leftarrow a_1 : A_1, \ldots, a_n : A_n\}\ \mathsf{type}_{\mathsf{f}}}\ (\mathrm{T}\{\}) \qquad \frac{\Xi \vdash \tau\ \mathsf{type}_{\mathsf{f}} \quad \Xi \vdash \sigma\ \mathsf{type}_{\mathsf{f}}}{\Xi \vdash \tau \to \sigma\ \mathsf{type}_{\mathsf{f}}}\ (\mathrm{T}\to)$$

The following two rules are not part of Polarized SILL proper, and they will only be used in chapter 8 as technical tools to define the denotations of recursion. There is an instance of each rule for each $n \in \mathbb{N}$:

$$\frac{\Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}}{\Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+}}\ (\mathrm{C}\rho_n^{+}) \qquad \frac{\Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{-} \vdash A\ \mathsf{type}_{\mathsf{s}}^{-}}{\Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{-}}\ (\mathrm{C}\rho_n^{-})$$

5.B. Complete Listing of Multiset-Rewriting Rules for Polarized SILL

$$\mathsf{msg}(a, m^{+}), \mathsf{proc}(b, a \to b) \to \mathsf{msg}(b, [b/a]m^{+}) \tag{64}$$

$$\mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m_{b,c}^{-}) \to \mathsf{msg}(c, [a/b]m_{b,c}^{-}) \tag{65}$$

$$\forall \Delta_1, \Delta_2, c.\mathsf{proc}(c, a \leftarrow P;\ Q) \to \exists b.\mathsf{proc}(b, [b/a]P), \mathsf{proc}(c, [b/a]Q) \tag{66}$$

$$\forall a, \overline{a_i}.\mathbf{eval}(M, a \leftarrow \{P\} \leftarrow \overline{a_i}), \mathsf{proc}(a, a \leftarrow \{M\} \leftarrow \overline{a_i}) \to \mathsf{proc}(a, P) \tag{73}$$

$$\forall a.\mathsf{proc}(a, \mathsf{close}\ a) \to \mathsf{msg}(a, \mathsf{close}\ a) \tag{68}$$

$$\forall \Delta, a, c.\mathsf{msg}(a, \mathsf{close}\ a), \mathsf{proc}(c, \mathsf{wait}\ a;\ P) \to \mathsf{proc}(c, P) \tag{67}$$

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{shift};\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a) \tag{80}$$

$$\forall \Delta, a, d, c.\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{shift};\ d \leftarrow a), \mathsf{proc}(c, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [d/a]P) \tag{81}$$

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P), \mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \to d) \to \mathsf{proc}(d, [d/a]P)$$

$$\forall \Delta, a, c.\mathsf{proc}(c, \mathsf{send}\ a\ \mathsf{shift};\ P) \to \exists d.\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{shift};\ a \to d), \mathsf{proc}(c, [d/a]P)$$

$$\forall \Delta, a.\mathsf{proc}(a, a.k;\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, a.k;\ d \to a) \tag{78}$$

$$\forall a, d, \Delta, c.\mathsf{msg}(a, a.k;\ d \to a), \mathsf{proc}(c, \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L}) \to \mathsf{proc}(c, [d/a]P_k) \tag{79}$$

$$\forall a, d, \Delta.\mathsf{proc}(a, \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L}), \mathsf{msg}(d, a.k;\ a \leftarrow d) \to \mathsf{proc}(d, [d/a]P_k) \tag{86}$$

$$\forall \Delta, a, c.\mathsf{proc}(c, a.k;\ P) \to \exists d.\mathsf{msg}(d, a.k;\ a \leftarrow d), \mathsf{proc}(c, [d/a]P) \tag{87}$$

$$\forall \Delta, b, a.\mathsf{proc}(a, \mathsf{send}\ a\ b;\ P) \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ b;\ d \to a) \tag{69}$$

$$\forall a, e, d, \Delta, c.\mathsf{msg}(a, \mathsf{send}\ a\ e;\ d \to a), \mathsf{proc}(c, b \leftarrow \mathsf{recv}\ a;\ P) \to \mathsf{proc}(c, [e, d/b, a]P) \tag{70}$$

$$\forall a, e, d, \Delta, c.\mathsf{proc}(a, b \leftarrow \mathsf{recv}\ a;\ P), \mathsf{msg}(d, \mathsf{send}\ a\ e;\ a \leftarrow d) \to \mathsf{proc}(d, [e, d/b, a]P) \tag{71}$$

$$\forall \Delta, b, a, c.\mathsf{proc}(c, \mathsf{send}\ a\ b;\ P) \to \exists d.\mathsf{msg}(d, \mathsf{send}\ a\ b;\ a \leftarrow d), \mathsf{proc}(c, [d/a]P) \tag{72}$$

$$\forall a, \Delta.\mathbf{eval}(M, v), \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) \to \\ \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a) \tag{74}$$

$$\forall \Delta, a, d, c.\mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a), \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) \to \\ \to \mathsf{proc}(c, [d, v/a, x]P) \tag{75}$$

$$\forall \Delta, a, d, c.\mathsf{proc}(a, x \leftarrow \mathsf{input}\ a;\ P), \mathsf{msg}(d, \_ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d) \to \\ \to \mathsf{proc}(d, [d, v/a, x]P) \tag{76}$$

$$\forall a, \Delta.\mathbf{eval}(M, v), \mathsf{proc}(c, \_ \leftarrow \mathsf{output}\ a\ M;\ P) \rightarrow \\ \rightarrow \exists d.\mathsf{msg}(d, \_ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d), \mathsf{proc}(c, [d/a]P) \tag{77}$$

$$\forall \Delta, a.\mathsf{proc}(a, \mathsf{send}\ a\ \mathsf{unfold};\ P) \rightarrow \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \rightarrow a) \tag{82}$$

$$\forall \Delta, a, d.\mathsf{msg}(a, \mathsf{send}\ a\ \mathsf{unfold};\ d \rightarrow a), \mathsf{proc}(c, \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P) \rightarrow \mathsf{proc}(c, [d/a]P) \tag{83}$$

$$\forall \Delta, a, d.\mathsf{proc}(a, \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P), \mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{unfold};\ a \leftarrow d) \rightarrow \mathsf{proc}(d, [d/a]P) \tag{88}$$

$$\forall \Delta, a, c.\mathsf{proc}(c, \mathsf{send}\ a\ \mathsf{unfold};\ P) \rightarrow \exists d.\mathsf{msg}(d, \mathsf{send}\ a\ \mathsf{unfold};\ a \leftarrow d), \mathsf{proc}(c, [d/a]P) \tag{89}$$

CHAPTER 6

Observed Communication Semantics

A longstanding idea in concurrency theory is that processes can only interact with their environments through communication, and that we can only observe systems by communicating with them. Indeed, as far back as 1980, Milner [Mil80, p. 2] wrote "we suppose that the only way to observe a [concurrent] system is to communicate with it".

In this chapter, we make the above intuitions mathematically rigorous by giving Polarized SILL an *observed communication semantics*. Observed communication semantics, introduced by Atkey [Atk17], define the meaning of a process to be the communications observed on its channels.

In section 6.1, we make the notion of a session-typed communication explicit. We endow session-typed communications with a notion of approximation. This approximation will be used later to relate various notions of equivalence. We also characterize infinite communications by their finite approximations.

In section 6.2 we show how to observe communications on free channels of configurations. We first do so using a coinductively defined judgment that observes communications on fair executions. We show that the choice of fair trace does not matter. We also show that we can instead consider only finite prefixes of fair executions. The communications observed on the finite prefixes approximate and determine those observed on the complete fair execution.

We generalize from single channels to sets of channels in section 6.3.

6.1. Session-Typed Communications *qua* Communications

We begin by defining session-typed communications. Let a communication v be a (potentially infinite) tree generated by the following grammar, where k ranges over labels and f ranges over functional values such that $\cdot \Vdash f : \tau$ for some τ. We explain these communications v below when we associate them with session types.

$$
\begin{array}{rll}
v, v' ::= & \bot & \text{empty communication} \\
\mid & \mathsf{close} & \text{close message} \\
\mid & (\mathsf{unfold}, v) & \text{unfolding message} \\
\mid & (k, v) & \text{choice message} \\
\mid & (v, v') & \text{channel message} \\
\mid & (\mathsf{shift}, v) & \text{shift message} \\
\mid & (\mathsf{val}\ f, v) & \text{functional value message}
\end{array}
$$

The judgment $v \ \varepsilon \ A$ means that the syntactic communication v has closed type A. It is coinductively defined by the following rules:

$$\frac{\cdot \vdash A\ \mathsf{type}_s}{\bot \ \varepsilon \ A}\ (\text{C-}\bot) \qquad \frac{}{\mathsf{close} \ \varepsilon \ \mathbf{1}}\ (\text{C-}\mathbf{1}) \qquad \frac{v \ \varepsilon \ [\rho\alpha.A/\alpha]A}{(\mathsf{unfold}, v) \ \varepsilon \ \rho\alpha.A}\ (\text{C-}\rho)$$

$$\frac{v_k \ \varepsilon \ A_k \quad (k \in L)}{(k, v_k) \ \varepsilon \ \oplus\{l : A_l\}_{l \in L}}\ (\text{C-}\oplus) \qquad \frac{v_k \ \varepsilon \ A_k \quad (k \in L)}{(k, v_k) \ \varepsilon \ \&\{l : A_l\}_{l \in L}}\ (\text{C-}\&)$$

$$\frac{v \ \varepsilon \ A \quad v' \ \varepsilon \ B}{(v, v') \ \varepsilon \ A \otimes B}\ (\text{C-}\otimes) \qquad \frac{v \ \varepsilon \ A \quad v' \ \varepsilon \ B}{(v, v') \ \varepsilon \ A \multimap B}\ (\text{C-}\multimap)$$

$$\frac{v \ \varepsilon \ A}{(\mathsf{shift}, v) \ \varepsilon \ \downarrow A}\ (\text{C-}\downarrow) \qquad \frac{v \ \varepsilon \ A}{(\mathsf{shift}, v) \ \varepsilon \ \uparrow A}\ (\text{C-}\uparrow)$$

$$\frac{\cdot \Vdash f : \tau \quad v \ \varepsilon \ A}{(\mathsf{val}\ f, v) \ \varepsilon \ \tau \wedge A}\ (\text{C-}\wedge) \qquad \frac{\cdot \Vdash f : \tau \quad v \ \varepsilon \ A}{(\mathsf{val}\ f, v) \ \varepsilon \ \tau \supset A}\ (\text{C-}\supset)$$

Every closed session type A has an empty communication $\bot$ representing the absence of communication of that type. The communication close represents the close message. A communication of type $\oplus\{l : A_l\}_{l \in L}$ or $\&\{l : A_l\}_{l \in L}$ is a label $k \in L$ followed by a communication v_k of type A_k, whence the communication (k, v_k). Though by itself the communication (k, v_k) does not capture the direction in which the label k travelled, this poses no problem to our development: we almost never consider communications without an associated session type, and the polarity of the type specifies the direction in which k travels. We cannot directly observe channels, but we can observe communications over channels. Consequently, we observe a communication of type $A \otimes B$ or $A \multimap B$ as a pair (v, v') of communications v of type A and v' of type B. This is analogous to the semantics of $A \otimes B$ in the "folklore" relational semantics of classical linear logic proofs [Atk17; Bar91]. A communication of type $\rho\alpha.A$ is an unfold message followed by a communication of type $[\rho\alpha.A/\alpha]A$. A communication of type $\tau \wedge A$ or $\tau \supset A$ is a value f of type τ followed by a communication of type A.

We will consider various relations on communications, and we expect these to be "type-indexed":

Definition 6.1.1. A **type-indexed relation $\mathfrak{R}$ on communications** is a family of relations $(\mathfrak{R}_A)_A$ indexed by session types A, where $(v, w) \in \mathfrak{R}_A$ only if $v \ \varepsilon \ A$ and $w \ \varepsilon \ A$. In this case, we write $v \ \mathfrak{R} \ w \ \varepsilon \ A$. ◀

Given some relation $\leqslant$ on terms, we can endow session-typed communications with a notion of simulation $\lesssim/\leqslant$. Intuitively, $u \lesssim/\leqslant w$ means that u approximates w, or that w carries at least as much information as u. Functional values aside, it suggests that u is a potentially incomplete version of w. In this regard, it is analogous to the ordering on domains of lazy natural numbers [Fre90; Esc93]. Though we intend for $\leqslant$ to be a preorder, it is not required to be one. This relaxation is for purely technical reasons: it simplifies the task of relating $\lesssim/\leqslant$ to other relations on communications.

Definition 6.1.2. Let $\leqslant$ be a type-indexed relation on terms. **Communication simulation $\lesssim$ modulo $\leqslant$** is the largest type-indexed family $\lesssim/\leqslant$ of relations $(\lesssim_A)_A$ on session-typed communications defined

by the following rules. When $\leqslant$ is clear from context, we write $\lesssim$ for $\lesssim/\leqslant$.

$$\frac{w \mathrel{\varepsilon} A}{\bot \lesssim w \mathrel{\varepsilon} A}\ (\text{CS-}\bot) \qquad \frac{}{\mathsf{close} \lesssim \mathsf{close} \mathrel{\varepsilon} \mathbf{1}}\ (\text{CS-}\mathbf{1}) \qquad \frac{v \lesssim w \mathrel{\varepsilon} [\rho\alpha.A/\alpha]A}{(\mathsf{unfold}, v) \lesssim (\mathsf{unfold}, w) \mathrel{\varepsilon} \rho\alpha.A}\ (\text{CS-}\rho)$$

$$\frac{v_k \lesssim w_k \mathrel{\varepsilon} A_k \quad (k \in L)}{(k, v_k) \lesssim (k, w_k) \mathrel{\varepsilon} \oplus\{l : A_l\}_{l \in L}}\ (\text{CS-}\oplus) \qquad \frac{v_k \lesssim w_k \mathrel{\varepsilon} A_k \quad (k \in L)}{(k, v_k) \lesssim (k, w_k) \mathrel{\varepsilon} \&\{l : A_l\}_{l \in L}}\ (\text{CS-}\&)$$

$$\frac{v \lesssim w \mathrel{\varepsilon} A \quad v' \lesssim w' \mathrel{\varepsilon} B}{(v, v') \lesssim (w, w') \mathrel{\varepsilon} A \otimes B}\ (\text{CS-}\otimes) \qquad \frac{v \lesssim w \mathrel{\varepsilon} A \quad v' \lesssim w' \mathrel{\varepsilon} B}{(v, v') \lesssim (w, w') \mathrel{\varepsilon} A \multimap B}\ (\text{CS-}\multimap)$$

$$\frac{v \lesssim w \mathrel{\varepsilon} A}{(\mathsf{shift}, v) \lesssim (\mathsf{shift}, w) \mathrel{\varepsilon} \downarrow A}\ (\text{CS-}\downarrow) \qquad \frac{v \lesssim w \mathrel{\varepsilon} A}{(\mathsf{shift}, v) \lesssim (\mathsf{shift}, w) \mathrel{\varepsilon} \uparrow A}\ (\text{CS-}\uparrow)$$

$$\frac{\cdot \Vdash f \leqslant f' : \tau \quad v \lesssim w \mathrel{\varepsilon} A}{(\mathsf{val}\, f, v) \lesssim (\mathsf{val}\, f', w) \mathrel{\varepsilon} \tau \wedge A}\ (\text{CS-}\wedge) \qquad \frac{\cdot \Vdash f \leqslant f' : \tau \quad v \lesssim w \mathrel{\varepsilon} A}{(\mathsf{val}\, f, v) \lesssim (\mathsf{val}\, f', w) \mathrel{\varepsilon} \tau \supset A}\ (\text{CS-}\supset)$$

◀

Remark 6.1.3. We do not ask for $\lesssim$ to be a partial order. This is because antisymmetry forces communication equivalence for communications of type $\tau \wedge A$ to hold only when the transmitted values are equal on the nose. This is too fine of an equivalence: we would like to allow communications of type $\tau \wedge A$ to be "equivalence" whenever the values of type τ are in some sense "equivalent", without insisting that that equivalence be syntactic equality.

Proposition 6.1.4. *The function $\lesssim/(-)$ is monotone, ω-continuous, and ω-cocontinuous. The relation $\lesssim/\leqslant$ is respectively reflexive or transitive whenever $\leqslant$ is reflexive or transitive. It is a type-indexed relation.*

Proof. We begin by showing that the function is well-defined. Let $\mathcal{R}$ be the complete lattice of all type-indexed relations on session-typed communications, and let $\mathcal{F}$ be the complete lattice of all type-indexed relations on functional terms. For each $\mathfrak{F} \in \mathcal{F}$, the above rules define a rule functional $\Phi(\mathfrak{F}, -) : \mathcal{R} \to \mathcal{R}$. It is ω-cocontinuous by [San12, Theorem 2.9.4]. It extends to a monotone function $\Phi : \mathcal{F} \times \mathcal{R} \to \mathcal{R}$. We observe that $\lesssim/(-)$ is given by $\left(\left(\Phi^{\mathrm{op}}\right)^{\dagger}\right)^{\mathrm{op}} : \mathcal{F} \to \mathcal{R}$. Indeed, the greatest fixed point of $\Phi(\mathfrak{F}, -)$ is the initial fixed point $\left(\Phi^{\mathrm{op}}\right)^{\dagger}(\mathfrak{F})$ of $\Phi^{\mathrm{op}}(\mathfrak{F}, -) : \mathcal{R}^{\mathrm{op}} \to \mathcal{R}^{\mathrm{op}}$, where $\Phi^{\mathrm{op}} : \mathcal{F}^{\mathrm{op}} \times \mathcal{R}^{\mathrm{op}} \to \mathcal{R}^{\mathrm{op}}$, and $\left(\Phi^{\mathrm{op}}\right)^{\dagger} : \mathcal{F}^{\mathrm{op}} \to \mathcal{R}^{\mathrm{op}}$ is given by proposition 4.3.1. By the same proposition, $\lesssim/(-)$ is monotone, ω-continuous, and ω-cocontinuous.

We use the coinduction proof principle to show that $\lesssim$ is reflexive. Let Δ be the identity relation on session-typed communications. A case analysis on the rules shows that $\Delta \subseteq \Phi(\leqslant, \Delta)$. Because $\lesssim/\leqslant$ is the greatest post-fixed point of $\Phi(\leqslant, -)$, we conclude that it contains Δ, i.e., that it is reflexive.

Assume now that $\leqslant$ is a preorder. We use the same technique to show that $\lesssim/\leqslant$ is a preorder. Now let $\lesssim^{+}$ be the transitive closure of $\lesssim/\leqslant$. Recall that the transitive closure $\mathfrak{R}^{+}$ of a relation $\mathfrak{R}$ can be calculated by

$$\mathfrak{R}^{+} = \bigcup_{n=1}^{\infty} \mathfrak{R}^{n},$$

where $\mathfrak{R}^{n}$ is the n-fold composition of $\mathfrak{R}$ with itself. The functional $\Phi(\leqslant, -)$ is ω-continuous by [San12, Exercise 2.9.2]. In particular, this implies that

$$\Phi(\leqslant, \mathfrak{R}^{+}) = \bigcup_{n=1}^{\infty} \Phi(\leqslant, \mathfrak{R}^{n}).$$

Thus, to show that $\mathfrak{R}^{+} \subseteq \Phi(\leqslant, \mathfrak{R}^{+})$, it is sufficient to show that $\mathfrak{R}^{n} \subseteq \Phi(\leqslant, \mathfrak{R}^{n})$ for all n. Recall that $\lesssim$ is the greatest post-fixed point of $\Phi(\leqslant, \mathfrak{R}^{n})$. This means that to show that $\lesssim$ is transitive, i.e., $\lesssim^{+} \subseteq \lesssim$, it is sufficient to show that $\lesssim^{+} \subseteq \Phi(\leqslant, \lesssim^{+})$. We proceed by case analysis on n to show that $\lesssim^{n} \subseteq \Phi(\leqslant, \lesssim^{n})$. The case $n = 1$ is immediate by definition of $\lesssim$ as the greatest fixed point of $\Phi(\leqslant, -)$. We now show the case $n = m + 1$. Assume that $u \lesssim^{n} w \mathrel{\varepsilon} A$ because $u \lesssim v \mathrel{\varepsilon} A$ and $v \lesssim^{m} w \mathrel{\varepsilon} A$. We show that $u\ \Phi(\leqslant, \lesssim^{n})\ w \mathrel{\varepsilon} A$. We proceed by case analysis on the rule that formed $u \lesssim v \mathrel{\varepsilon} A$, giving several illustrative cases:

Case (CS-$\perp$): Then $u = \perp$, and $w \varepsilon A$ because we assumed that $\leqslant$ was a type-indexed relation on session-typed communications. So $u\ \Phi(\leqslant, \leqslant^n)\ w \varepsilon A$ thanks to (CS-$\perp$).

Case (CS-$\oplus$): Then $A = \oplus\{l : A_l\}_{l\in L}$, $u = (k, u')$, and $v = (k, v')$ for some u' and v'. By hypothesis, $u' \leqslant v' \varepsilon A_k$. And induction on m reveals that $w = (k, w')$ for some $v' \leqslant^m w' \varepsilon A_k$. So $u' \leqslant^n w' \varepsilon A_k$. By (CS-$\oplus$), we then get $u\ \Phi(\leqslant, \leqslant^n)\ w \varepsilon A$ as desired.

Case (CS-$\wedge$): Then $A = \tau \wedge B$, $u = (\mathsf{val}\ f, u')$, and $v = (\mathsf{val}\ g, v')$ for some f, g, u', and v'. By hypothesis, $u' \leqslant v' \varepsilon B$ and $\cdot \Vdash f \leqslant g : \tau$. And induction on m reveals that $w = (\mathsf{val}\ h, w')$ for some $v' \leqslant^m w' \varepsilon B$ and $\cdot \Vdash g \leqslant^m h : \tau$. So $\cdot \Vdash f \leqslant h : \tau$ and $u' \leqslant^n w' \varepsilon B$. By (CS-$\wedge$), we then get $u\ \Phi(\leqslant, \leqslant^n)\ w \varepsilon A$ as desired. □

Definition 6.1.5. Let $\equiv$ be a type-indexed relation on terms. **Communication equivalence $\doteq$ modulo $\equiv$**, written $\doteq/\equiv$, is given by $v \doteq/\equiv w \varepsilon A$ if and only if both $v \leqslant/\equiv w \varepsilon A$ and $w \leqslant/\equiv v \varepsilon A$. When $\equiv$ is clear from context, we write $\doteq$ for $\doteq/\equiv$. ◂

Proposition 6.1.6. *Communication equivalence $\doteq/\equiv$ is a type-indexed relation. It is an equivalence relation whenever $\equiv$ is a preorder.*

Proof. It follows from proposition 6.1.4 that it is type-indexed. Assume now that $\equiv$ is a preorder. By proposition 6.1.4, $\leqslant/\equiv$ is a preorder. Then by definition, $\doteq/\equiv$ is the intersection of a preorder and its opposite. But in general $< \cap <^{\mathrm{op}}$ is an equivalence relation whenever $<$ is a preorder. We conclude that $\doteq/\equiv$ is an equivalence relation. □

Communication equivalence modulo $=$ holds if and only if two communications are equal on the nose:

Proposition 6.1.7. *For all A, $u \doteq/= v \varepsilon A$ if and only if $u = v$.*

Proof. Necessity is immediate by reflexivity of $\doteq/=$. Sufficiency comes from recognizing $\doteq/=$ as the notion of bisimulation given by the coinductive definition of $w \varepsilon A$, and that by [JR12, Theorem 2.7.2], bisimilar elements of the terminal coalgebra are equal. □

Proposition 6.1.8. *"Communication simulation modulo" and "communication equivalence modulo" are related by the identity $\left(\leqslant/\leqslant \cap (\leqslant/\leqslant)^{\mathrm{op}}\right) = \left(\doteq/(\leqslant \cap \leqslant^{\mathrm{op}})\right)$.*

Proof. Let Φ be the functional defining $\leqslant/(-)$, and set $I = \leqslant \cap \leqslant^{\mathrm{op}}$. Observe for all relations $\mathfrak{X}$, $\mathfrak{Y}$, $\mathfrak{Z}$ that $\Phi(\mathfrak{X} \cap \mathfrak{Y}, \mathfrak{Z}) = \Phi(\mathfrak{X}, \mathfrak{Z}) \cap \Phi(\mathfrak{Y}, \mathfrak{Z})$. We compute, where we use the syntax $\nu X.F(X)$ for the greatest fixed point of F, that:

$$\begin{aligned}
&\doteq/I \\
&= (\leqslant/I) \cap (\leqslant/I)^{\mathrm{op}} \\
&= (\nu\mathfrak{V}.\Phi(I, \mathfrak{V})) \cap (\nu\mathfrak{V}.\Phi(I, \mathfrak{V}))^{\mathrm{op}} \\
&= (\nu\mathfrak{V}.\Phi(I, \mathfrak{V})) \cap \nu\mathfrak{V}.(\Phi(I, \mathfrak{V}))^{\mathrm{op}} \\
&= \nu\mathfrak{V}.\Phi(\leqslant, \mathfrak{V}) \cap \Phi(\leqslant^{\mathrm{op}}, \mathfrak{V}) \cap (\Phi(\leqslant, \mathfrak{V}))^{\mathrm{op}} \cap (\Phi(\leqslant^{\mathrm{op}}, \mathfrak{V}))^{\mathrm{op}},
\end{aligned}$$

and analogously,

$$\begin{aligned}
&\left(\leqslant/\leqslant \cap (\leqslant/\leqslant)^{\mathrm{op}}\right) \\
&= \nu\mathfrak{V}.\Phi(\leqslant, \mathfrak{V}) \cap (\Phi(\leqslant, \mathfrak{V}))^{\mathrm{op}}.
\end{aligned}$$

To show that the fixed points are equal, it is sufficient to show that they have the same post-fixed points. Set

$$\begin{aligned}
L(\mathfrak{V}) &= \Phi(\leqslant, \mathfrak{V}) \cap \Phi(\leqslant^{\mathrm{op}}, \mathfrak{V}) \cap (\Phi(\leqslant, \mathfrak{V}))^{\mathrm{op}} \cap (\Phi(\leqslant^{\mathrm{op}}, \mathfrak{V}))^{\mathrm{op}}, \\
R(\mathfrak{V}) &= \Phi(\leqslant, \mathfrak{V}) \cap (\Phi(\leqslant, \mathfrak{V}))^{\mathrm{op}}.
\end{aligned}$$

Clearly every post-fixed point of L is a post-fixed point of R: $L(\mathfrak{V}) \subseteq R(\mathfrak{V})$ for all $\mathfrak{V}$. Conversely, assume that $\mathfrak{V}$ is a post-fixed point of R, i.e., $\mathfrak{V} \subseteq R(\mathfrak{V})$. We show that $\mathfrak{V}$ is a post-fixed point of L by showing that $R(\mathfrak{V}) \subseteq L(\mathfrak{V})$. Assume that $v\ R(\mathfrak{V})\ w \varepsilon A$. Then $v\ \Phi(\leqslant, \mathfrak{V})\ w \varepsilon A$. A case

analysis then reveals that $v\ (\Phi(\leqslant, \mathfrak{V}))^{\mathrm{op}}\ w\ \varepsilon\ A$ by the same rule. The result follows by case analysis on this rule. We give a few illustrative cases.

Case (CS-$\bot$)**:** Then $v = w = \bot$, and a straightforward check gives $\bot\ L(\mathfrak{V})\ \bot\ \varepsilon\ A$.

Case (CS-$\oplus$)**:** Then $v = (k, v')$, $w = (k, w')$, $v'\ \mathfrak{V}\ w'\ \varepsilon\ A_k$, and $w'\ \mathfrak{V}\ v'\ \varepsilon\ A_k$. A straightforward check again gives $v\ L(\mathfrak{V})\ w\ \varepsilon\ A$.

Case (CS-$\wedge$)**:** Then $A = \tau \wedge B$, $v = (\mathsf{val}\ f, v')$, $w = (\mathsf{val}\ g, w')$, $f \leqslant g$, $g \leqslant f$, $v'\ \mathfrak{V}\ w'\ \varepsilon\ B$, and $w'\ \mathfrak{V}\ v'\ \varepsilon\ B$. A straightforward check gives that $v\ L(\mathfrak{V})\ w\ \varepsilon\ A$. □

We now show that communications are uniquely determined by their finite approximations. This opens the door to reasoning about $\lesssim/\leqslant$ using inductive techniques.

Definition 6.1.9. The **height n approximation** $\lfloor w \rfloor_n$ of a communication w is defined by induction on n and recursion on w:

$$\lfloor w \rfloor_0 = \bot \qquad \lfloor \bot \rfloor_{n+1} = \bot$$
$$\lfloor \mathsf{close} \rfloor_{n+1} = \mathsf{close} \qquad \lfloor (\mathsf{val}\ f, v) \rfloor_{n+1} = (\mathsf{val}\ f, \lfloor v \rfloor_n)$$
$$\lfloor (k, v) \rfloor_{n+1} = (k, \lfloor v \rfloor_n) \qquad \lfloor (u, v) \rfloor_{n+1} = (\lfloor u \rfloor_n, \lfloor v \rfloor_n)$$
$$\lfloor (\mathsf{shift}, v) \rfloor_{n+1} = (\mathsf{shift}, \lfloor v \rfloor_n) \qquad \lfloor (\mathsf{unfold}, v) \rfloor_{n+1} = (\mathsf{unfold}, \lfloor v \rfloor_n)$$ ◂

Proposition 6.1.10. *If $w\ \varepsilon\ A$, then $\lfloor w \rfloor_n\ \varepsilon\ A$ for all n.*

Proof. By induction on n. The base case is immediate. The inductive step follows by a case analysis on the rule used to form $w\ \varepsilon\ A$. □

Proposition 6.1.11. *For all reflexive $\leqslant$, all n, and all $w\ \varepsilon\ A$, $\lfloor w \rfloor_n \lesssim/\leqslant \lfloor w \rfloor_{n+1}\ \varepsilon\ A$.*

Proof. By induction on n. The base case is given by (CS-$\bot$). The inductive step is given by case analysis on $w\ \varepsilon\ A$. Reflexivity of $\leqslant$ is required for the cases $A = \tau \wedge B$ and $A = \tau \supset B$. □

Proposition 6.1.12. *For all $w\ \varepsilon\ A$ and $u\ \varepsilon\ A$, $u \lesssim/\leqslant w\ \varepsilon\ A$ if and only if, for all n, $\lfloor u \rfloor_n \lesssim/\leqslant w\ \varepsilon\ A$.*

Proof. We proceed by induction on n to show that for all $n \in \mathbb{N}$ and for all $w\ \varepsilon\ A$ and $u\ \varepsilon\ A$, $u \lesssim/\leqslant w\ \varepsilon\ A$ implies $\lfloor u \rfloor_n \lesssim/\leqslant w\ \varepsilon\ A$. The base case is immediate by (CS-$\bot$). Assume the result for some n. We show that $\lfloor u \rfloor_{n+1} \lesssim/\leqslant w\ \varepsilon\ A$ by case analysis on the rule used to form $u \lesssim/\leqslant w\ \varepsilon\ A$. We give two illustrative cases; the rest follow by analogy.

Case (CS-$\bot$)**:** Then $u = \bot$ and $\lfloor u \rfloor_{n+1} = \bot$. We are done by (CS-$\bot$).

Case (CS-$\wedge$)**:** Then $A = \tau \wedge B$, $u = (\mathsf{val}\ f, u')$, and $w = (\mathsf{val}\ g, w')$ with $\cdot \Vdash f \leqslant g : \tau$ and $u' \lesssim/\leqslant w'\ \varepsilon\ B$. By definition, $\lfloor u \rfloor_{n+1} = (\mathsf{val}\ f, \lfloor u' \rfloor_n)$. By the induction hypothesis, $\lfloor u' \rfloor_n \lesssim/\leqslant w'\ \varepsilon\ B$. By (CS-$\wedge$), $(\mathsf{val}\ f, \lfloor u' \rfloor_n) \lesssim/\leqslant (\mathsf{val}\ g, w')\ \varepsilon\ A$ as desired.

To show the converse, let T be the set of triples $\{(u, w, A) \mid \forall n \in \mathbb{N}\ .\ \lfloor u \rfloor_n \lesssim/\leqslant w\ \varepsilon\ A\}$. We want to show that if $(u, w, A) \in T$, then $u \lesssim/\leqslant w\ \varepsilon\ A$. By the coinduction proof principle [San12, p. 49], it is sufficient to show that T is "closed backwards" under the rules defining $\lesssim/\leqslant$. Let $(u, v, A) \in T$ be arbitrary. We proceed by case analysis on u and A to show that there is a rule whose conclusion is (u, v, A) and whose premises are in T. If $u = \bot$, then we are done by (CS-$\bot$), so assume that $u \neq \bot$. We proceed by case analysis on A. We show two cases; the rest follow by analogy.

Case $A = \mathbf{1}$**:** The only possible value for u and v is $u = v = \mathsf{close}$. So $(u, w, \mathbf{1}) \in T$ by (CS-$\mathbf{1}$).

Case $A = \tau \wedge B$**:** Then $u = (\mathsf{val}\ f, u')$ for some value $\cdot \Vdash f : \tau$ and some $u'\ \varepsilon\ B$, and $v = (\mathsf{val}\ g, v')$ for some value $\cdot \Vdash g : \tau$ and some $v'\ \varepsilon\ B$. By assumption, $\lfloor (\mathsf{val}\ f, u') \rfloor_n \lesssim/\leqslant (\mathsf{val}\ g, v')\ \varepsilon\ A$ for all n. By inversion, for all $n = m + 1 \geq 1$, the last rule in the derivation must have been (CS-$\wedge$) with $\lfloor u' \rfloor_m \lesssim/\leqslant v'\ \varepsilon\ B$ and with its side condition $\cdot \Vdash f \leqslant g : \tau$ satisfied. So $(u', w', B) \in T$. Then $(u, w, A) \in T$ by (CS-$\wedge$) with the premise $(u', w', B) \in T$ and the side condition $\cdot \Vdash f \leqslant g : \tau$. □

Definition 6.1.13. Where $\leqslant$ is a type-indexed preorder on functional values, let $\langle\!\langle A \rangle\!\rangle_\leqslant$ be the set of communications $w\ \varepsilon\ A$ ordered by the preorder $\lesssim/\leqslant$. ◂

COROLLARY 6.1.14. *If $\leqslant$ is a preorder, then for all $w \varepsilon A$, w is a[1] least upper bound of $(\lfloor w \rfloor_n)_{n\in\mathbb{N}}$ in the preorder $\langle\!\langle A \rangle\!\rangle_{\leqslant}$.*

6.2. Session-Typed Communications on Single Channels

In this section, we show how to observe session-typed communications $v \varepsilon A$ on a single channel c in a trace T. We capture these observations using a coinductively defined judgment $T \rightsquigarrow v \varepsilon A \mid c$. This judgment defines a total function from free channel names in T to session-typed communications $v \varepsilon A$. We show that the type of the observed communications agrees with the type of the channel, i.e., that $T \rightsquigarrow v \varepsilon A \mid c$ implies $T \vdash c : A$.[2] We will also show that the communication observed on c is *independent* of the choice of trace T, provided that T is fair. Finally, we show that the communications observed from finite prefixes of T both approximate and determine the observations on the entirety of T.

Given a trace $T = (M_0, (r_i; (\theta_i, \xi_i))_i)$, we write $\mathcal{T}$ for the support of T, that is, $x \in \mathcal{T}$ if and only if $x \in M_i$ for some i. The judgment $T \rightsquigarrow v \varepsilon A \mid c$ is coinductively defined by the following rules, i.e., it is the largest set of triples (v, c, A) closed under the following rules.

We observe no communications on a channel c if and only if c carried no message. Subject to the side condition that $c \neq \mathsf{cc}(\mathsf{msg}(d, m))$ for all $\mathsf{msg}(d, m) \in \mathcal{T}$, we have the rule

$$\frac{T \vdash c : A}{T \rightsquigarrow \bot \varepsilon A \mid c}\ (\text{O-}\bot) \quad \text{whenever } \forall \mathsf{msg}(d, m) \in \mathcal{T}.\ c \neq \mathsf{cc}(\mathsf{msg}(d, m)).$$

We observe a close message on c if and only if the close message was sent on c:

$$\frac{\mathsf{msg}(c, \mathsf{close}\ c) \in \mathcal{T}}{T \rightsquigarrow \mathsf{close} \varepsilon \mathbf{1} \mid c}\ (\text{O-}\mathbf{1})$$

We observe label transmission as labelling communications on the continuation channel. We rely on the judgment $T \vdash c : \oplus\{l : A_l\}_{l\in L}$ or $T \vdash c : \&\{l : A_l\}_{l\in L}$ to determine the type of c:

$$\frac{\mathsf{msg}(c, c.l;\ d \to c) \in \mathcal{T} \quad T \rightsquigarrow v \varepsilon A_l \mid d \quad T \vdash c : \oplus\{l : A_l\}_{l\in L}}{T \rightsquigarrow (l, v) \varepsilon \oplus\{l : A_l\}_{l\in L} \mid c}\ (\text{O-}\oplus)$$

$$\frac{\mathsf{msg}(d, c.l;\ c \leftarrow d) \in \mathcal{T} \quad T \rightsquigarrow v \varepsilon A_l \mid d \quad T \vdash c : \&\{l : A_l\}_{l\in L}}{T \rightsquigarrow (l, v) \varepsilon \&\{l : A_l\}_{l\in L} \mid c}\ (\text{O-}\&)$$

As described above, we observe channel transmission as pairing of communications:

$$\frac{\mathsf{msg}(c, \mathsf{send}\ c\ a;\ d \to c) \in \mathcal{T} \quad T \rightsquigarrow u \varepsilon A \mid a \quad T \rightsquigarrow v \varepsilon B \mid d}{T \rightsquigarrow (u, v) \varepsilon A \otimes B \mid c}\ (\text{O-}\otimes)$$

$$\frac{\mathsf{msg}(d, \mathsf{send}\ c\ a;\ c \leftarrow d) \in \mathcal{T} \quad T \rightsquigarrow u \varepsilon A \mid a \quad T \rightsquigarrow v \varepsilon B \mid d}{T \rightsquigarrow (u, v) \varepsilon A \multimap B \mid c}\ (\text{O-}\multimap)$$

We observe the unfold and shift messages directly:

$$\frac{\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{unfold};\ d \to c) \in \mathcal{T} \quad T \rightsquigarrow v \varepsilon [\rho\alpha.A/\alpha]A \mid d}{T \rightsquigarrow (\mathsf{unfold}, v) \varepsilon \rho\alpha.A \mid c}\ (\text{O-}\rho^+)$$

$$\frac{\mathsf{msg}(d, \mathsf{send}\ c\ \mathsf{unfold};\ c \leftarrow d) \in \mathcal{T} \quad T \rightsquigarrow v \varepsilon [\rho\alpha.A/\alpha]A \mid d}{T \rightsquigarrow (\mathsf{unfold}, v) \varepsilon \rho\alpha.A \mid c}\ (\text{O-}\rho^-)$$

$$\frac{\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{shift};\ d \leftarrow c) \in \mathcal{T} \quad T \rightsquigarrow v \varepsilon A \mid d}{T \rightsquigarrow (\mathsf{shift}, v) \varepsilon \downarrow A \mid c}\ (\text{O-}\downarrow)$$

$$\frac{\mathsf{msg}(d, \mathsf{send}\ c\ \mathsf{shift};\ c \to d) \in \mathcal{T} \quad T \rightsquigarrow v \varepsilon A \mid d}{T \rightsquigarrow (\mathsf{shift}, v) \varepsilon \uparrow A \mid c}\ (\text{O-}\uparrow)$$

[1] In contrast to least upper bounds in partial orders, least upper bounds in preorders are not necessarily unique.
[2] Recall definition 5.9.2.

Finally, we observe functional values:

$$\frac{\mathsf{msg}(c, \_ \leftarrow \mathsf{output}\ c\ f;\ d \to c) \in \mathcal{T} \quad T \rightsquigarrow v\ \varepsilon\ A \mid d \quad T \vdash c : \tau \wedge A}{T \rightsquigarrow (\mathsf{val}\ f, v)\ \varepsilon\ \tau \wedge A \mid c}\ (\text{O-}\wedge)$$

$$\frac{\mathsf{msg}(d, \_ \leftarrow \mathsf{output}\ c\ f;\ c \leftarrow d) \in \mathcal{T} \quad T \rightsquigarrow v\ \varepsilon\ A \mid d \quad T \vdash c : \tau \wedge A}{T \rightsquigarrow (\mathsf{val}\ f, v)\ \varepsilon\ \tau \supset A \mid c}\ (\text{O-}\supset)$$

We set out to show that for any process trace T, the judgment $T \rightsquigarrow v\ \varepsilon\ A \mid c$ defines a total function from channel names c in T to session-typed communications $v\ \varepsilon\ A$.

We begin by showing that if $T \rightsquigarrow v\ \varepsilon\ A \mid c$, then this session-typed communication $v\ \varepsilon\ A$ is unique. We use a bisimulation approach and follow standard techniques to define bisimulations for $T \rightsquigarrow v\ \varepsilon\ A \mid c$. We interpret the premises the rules defining $T \rightsquigarrow v\ \varepsilon\ A \mid c$ that are not of the form $T \rightsquigarrow w\ \varepsilon\ B \mid d$ as side conditions, giving an instance of the rule for each such premise. For example, the rule (O-$\oplus$) should be seen as a family of rules (O-$\oplus$-c-d-l), where we have a rule

$$\frac{T \rightsquigarrow v\ \varepsilon\ A_l \mid d}{T \rightsquigarrow (l, v)\ \varepsilon\ \oplus\{l : A_l\}_{l \in L} \mid c}\ (\text{O-}\oplus\text{-}c\text{-}d\text{-}l)$$

for each $\mathsf{msg}(c, c.l;\ c \leftarrow d) \in \mathcal{T}$ such that $T \vdash c : \oplus\{l : A_l\}_{l \in L}$. A symmetric binary relation $\mathfrak{R}$ on observed communications in T is a bisimulation if:

- if $(T \rightsquigarrow \bot\ \varepsilon\ A \mid c, T \rightsquigarrow w\ \varepsilon\ A' \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow \bot\ \varepsilon\ A \mid c$ by the instance of (O-$\bot$) for $T \vdash c : A$, then $w = \bot$ and $A' = A$;
- if $(T \rightsquigarrow \mathsf{close}\ \varepsilon\ \mathbf{1} \mid c, T \rightsquigarrow w\ \varepsilon\ A \mid c) \in \mathfrak{R}$, then $w = \mathsf{close}$ and $A = \mathbf{1}$;
- if $(T \rightsquigarrow (l, v)\ \varepsilon\ \oplus\{l : A_l\}_{l \in L} \mid c, T \rightsquigarrow w\ \varepsilon\ A \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (l, v)\ \varepsilon\ \oplus\{l : A_l\}_{l \in L} \mid c$ by the instance of (O-$\oplus$) for $\mathsf{msg}(c, c.l;\ d \to c) \in \mathcal{T}$, then $w = (l, v')$ for some v', $A = \oplus\{l : A_l\}_{l \in L}$, and $(T \rightsquigarrow v\ \varepsilon\ A_l \mid d, T \rightsquigarrow w\ \varepsilon\ A_l \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (l, v)\ \varepsilon\ \&\{l : A_l\}_{l \in L} \mid c, T \rightsquigarrow w\ \varepsilon\ A \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (l, v)\ \varepsilon\ \&\{l : A_l\}_{l \in L} \mid c$ by the instance of (O-&) for $\mathsf{msg}(c, c.l;\ c \leftarrow d) \in \mathcal{T}$, then $w = (l, v')$ for some v', $A = \&\{l : A_l\}_{l \in L}$, and $(T \rightsquigarrow v\ \varepsilon\ A_l \mid d, T \rightsquigarrow w\ \varepsilon\ A_l \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (u, v)\ \varepsilon\ A \otimes B \mid c, T \rightsquigarrow w\ \varepsilon\ C \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (u, v)\ \varepsilon\ A \otimes B \mid c$ was formed by the instance of (O-$\otimes$) for $\mathsf{msg}(c, \mathsf{send}\ c\ a;\ d \to c) \in \mathcal{T}$, then $v = (u', v')$ and $C = A' \otimes B'$ for some u', v', A', B', and $(T \rightsquigarrow u\ \varepsilon\ A \mid a, T \rightsquigarrow u'\ \varepsilon\ A' \mid a) \in \mathfrak{R}$ and $(T \rightsquigarrow v\ \varepsilon\ A \mid d, T \rightsquigarrow v'\ \varepsilon\ A' \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (u, v)\ \varepsilon\ A \multimap B \mid c, T \rightsquigarrow w\ \varepsilon\ C \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (u, v)\ \varepsilon\ A \multimap B \mid c$ was formed by the instance of (O-$\multimap$) for $\mathsf{msg}(c, \mathsf{send}\ c\ a;\ c \leftarrow d) \in \mathcal{T}$, then $v = (u', v')$ and $C = A' \multimap B'$ for some u', v', A', B', and $(T \rightsquigarrow u\ \varepsilon\ A \mid a, T \rightsquigarrow u'\ \varepsilon\ A' \mid a) \in \mathfrak{R}$ and $(T \rightsquigarrow v\ \varepsilon\ A \mid d, T \rightsquigarrow v'\ \varepsilon\ A' \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (\mathsf{unfold}, v)\ \varepsilon\ \rho\alpha.A \mid c, T \rightsquigarrow w\ \varepsilon\ B \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (\mathsf{unfold}, v)\ \varepsilon\ \rho\alpha.A \mid c$ was formed by the instance of (O-ρ^+) for $\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{unfold};\ d \to c) \in \mathcal{T}$, then $w = (\mathsf{unfold}, v')$ for some v' and $\rho\alpha'.A'$ such that $(T \rightsquigarrow v\ \varepsilon\ [\rho\alpha.A/\alpha]A \mid d, T \rightsquigarrow v'\ \varepsilon\ [\rho\alpha'.A'/\alpha']A' \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (\mathsf{unfold}, v)\ \varepsilon\ \rho\alpha.A \mid c, T \rightsquigarrow w\ \varepsilon\ B \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (\mathsf{unfold}, v)\ \varepsilon\ \rho\alpha.A \mid c$ was formed by the instance of (O-ρ^-) for $\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{unfold};\ c \leftarrow d) \in \mathcal{T}$, then $w = (\mathsf{unfold}, v')$ for some v' and $\rho\alpha'.A'$ such that $(T \rightsquigarrow v\ \varepsilon\ [\rho\alpha.A/\alpha]A \mid d, T \rightsquigarrow v'\ \varepsilon\ [\rho\alpha'.A'/\alpha']A' \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (\mathsf{shift}, v)\ \varepsilon \downarrow A \mid c, T \rightsquigarrow w\ \varepsilon\ B \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (\mathsf{shift}, v)\ \varepsilon \downarrow A \mid c$ was formed by the instance of (O-$\downarrow$) for $\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{shift};\ d \leftarrow c) \in \mathcal{T}$, then $w = (\mathsf{shift}, v')$ and $B = \downarrow B'$ for some v' and B' such that $(T \rightsquigarrow v\ \varepsilon\ A' \mid d, T \rightsquigarrow v'\ \varepsilon\ B' \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (\mathsf{shift}, v)\ \varepsilon \downarrow A \mid c, T \rightsquigarrow w\ \varepsilon\ B \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (\mathsf{shift}, v)\ \varepsilon \uparrow A \mid c$ was formed by the instance of (O-$\uparrow$) for $\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{shift};\ c \to d) \in \mathcal{T}$, then $w = (\mathsf{shift}, v')$ and $B = \uparrow B'$ for some v' and B' such that $(T \rightsquigarrow v\ \varepsilon\ A' \mid d, T \rightsquigarrow v'\ \varepsilon\ B' \mid d) \in \mathfrak{R}$;
- if $(T \rightsquigarrow (\mathsf{val}\ f, v)\ \varepsilon\ \tau \wedge A \mid c, T \rightsquigarrow w\ \varepsilon\ B \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (\mathsf{val}\ f, v)\ \varepsilon\ \tau \wedge A \mid c$ was formed by the instance of (O-$\wedge$) for $\mathsf{msg}(c, \_ \leftarrow \mathsf{output}\ c\ f;\ d \to c) \in \mathcal{T}$, then

$w = (\mathsf{val}\ f, v')$ and $B = \tau \wedge B'$ for some v' and B' such that $(T \rightsquigarrow v \ \varepsilon\ A' \mid d, T \rightsquigarrow v' \ \varepsilon\ B' \mid d) \in \mathfrak{R}$;

- if $(T \rightsquigarrow (\mathsf{val}\ f, v) \ \varepsilon\ \tau \supset A \mid c, T \rightsquigarrow w \ \varepsilon\ B \mid c) \in \mathfrak{R}$ and $T \rightsquigarrow (\mathsf{val}\ f, v) \ \varepsilon\ \tau \supset A \mid c$ was formed by the instance of (O-$\supset$) for $\mathsf{msg}(c, \_ \leftarrow \mathsf{output}\ c\ f;\ c \leftarrow d) \in \mathcal{T}$, then $w = (\mathsf{val}\ f, v')$ and $B = \tau \supset B'$ for some v' and B' such that $(T \rightsquigarrow v \ \varepsilon\ A' \mid d, T \rightsquigarrow v' \ \varepsilon\ B' \mid d) \in \mathfrak{R}$.

Proposition 6.2.1. *If T is a trace from a well-typed configuration, then for all c, if $T \rightsquigarrow v \ \varepsilon\ A \mid c$ and $T \rightsquigarrow w \ \varepsilon\ B \mid c$, then $v = w$ and $A = B$. Moreover, if $T \rightsquigarrow v \ \varepsilon\ A \mid c$, then its derivation is unique.*

Proof. Fix some trace T and let $\mathfrak{R}$ be the relation

$$\mathfrak{R} = \{(T \rightsquigarrow v \ \varepsilon\ A \mid c, T \rightsquigarrow w \ \varepsilon\ B \mid c) \mid \exists v, w, c, A, B. T \rightsquigarrow v \ \varepsilon\ A \mid c \wedge T \rightsquigarrow w \ \varepsilon\ B \mid c\}.$$

We show that it is a bisimulation. Let $(T \rightsquigarrow v \ \varepsilon\ A \mid c, T \rightsquigarrow w \ \varepsilon\ B \mid c) \in \mathfrak{R}$ be arbitrary. It follows from corollaries 5.9.3 and 5.9.8 that at most one rule is applicable to form a judgment of the form $T \rightsquigarrow \cdot \ \varepsilon\ \cdot \mid c$ (with c fixed), so $T \rightsquigarrow v \ \varepsilon\ A \mid c$ and $T \rightsquigarrow w \ \varepsilon\ B \mid c$ were both formed by the same rule. We proceed by case analysis on this rule. We only give a few illustrative cases; the rest will follow by analogy.

Case (O-$\bot$): The conclusions are equal, so we are done.

Case (O-$\otimes$) for $\mathsf{msg}(c, \mathsf{send}\ c\ a;\ d \to c)$: Then there exist $r, r', u, u', C, C', D, D'$ such that $v = (r, u)$, $w = (r', u')$, $A = C \otimes D$, $B = C' \otimes D'$, $T \rightsquigarrow r \ \varepsilon\ C \mid a$, $T \rightsquigarrow r' \ \varepsilon\ C' \mid a$, $T \rightsquigarrow u \ \varepsilon\ D \mid d$, and $T \rightsquigarrow u' \ \varepsilon\ D' \mid d$. But $(T \rightsquigarrow r \ \varepsilon\ C \mid a, T \rightsquigarrow r' \ \varepsilon\ C' \mid a) \in \mathfrak{R}$ and $(T \rightsquigarrow u \ \varepsilon\ D \mid d, T \rightsquigarrow u' \ \varepsilon\ D' \mid d) \in \mathfrak{R}$, so we are done.

Case (O-$\wedge$) for $\mathsf{msg}(c, \_ \leftarrow \mathsf{output}\ c\ f;\ d \to c)$: Then there exist v', w', A', B' such that $v = (\mathsf{val}\ f, v')$, $w = (\mathsf{val}\ f, w')$, $A = \tau \wedge A'$, $B = \tau \wedge B'$, $T \rightsquigarrow v' \ \varepsilon\ A' \mid d$, and $T \rightsquigarrow w' \ \varepsilon\ B' \mid d$. But $(T \rightsquigarrow v' \ \varepsilon\ A' \mid d, T \rightsquigarrow w' \ \varepsilon\ B' \mid d) \in \mathfrak{R}$, so we are done.

It follows that $\mathfrak{R}$ is a bisimulation.

Consider arbitrary $T \rightsquigarrow v \ \varepsilon\ A \mid c$ and $T \rightsquigarrow w \ \varepsilon\ B \mid c$. They are related by $\mathfrak{R}$, so they are bisimilar. By [JR12, Theorem 2.7.2], bisimilar elements of the terminal coalgebra are equal. It follows that $v = w$ and $A = B$ as desired.

To see that the derivation of $T \rightsquigarrow v \ \varepsilon\ A \mid c$ is unique, recall from above that at most one rule is applicable to form a judgment of the form $T \rightsquigarrow \cdot \ \varepsilon\ \cdot \mid c$ (with c fixed). Because each rule has only judgments of this form as its hypotheses, it follows that at each step in the derivation, exactly one rule instance can be applied to justify a given hypothesis. So the derivation is unique. □

Next, we set out to show that an observed communication exists for every channel appearing in a trace. This involves explicitly constructing a potentially infinite proof tree. The following definition of a tree in terms of its rooted paths is useful for doing so:

Definition 6.2.2 ([San12, Remark 2.11.1])**.** A **tree** over a set X is a set T of non-empty finite sequences of elements of X such that

(1) there is only one sequence of length one (corresponding to the root of the tree); and
(2) if the sequence $x_0, \ldots, x_{n+1}$ is in T, then so is $x_0, \ldots, x_n$. ◂

We generalize it to allow us to order branches, e.g., to talk about "left" and "right" branches.

Definition 6.2.3. An **ordered tree** over a set X is a tree over $\mathbb{N} \times X$ such that

(1) there is only one sequence of length one;
(2) if the sequence $(n_0, x_0), \ldots, (n_{m-1}, x_{m-1}), (n_m + 1, x_m)$ is in T $(m \geq 0)$, then so is $(n_0, x_0), \ldots, (n_{m-1}, x_{m-1}), (n_m, x)$ for some x. ◂

Definition 6.2.4. Let T be a tree over a set X. The **subtree** rooted at $x_0, \ldots, x_n$ is the tree

$$\{x_n, \ldots, x_{n+m} \mid x_0, \ldots, x_n, \ldots, x_{n+m} \in X\}.$$ ◂

Next, we characterize communications $v \varepsilon A$ as trees. Let L be the set of labels that can be sent in communications. The set of *communication tags* is given by:

$$\mathsf{CommTags} = \{\bot, \mathsf{close}, \mathsf{pair}, \mathsf{unfold}, \mathsf{shift}\} \cup \{\mathsf{val}\ f \mid f\ \mathsf{val} \wedge \exists \tau.(\cdot \Vdash f : \tau)\} \cup L$$

Communications represent ordered trees, e.g., the communications (v, v') and (v', v) are distinct whenever $v \neq v'$.

Lemma 6.2.5. *Assume all labels are drawn from some set L. The following corecursive function ψ is well-defined and it injectively maps set of communications into the set $\mathcal{A}$ of ordered trees over* CommTags*:*

$$\psi(v) = \begin{cases} \{(\mathsf{o}, \bot)\} & v = \bot \\ \{(\mathsf{o}, \mathsf{close})\} & v = \mathsf{close} \\ \bigcup_{\mathsf{o} \leq i \leq \mathsf{1}} \{(\mathsf{o}, \mathsf{pair}), (i, t), \sigma \mid ((\_, t), \sigma) \in \psi(v_i)\} & v = (v_\mathsf{o}, v_\mathsf{1}) \\ \{(\mathsf{o}, \mathsf{unfold}), x \mid x \in \psi(v')\} & v = (\mathsf{unfold}, v') \\ \{(\mathsf{o}, \mathsf{shift}), x \mid x \in \psi(v')\} & v = (\mathsf{shift}, v') \\ \{(\mathsf{o}, \mathsf{val}\ f), x \mid x \in \psi(v')\} & v = (\mathsf{val}\ f, v') \\ \{(\mathsf{o}, l), x \mid x \in \psi(v')\} & v = (l, v') \end{cases}$$

If T is a subtree rooted at some $x_\mathsf{o}, \ldots, x_n$ in $\psi(v)$, then it is the image of some communication w that is a subphrase of v.

Proof. This function is clearly injective. Indeed, the only point of subtlety is ensuring that (v_1, v_2) and (v_2, v_1) do not map to the same tree, but this is ensured by the left branch with l and the right branch with r. □

Proposition 6.2.6. *If T is a trace from $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$, then for all c, if $T \vdash c : A$, then $T \rightsquigarrow v \varepsilon A \mid c$ for some v.*

Proof. We explicitly construct the proof tree as an ordered tree over the set of rules forming $T \rightsquigarrow v \varepsilon A \mid c$.

Assume first that $T \vdash c : A$ but that $c = \mathsf{cc}(\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o}))$ for no $\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o}) \in \mathcal{T}$. In this case, $T \rightsquigarrow \bot \varepsilon A \mid c$ by (O-$\bot$) and we are done.

Otherwise, assume that $T \vdash c : A$ and that $c = \mathsf{cc}(\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o}))$ for some $\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o}) \in \mathcal{T}$. The proof is subtle, for we must show that the potentially infinite communication $v \varepsilon A$ exists, and then construct a potentially infinite proof tree whose conclusion already contains this communication. We proceed as follows:

(1) We describe how the message facts in $\mathcal{T}$ describe a tree M in message facts rooted at $\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o})$.
(2) We convert this tree M into an ordered tree S over the set $\mathsf{CommTags} \times (\mathcal{T} \cup \mathsf{Chans})$, where Chans is the set of free channel names appearing in $\mathcal{T}$. This tree S will act as a sort of "skeleton" or "outline" for the proof that $T \rightsquigarrow v \varepsilon A \mid c$ for some v.
(3) We use lemma 6.2.5 to extract a communication v' from the first component of the elements in the paths in S.
(4) We use S to construct a proof tree that $T \rightsquigarrow v' \varepsilon A \mid c$.

We begin by describing the tree M over $\mathcal{T}$ rooted at $\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o})$. By corollary 5.9.8, $\mathcal{T}$ contains at most one fact $\mathsf{msg}(d_\mathsf{o}, m_\mathsf{o})$ such that $c = \mathsf{cc}(\mathsf{msg}(d_\mathsf{o}, m_\mathsf{o}))$ for all c . The sequence $\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o})$ of length one exists by assumption. The sequences $x_\mathsf{o}, \ldots, x_{n+1}$ in M are given by all sequences $\mathsf{msg}(a_\mathsf{o}, m_\mathsf{o}), \ldots, \mathsf{msg}(a_{n+1}, m_{n+1}.)$ where for $\mathsf{o} \leq i < i + 1 \leq n + 1$, both

(1) $\mathsf{cc}(\mathsf{msg}(a_{i+1}, m_{i+1})) \subseteq \mathsf{fc}(\mathsf{msg}(a_i, m_i))$, and
(2) $\mathsf{msg}(a_{i+1}, m_{i+1}) \neq \mathsf{msg}(a_i, m_i)$.

Observe that a sequence $x_1, \ldots, x_n \in M$ is maximally long in M if and only if one of the following two conditions holds:

(1) $x_n = \mathsf{msg}(a_n, \mathsf{close}\ a_n)$ is the close message, or
(2) $x_n = \mathsf{msg}(a_n, m_n)$ and there is no other message fact in c whose carrier channel appears free in $\mathsf{msg}(a_n, m_n)$.

Next, we translate the tree M into an ordered tree[3] S over $\mathsf{CommTags} \times (\mathcal{T} \cup \mathsf{Chans})$ as follows:

(1) if $x_1, \ldots, x_n \in M$, then $y_1, \ldots, y_n \in S$, where for $1 \leq i \leq n$, y_i is given by:

$$y_i = \begin{cases} ((t, \mathsf{close}), x_i) & x_i = \mathsf{msg}(a_i, \mathsf{close}\ a_i) \\ ((t, \mathsf{pair}), x_i) & x_i = \mathsf{msg}(a_i, \mathsf{send}\ a_i\ b_i;\ d_i \to a_i) \\ ((t, \mathsf{pair}), x_i) & x_i = \mathsf{msg}(a_i, \mathsf{send}\ a_i\ b_i;\ d_i \to a_i) \\ ((t, \mathsf{pair}), x_i) & x_i = \mathsf{msg}(d_i, \mathsf{send}\ a_i\ b_i;\ a_i \leftarrow d_i) \\ ((t, \mathsf{unfold}), x_i) & x_i = \mathsf{msg}(a_i, \mathsf{send}\ a_i\ \mathsf{unfold};\ d_i \to a_i) \\ ((t, \mathsf{unfold}), x_i) & x_i = \mathsf{msg}(d_i, \mathsf{send}\ a_i\ \mathsf{unfold};\ a_i \leftarrow d_i) \\ ((t, \mathsf{shift}), x_i) & x_i = \mathsf{msg}(a_i, \mathsf{send}\ a_i\ \mathsf{shift};\ d_i \leftarrow a_i) \\ ((t, \mathsf{shift}), x_i) & x_i = \mathsf{msg}(d_i, \mathsf{send}\ a_i\ \mathsf{shift};\ a_i \to d_i) \\ ((t, \mathsf{val}\ f), x_i) & x_i = \mathsf{msg}(a_i, \_ \leftarrow \mathsf{output}\ a_i\ f;\ d_i \to a_i) \\ ((t, \mathsf{val}\ f), x_i) & x_i = \mathsf{msg}(d_i, \_ \leftarrow \mathsf{output}\ a_i\ f;\ a_i \leftarrow d_i) \\ ((t, l), x_i) & x_i = \mathsf{msg}(a_i, a_i.l;\ d_i \to a_i) \\ ((t, l), x_i) & x_i = \mathsf{msg}(d_i, a_i.l;\ a_i \leftarrow d_i) \end{cases}$$

where the tag t is given by

$$t = \begin{cases} 1 & i = j + 1 \wedge x_j = \mathsf{msg}(a_j, \mathsf{send}\ a_j\ b_j;\ d_j \to a_j) \wedge \mathsf{cc}(x_i) = \{d_j\} \\ 1 & i = j + 1 \wedge x_j = \mathsf{msg}(d_j, \mathsf{send}\ a_j\ b_j;\ a_j \leftarrow d_j) \wedge \mathsf{cc}(x_i) = \{d_j\} \\ 0 & \mathsf{otherwise} \end{cases}$$

(2) if $x_1, \ldots, x_n \in M$ and $x_n \neq \mathsf{msg}(a_n, \mathsf{close}\ a_n)$ and there exists a $d \in \mathsf{fc}(x_n)$ such that $d \neq \mathsf{cc}(\mathsf{msg}(e, m))$ for all $\mathsf{msg}(e, m) \in \mathcal{T}$, then $y_1, \ldots, y_n, y_{n+1} \in S$ where for $1 \leq i \leq n$, where y_i is given by the recipe above for $1 \leq i \leq n$, and $y_{n+1} = ((t, \bot), d)$ where $t = 0$ or $t = 1$ is determined from x_n as above.

Taking the first projection of each element in each sequence in S gives a tree M' over the set $\mathsf{CommTags}$. It is clearly in the image of the set of communications under the injection from lemma 6.2.5. Taking the preimage of M', we get a communication v.

Next, we show that $T \rightsquigarrow v\ \varepsilon\ A \mid c$. By [San12, Theorem 2.12.5], it is sufficient to give a winning strategy to the verifier for the coinductive game induced by the rules defining $T \rightsquigarrow v\ \varepsilon\ c \mid A$ [San12, § 2.12]. The refuter makes the first move, playing $x_0 = T \rightsquigarrow v\ \varepsilon\ c \mid A$. The verifier must provide a set J_0 such that $T \rightsquigarrow v\ \varepsilon\ c \mid A$ is the conclusion of some rule with hypotheses J_0. The refuter then chooses an $x_1 \in J_0$, to which the verifier must provide a set J_1 such that x_1 is the conclusion of some rule with hypotheses J_1. The game proceeds in this way, producing a sequence $x_0, J_0, \ldots, x_n, J_n, \ldots$. The verifier wins if this sequence is infinite, or if $J_n = \varnothing$ for some n.

At the outset, we observe that:

(1) $T \vdash c : A$ by hypothesis,
(2) v is the preimage of the subtree $S_0 = S$ rooted at the root s_0 of S.

For all $i \geq 0$, the verifier chooses J_i satisfying the following conditions:

(1) x_i is the conclusion of a rule with hypotheses J_i;
(2) for all $T \rightsquigarrow d\ \varepsilon\ w \mid A \in J_i$,
   (a) $T \vdash d : A$,
   (b) w is the preimage of a subtree S_w of S rooted at s_w, and s_i is a prefix of s_w,
   (c) if $w = \bot$, then the root of S_w is $((\_, \bot), c)$,

[3] We reassociated the parentheses of elements in this tree for convenience.

(d) if $w \neq \bot$, then the root of S_w is $(\_, \mathsf{msg}(e, m))$ for some $\mathsf{msg}(e, m)$ with $d = \mathsf{cc}(\mathsf{msg}(e, m))$, and
(e) where $(\_, \mathsf{msg}(e, m))$ is the last element[4] of s_i, $d \in \mathsf{fc}(\mathsf{msg}(e, m))$.

Given a choice $x_{i+1} \in J_i$ by the verifier, let the sequence s_{i+1} be given by the corresponding s_w guaranteed by item 2b.

Given an $x_i = T \rightsquigarrow w \ \varepsilon \ c \mathbin{/} B$ chosen by the refuter, the verifier's choice of J_i is given by case analysis on w. We give the illustrative cases:

Case $\bot$: We proceed by case analysis on i:
Subcase $i = 0$: This case is impossible by the assumption made at the beginning of this proof.
Subcase $i = j + 1$: Then $\bot$ is the preimage of $((\_, \bot), c)$ by lemma 6.2.5 and item 2c. We choose $J_i = \varnothing$. Then $T \rightsquigarrow \bot \ \varepsilon \ c \mathbin{/} B$ by (O-$\bot$). We check that the rule is in fact applicable:
(1) $T \vdash c : B$ because $T \rightsquigarrow w \ \varepsilon \ c \mathbin{/} B \in J_j$, and J_j satisfies assumption item 2a by construction.
(2) $c \neq \mathsf{cc}(\mathsf{msg}(d, m))$ for all $\mathsf{msg}(d, m) \in \mathcal{T}$. Indeed, by assumption item 2c, the root of s_w is $((\_, \bot), c)$. By construction of S, $((\_, \bot), c)$ appears in a sequence if and only if $c \neq \mathsf{cc}(\mathsf{msg}(d, m))$ for all $\mathsf{msg}(d, m) \in \mathcal{T}$.

We conclude that the rule is applicable. Because J_i is empty, the conditions stipulated by item 2 hold vacuously.

Case close: By construction and by lemma 6.2.5, the root of S_w is $((\_, \mathsf{close}), \mathsf{msg}(e, \mathsf{close}\ e))$. By item 2d, $c = \mathsf{cc}(\mathsf{msg}(e, \mathsf{close}\ e)) = e$, so $c = e$. By proposition 5.9.4, $T \vdash c : \mathbf{1}$. By item 2a, $T \vdash d : B$, so by corollary 5.9.3, $B = \mathbf{1}$. Pick $J_i = \varnothing$. Then $T \rightsquigarrow \mathsf{close} \ \varepsilon \ c \mathbin{/} \mathbf{1}$ by (O-$\mathbf{1}$).

Case (u, u'): By construction and by lemma 6.2.5, the root of S_w is $((\_, \mathsf{pair}), \mathsf{msg}(g, m))$. By item 2d, the carrier channel of $\mathsf{msg}(g, m)$ is c. By construction of S, $\mathsf{msg}(g, m)$ must be one of the following:
Subcase $\mathsf{msg}(c, \mathsf{send}\ c\ e;\ d \to c)$: By proposition 5.9.4, $T \vdash c : E \otimes D$, $T \vdash e : E$, and $T \vdash d : D$. Pick $J_i = \{T \rightsquigarrow u \ \varepsilon \ e \mathbin{/} E, T \rightsquigarrow u' \ \varepsilon \ d \mathbin{/} D\}$. Then $T \rightsquigarrow (u, u') \ \varepsilon \ E \otimes D \mathbin{/} c$ by (O-$\otimes$).
Subcase $\mathsf{msg}(d, \mathsf{send}\ c\ e;\ c \leftarrow d)$: By proposition 5.9.4, $T \vdash c : E \multimap D$, $T \vdash e : E$, and $T \vdash d : D$. Pick $J_i = \{T \rightsquigarrow u \ \varepsilon \ e \mathbin{/} E, T \rightsquigarrow u' \ \varepsilon \ d \mathbin{/} D\}$. Then $T \rightsquigarrow (u, u') \ \varepsilon \ E \multimap D \mathbin{/} c$ by (O-$\multimap$).
We show this choice of J_i satisfies the invariant. We show that $T \vdash e : E$ satisfies item 2; the analysis for $T \vdash d : D$ is analogous. By the above case analysis, we have $T \vdash e : E$, satisfying item 2a. By assumption, (u, u') is the preimage of a subtree S_w rooted at s_w. By lemma 6.2.5, u is the preimage of a subtree S_u rooted at $s_u = s_w, ((\mathsf{l}, \_), \_)$ for some $((\mathsf{l}, \_), \_)$. By construction of S, the root $((\mathsf{l}, \_), \_)$ of S_u satisfies items 2c and 2d. Item 2e is also satisfied by construction of S.

Case (unfold, u): By construction and by lemma 6.2.5, the root of S_w is $((\_, \mathsf{unfold}), \mathsf{msg}(g, m))$. By item 2d, the carrier channel of $\mathsf{msg}(g, m)$ is c. By construction of S, $\mathsf{msg}(g, m)$ must be one of the following:
Subcase $\mathsf{msg}(c, \mathsf{send}\ c\ \mathsf{unfold};\ d \to c)$: By proposition 5.9.4, $T \vdash c : \rho\delta.D$ and $T \vdash d : [\rho\delta.D/\delta]D$. Pick $J_i = \{T \rightsquigarrow u \ \varepsilon \ d \mathbin{/} [\rho\delta.D/\delta]D\}$. Then $T \rightsquigarrow u \ \varepsilon \ [\rho\delta.D/\delta]D \mathbin{/} d$ by (O-ρ^+).
Subcase $\mathsf{msg}(d, \mathsf{send}\ c\ \mathsf{unfold};\ c \leftarrow d)$: By proposition 5.9.4, $T \vdash c : \rho\delta.D$ and $T \vdash d : [\rho\delta.D/\delta]D$. Pick $J_i = \{T \rightsquigarrow u \ \varepsilon \ d \mathbin{/} [\rho\delta.D/\delta]D\}$. Then $T \rightsquigarrow u \ \varepsilon \ [\rho\delta.D/\delta]D \mathbin{/} d$ by (O-ρ^-).
We show this choice of J_i satisfies the invariant. Item 2a is satisfied by the above case analysis. By assumption, (unfold, u) is the preimage of a subtree S_w rooted at s_w. By lemma 6.2.5, u is the preimage of a subtree S_u rooted at $s_u = s_w, ((\mathsf{u}, \_), \_)$ for some $((\mathsf{u}, \_), \_)$. By construction of S, the root $((\mathsf{u}, \_), \_)$ of S_u satisfies items 2c and 2d. Item 2e is also satisfied by construction of S.

The above describes a winning strategy for the verifier, for in each case the verifier can always make a move. □

The converse of proposition 6.2.6 also holds:

Corollary 6.2.7. *If T is a trace from $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$, then for all c, if $T \rightsquigarrow v \ \varepsilon \ A \mathbin{/} c$, then $T \vdash c : A$.*

[4] We can assume that it is of this form and not $(\bot, a)$ because we are giving the verifier a winning strategy, and the verifier would lose were it not the case.

Proof. Assume $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$. The judgment $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$ is only defined for channel names that appear free in T. Because $T \vdash c : (\cdot)$ is a total function from these channel to types, there exists a B such that $T \vdash c : B$. By proposition 6.2.6, there exists a w such that $T \rightsquigarrow w \mathrel{\varepsilon} B \mid c$. By proposition 6.2.1, $A = B$, so $T \vdash c : A$. □

So far, we have checked that $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$ defines a function from free channels c to communications v and types A. We now check that these communications are actually session-typed communications:

Proposition 6.2.8. *If $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$, then $v \mathrel{\varepsilon} A$.*

Proof. By coinduction on the rules defining $v \mathrel{\varepsilon} A$. Set $P = \{v \mathrel{\varepsilon} A \mid \exists c. T \rightsquigarrow v \mathrel{\varepsilon} A \mid c\}$. We must show that for all $v \mathrel{\varepsilon} A \in P$, there exists a rule with conclusion $v \mathrel{\varepsilon} A$ and hypotheses $H \subseteq P$. Let $v \mathrel{\varepsilon} A \in P$ be arbitrary. We proceed by case analysis on $v \mathrel{\varepsilon} A$, giving only the illustrative cases:

Case $\bot \mathrel{\varepsilon} A$**:** Note that $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$ is only defined when $\cdot \vdash A \ \mathsf{type}_s$. So $\cdot \vdash A \ \mathsf{type}_s$ by definition of P and because $\bot \mathrel{\varepsilon} A \in P$. So we are done by (C-$\bot$).

Case $\mathsf{close} \mathrel{\varepsilon} \mathbf{1}$**:** Immediate by (C-**1**).

Case $(l, v) \mathrel{\varepsilon} \oplus\{l : A_l\}_{l \in L}$**:** This case arises because $T \rightsquigarrow (l, v) \mathrel{\varepsilon} \oplus\{l : A_l\}_{l \in L} \mid c$ for some c. But $T \rightsquigarrow (l, v) \mathrel{\varepsilon} \oplus\{l : A_l\}_{l \in L} \mid c$ must have been formed by (O-$\oplus$) applied to $T \rightsquigarrow v \mathrel{\varepsilon} A_l \mid d$ for some d. This implies $v \mathrel{\varepsilon} A_l \in P$. So we are done by (C-$\oplus$).

Case $(\mathsf{val}\ f, v) \mathrel{\varepsilon} \tau \wedge A$**:** Then $(\mathsf{val}\ f, v) \mathrel{\varepsilon} \tau \wedge A$ because $T \rightsquigarrow (\mathsf{val}\ f, v) \mathrel{\varepsilon} \tau \wedge A \mid c$ for some c. This last judgment must have been formed by (O-$\wedge$) applied to some $T \rightsquigarrow v \mathrel{\varepsilon} d \mid A$. This implies $v \mathrel{\varepsilon} A \in P$. We are done by (C-$\wedge$) if $\cdot \Vdash f : \tau$. Because (O-$\wedge$) formed $T \rightsquigarrow (\mathsf{val}\ f, v) \mathrel{\varepsilon} \tau \wedge A \mid c$, we know that $T \vdash c : \tau \wedge A$ and that $\mathsf{msg}(c, \_ \leftarrow \mathsf{output}\ c\ f;\ d \to c) \in \mathcal{T}$. By proposition 5.9.4 and corollary 5.9.3 we deduce $\cdot \Vdash f : \tau$ as desired. □

Combining propositions 6.2.1, 6.2.6 and 6.2.8, we get:

Corollary 6.2.9. *The judgment $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$ defines a total function from channel names c appearing free in T to session-typed communications $v \mathrel{\varepsilon} A$.*

The following theorem is an immediate consequence of corollary 6.2.9:

Theorem 6.2.10. *Let T be a fair execution of $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$. For all $c : A \in \Gamma, \mathrm{I}, \Delta$, there exist unique v such that $v \mathrel{\varepsilon} A$ and $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$.*

Proof. By assumption, $T \vdash c : A$ for all $c : A \in \Gamma, \mathrm{I}, \Delta$. Then by proposition 6.2.6, there exists a v such that $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$, and $v \mathrel{\varepsilon} A$ by proposition 6.2.8. By proposition 6.2.1, each such v is unique determined. □

The following theorem captures the confluence property typically enjoyed by SILL-style languages:

Theorem 6.2.11. *Let T and T' be a fair executions of $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$. For all $c : A \in \Gamma, \mathrm{I}, \Delta$, if $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$ and $T' \rightsquigarrow w \mathrel{\varepsilon} A \mid c$, then $v = w$.*

Proof. Assume $T \rightsquigarrow v \mathrel{\varepsilon} A \mid c$ and $T' \rightsquigarrow w \mathrel{\varepsilon} A \mid c$. By corollary 5.9.11, traces T and T' are union-equivalent, i.e., $\mathcal{T} = \mathcal{T}'$. It easily follows that $T' \rightsquigarrow w \mathrel{\varepsilon} A \mid c$ if and only if $T \rightsquigarrow w \mathrel{\varepsilon} A \mid c$. So $v = w$ by theorem 6.2.10. □

Theorem 6.2.11 crucially depends on fairness. Indeed, without fairness a process can have infinitely many observations. To see this, let Ω be the divergent process given by example 5.3.3 and let B be given by

$$\cdot\ ;\ a : \mathbf{1} \vdash \mathsf{fix}\ p.\mathsf{send}\ b\ \mathsf{unfold};\ b.l;\ p :: b : \rho\beta.\oplus\{l : \beta\}$$

Rule (66) is the first step of any execution of their composition $\cdot\ ;\ \cdot \vdash a \leftarrow \Omega;\ B :: b : \rho\beta.\oplus\{l : \beta\}$. It spawns Ω and B as separate processes. Without fairness, an execution could then consist exclusively of applications of rule (70) to Ω. This would give the observed communication $\bot$ on b. Alternatively,

B could take finitely many steps, leading to observations where b is a tree of correspondingly finite height. Fairness ensures that B and Ω both take infinitely many steps, leading to the unique observation $(\mathsf{unfold}, (l, (\mathsf{unfold}, \dots)))$ on b.

This notion of observed communication scales to support language extensions. Indeed, for each new session type one first defines its corresponding session-typed communications. Then, one specifies how to observe message judgments $\mathsf{msg}(c, m)$ in a trace as communications.

So far, our approaches for observing communications on channels have had a strictly coinductive flavour. We now show how we can construct them as the least upper bounds of sequences of approximations. Recall from definition 6.1.13 that $\langle\!\langle A \rangle\!\rangle_=$ is the set of communications of type A ordered by $\trianglelefteq/=$. Given some trace T, we let T^n be its prefix of n steps. Then, where v_n is given by $T^n \rightsquigarrow v_n \;\varepsilon\; A \mid c$ for each n, and v is given by $T \rightsquigarrow v \;\varepsilon\; A \mid c$, we show that v is the least upper bound of the ascending chain of v_n in $\langle\!\langle A \rangle\!\rangle_=$.

We begin by showing that the v_n form an ascending chain, i.e., that observing communications is monotone in the length of a trace.

Proposition 6.2.12. *For all n and all channels c, if $T^n \rightsquigarrow v_n \;\varepsilon\; A \mid c$ and $T^{n+1} \rightsquigarrow v_{n+1} \;\varepsilon\; A \mid c$, then $v_n \trianglelefteq/= v_{n+1} \;\varepsilon\; A$.*

Proof. All communications observed from a finite prefix are finite. We proceed by strong induction on the size of the derivation of $T^n \rightsquigarrow v_n \;\varepsilon\; A \mid c$. The base cases for $v_n \;\varepsilon\; A$ are given by the axioms:

Case (O-$\bot$): Then $v_n = \bot$. We are done by (CS-$\bot$).

Case (O-**1**): Then $v_n = \mathsf{close}$. Extending the trace by a single step does not affect the observed communication on c, so $v_{n+1} = \mathsf{close}$ as well. We are done by (CS-**1**).

Now we proceed to the inductive step. We show several illustrative cases; the remainder are analogous.

Case (O-$\otimes$): Then $T^n \rightsquigarrow (u_n, w_n) \;\varepsilon\; A \otimes B \mid c$ because $\mathsf{msg}(c, \mathsf{send}\ c\ a;\ d \to c)$ appears in T^n, and $T^n \rightsquigarrow u_n \;\varepsilon\; A \mid a$ and $T^n \rightsquigarrow w_n \;\varepsilon\; B \mid d$. Let u_{n+1} and w_{n+1} be given by $T^{n+1} \rightsquigarrow u_{n+1} \;\varepsilon\; A \mid a$ and $T^{n+1} \rightsquigarrow w_{n+1} \;\varepsilon\; B \mid d$. By the induction hypothesis, $u_n \trianglelefteq/= u_{n+1} \;\varepsilon\; A$ and $w_n \trianglelefteq/= w_{n+1} \;\varepsilon\; B$. By (O-$\otimes$), $T^{n+1} \rightsquigarrow (u_{n+1}, w_{n+1}) \;\varepsilon\; A \otimes B \mid c$. We are done by (CS-$\otimes$).

Case (O-$\wedge$): Then $T^n \rightsquigarrow (\mathsf{val}\ f, w_n) \;\varepsilon\; \tau \wedge A \mid c$ because $\mathsf{msg}(c, \_ \leftarrow \mathsf{output}\ c\ f;\ d \to c)$ appears in T^n, and $T^n \rightsquigarrow w_n \;\varepsilon\; A \mid d$. Let w_{n+1} be given by $T^{n+1} \rightsquigarrow w_{n+1} \;\varepsilon\; A \mid d$. By the induction hypothesis, $w_n \trianglelefteq/= w_{n+1} \;\varepsilon\; A$. By (O-$\wedge$), $T^{n+1} \rightsquigarrow (\mathsf{val}\ f, w_{n+1}) \;\varepsilon\; \tau \wedge A \mid c$. We are done by (CS-$\wedge$). □

Proposition 6.2.13. *Let T be a trace, and let v be given by $T \rightsquigarrow v \;\varepsilon\; A \mid c$. For each n, let T^n be the prefix of length n of T, and let v_n be given by $T^n \rightsquigarrow v_n \;\varepsilon\; A \mid c$. Then $v = \bigsqcup^{\uparrow}_n v_n$ in $\langle\!\langle A \rangle\!\rangle_=$.*

Proof. It is sufficient to show that:

(1) for all l, there exists an m such that $\lfloor v \rfloor_l \trianglelefteq/= v_m$; and
(2) for all m, there exists an u such that $v_m \trianglelefteq/= \lfloor v \rfloor_u$.

Indeed, item 1 implies that every upper bound of the v_m is an upper bound of the $\lfloor v \rfloor_m$, while item 2 implies that every upper bound of the $\lfloor v \rfloor_m$ is an upper bound of the v_m. So $\bigsqcup^{\uparrow}_m \lfloor v \rfloor_m \doteq/= \bigsqcup^{\uparrow}_m v_m \;\varepsilon\; A$. By proposition 6.1.7, this is an equality. By corollary 6.1.14, $\bigsqcup^{\uparrow}_m \lfloor v \rfloor_m = v$. So $v = \bigsqcup^{\uparrow}_n v_n$ as desired.

We begin with item 1. Consider some l, and let m be any m such that all of the messages appearing in the derivation of $\lfloor v \rfloor_l$ appear in T^m. For item 2, consider some m, and let u be height of v_m plus one. □

We now state a few basic properties of observations on single channels.

Definition 6.2.14. Let T be a trace of some configuration $\mathcal{C}$. A message fact $\mathsf{msg}(a, m)$ is **observable from c in T** if appears in the derivation of $T \rightsquigarrow v \;\varepsilon\; A \mid c$. ◂

The following proposition captures the intuitive fact that observed communications on c are entirely determined by the set of message facts observable from c. Recall from section 3.1.3

that a refreshing substitution for a trace $T = (M_0, (r_i; (\theta_i, \xi_i))_i)$ is a collection of fresh-constant substitutions $\eta = (\eta_i)_i$ such that $[\eta]T = (M_0, (r_i; (\theta_i, \eta_i))_i)$ is also a trace.

Proposition 6.2.15. *Let T and T' be fair traces of $\mathcal{C}$ and $\mathcal{C}'$, respectively. For all c, let v and v' be given by $T \rightsquigarrow v \ \varepsilon \ A \mid c$ and $T' \rightsquigarrow v' \ \varepsilon \ A \mid c$. If T can be refreshed to T'' such that T'' and T' have the same sets of observable message facts from c, then $v = v'$.*

Proof. Immediate from the fact that $T \rightsquigarrow v \ \varepsilon \ A \mid c$ is entirely determined by the set of messages observable from c, and that it is invariant under refreshing of channel names. □

Configurations with no common channels do not interfere with each other:

Proposition 6.2.16. *Consider multisets $\Gamma \vdash \mathcal{C} :: \Delta$ and $\Phi \vdash \mathcal{D} :: \Xi$ with disjoint sets of free channels, i.e., such that $\Gamma\Phi \vdash \mathcal{C}, \mathcal{D} :: \Delta\Xi$ is well formed. Then for all fair traces T of $\mathcal{C}, \mathcal{D}$ and T' of $\mathcal{C}$ and all $c \in \mathsf{fc}(\mathcal{C})$, $T \rightsquigarrow v \ \varepsilon \ A \mid c$ if and only if $T' \rightsquigarrow v \ \varepsilon \ A \mid c$.*

Proof. We claim that every fair trace T of $\mathcal{C}, \mathcal{D}$ induces a fair trace T' of $\mathcal{C}$. Explicitly, we use preservation (proposition 5.9.1) to show that

(1) every multiset in T is of the form $\Gamma\Phi \vdash \mathcal{C}', \mathcal{D}' :: \Delta\Xi$ for some $\Gamma \vdash \mathcal{C}' :: \Delta$ and $\Phi \vdash \mathcal{D}' :: \Xi$;
(2) every step in T is of the form $\mathcal{C}'[\mathcal{D}']^{\Phi}_{\Xi} \to \mathcal{C}'[\mathcal{D}'']^{\Phi}_{\Xi}$ or $\mathcal{D}'[\mathcal{C}']^{\Gamma}_{\Delta} \to \mathcal{D}'[\mathcal{C}'']^{\Gamma}_{\Delta}$.

It is sufficient to show that these properties hold for every finite prefix of T. We do so by induction on the number n of steps in the prefix. The result is immediate when $n = 0$. Assume the result for some n, then the last multiset is of the form $\Gamma\Phi \mid \mathrm{I} \vdash \mathcal{C}', \mathcal{D}' :: \Delta\Xi$ for some $\Gamma \vdash \mathcal{C}' :: \Delta$ and $\Phi \vdash \mathcal{D}' :: \Xi$. Assume some rule instance $r(\theta)$ is applicable to $\mathcal{C}', \mathcal{D}'$. If its active multiset intersects with both $\mathcal{C}'$ and $\mathcal{D}'$, then a case analysis on the rules reveals that it contains a message fact. By lemma 5.9.5, this implies that $\mathcal{C}'$ and $\mathcal{D}'$ have a free channel in common, a contradiction. So the active multiset of $r(\theta)$ is contained in $\mathcal{C}'$ or in $\mathcal{D}'$. We are done by preservation.

Taking the subsequence of steps of the form $\mathcal{D}'[\mathcal{C}']^{\Gamma}_{\Delta} \to \mathcal{D}'[\mathcal{C}'']^{\Gamma}_{\Delta}$ gives a trace T' of $\mathcal{C}$. It is fair because T is fair. Let $c \in \mathsf{fc}(\mathcal{C})$ be arbitrary. Every message fact observable from c in T is observable from c in T', and vice-versa. It follows that $T \rightsquigarrow v \ \varepsilon \ A \mid c$ if and only if $T' \rightsquigarrow v \ \varepsilon \ A \mid c$. The choice of trace T' for $\mathcal{C}$ does not matter by theorem 6.2.11. □

6.3. Observed Communications of Configurations

We use theorems 6.2.10 and 6.2.11 to define observations on channels in a configuration, independently of the trace:

Definition 6.3.1. Given $\Psi \subseteq \Gamma, \mathrm{I}, \Delta$, the **observed communication on** Ψ of $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$ is the tuple

$$\langle \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta \rangle_{\Psi} = (c : v_c)_{c \in \check{\Psi}}$$

of observed communications, where $T \rightsquigarrow v_c \ \varepsilon \ A \mid c$ for $c : A \in \Psi$ for some fair execution T of $\Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta$. If $c \in \check{\Psi}$, then we occasionally write $\langle \Gamma \mid \mathrm{I} \vdash \mathcal{C} :: \Delta \rangle_{\Psi}(c)$ for the communication v_c observed on c. ◀

Definition 6.3.1 is well-defined. Indeed, a fair execution T exists by corollary 5.9.11, and the observed communications do not depend on the choice of T by theorem 6.2.11. The v such that $T \rightsquigarrow v \ \varepsilon \ A \mid c$ exist and are unique by theorem 6.2.10.

Definition 6.3.1 is simplified by the fact that the multiset rewriting rules defining Polarized SILL are non-overlapping, so its fair traces are union-equivalent. Indeed, it was this fact that was used in the proof of theorem 6.2.11 to show that the communication on a channel was independent of the fair trace. In language extensions that do not satisfy this property, observed communications will be sets of tuples, instead of single tuples, but we conjecture that this should pose no significant difficulty to the theory.

Generally, we deem internal channels to be private and unobservable, and we only interact with configurations over their interfaces. However, definition 6.3.1 allows us to observe communications

on a configuration's internal channels. This lets us observe communications between configurations and experiments when defining "internal" observational equivalence in chapter 7.

Observed communications do not take into account the order in which a process sends on channels. For example, the following configurations have the same observed communications $(a : (l, \bot), b : (r, \bot))$ on a and b, even though they send on a and on b in different orders:

$$a : \&\{l : \mathbf{1}\} \vdash \mathsf{proc}(b, a.l;\ b.r;\ a \to b) :: b : \oplus\{r : \mathbf{1}\}$$
$$a : \&\{l : \mathbf{1}\} \vdash \mathsf{proc}(b, b.r;\ a.l;\ a \to b) :: b : \oplus\{r : \mathbf{1}\}.$$

The order in which channels are used is not reflected in observations for several reasons. First, messages are only ordered on a per-channel basis, and messages sent on different channels can arrive out of order. Second, each channel has a unique pair of endpoints, and the (Conf-C) rule organizes processes in a forest-like structure (cf. proposition 5.6.20). This means that two configurations communicating with a configuration $\mathcal{C}$ at the same time cannot directly communicate with each other to compare the order in which $\mathcal{C}$ sent them messages. In other words, the ordering of messages on different channels cannot be distinguished by configurations.

We lift the notion of communication simulation and equivalence to tuples of communications component-wise:

$$(c : v_c)_{c:C\in\Gamma} \trianglelefteq/\leqslant (c : w_c)_{c:C\in\Gamma} \iff \forall c : C \in \Gamma\ .\ v_c \trianglelefteq/\leqslant w_c\ \varepsilon\ C,$$
$$(c : v_c)_{c:C\in\Gamma} \doteq/\equiv (c : w_c)_{c:C\in\Gamma} \iff \forall c : C \in \Gamma\ .\ v_c \doteq/\equiv w_c\ \varepsilon\ C.$$

CHAPTER 7

Observational Preorders and Equivalences

We adopt an extensional view of process equivalence, where we say that two processes are equivalent if we cannot differentiate them through experimentation. Recall that communication is our sole means of interacting with processes, and that we can only observe communications. This suggests that processes should be deemed equivalent if, whenever we subject them to "communicating experiments", we observe equivalent communications. Because our substructural operational semantics and observed communication semantics are defined on configurations and not on processes, it is more natural to define observational equivalence on configurations instead of on processes. We will later show how to restrict our observational equivalences to processes.

We follow Milner [Mil80, chap. 2] and Hoare [Hoa85, p. 65] in identifying experimenting agents with processes themselves (strictly speaking, with configuration contexts). We also build on the "testing equivalences" framework introduced by De Nicola and Hennessy [DH84; Hen83; De 85]. Roughly speaking, this framework subjected processes to experiments that could potentially succeed, and it deemed two processes to be equivalent if they succeeded the same experiments. Their notion of experimental success was based on observing a "success" state. Instead of determining the success of experiments from process states, we determine it by observing communications. The following definition adapts Hennessy's state-based "computational systems" ([Hen83, Definition 2.1.1]) to our communication-based setting:

Definition 7.0.1. An **observation system** $\mathcal{S}$ on configurations is a pair $(\mathcal{X}, \leqslant)$, where

- $\mathcal{X} = (\mathcal{X}_{\Gamma \vdash \Delta})_{\Gamma,\Delta}$ is a type-indexed family of sets, where $\mathcal{X}_{\Gamma\vdash\Delta}$ is a set of pairs $(\mathcal{E}, C)$, where $\Lambda \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi$ is a context and $C \subseteq \Lambda, \mathrm{I}, \Xi$ is a subset of the channels free in $\mathcal{E}[\cdot]^{\Gamma}_{\Delta}$;
- $\leqslant$ is a type-indexed relation on terms.

We assume that $\mathcal{X}$ is closed under exchange: if Γ' and Δ' are permutations of Γ and Δ, respectively, then $\mathcal{X}_{\Gamma'\vdash\Delta'} = \mathcal{X}_{\Gamma\vdash\Delta}$. We also assume that $\mathcal{X}$ is closed under renaming. ◂

Intuitively, $\mathcal{X}_{\Gamma\vdash\Delta}$ is a set of **experiments** $\mathcal{E}$ and observation channels C on configurations with interface (Γ, Δ). In our development, we assume that the experiments $\mathcal{E}$ of observation systems are configuration contexts written in the same language as the configurations on which we are experimenting. However, this is not a necessary assumption: experiments could be written in any language, so long as its multiset rewriting semantics does not interfere with the hypotheses underlying the observed communication semantics of chapter 6.

Definition 7.0.2. Let $\mathcal{S} = (\mathcal{X}, \leqslant)$ be an observation system and $\leqslant$ a preorder. **Observational $\mathcal{S}$-simulation** is the type-indexed relation $\lesssim_{\mathcal{S}}$ on configurations such that $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}} \mathcal{D} :: \Delta$ if and only if for all $(\Lambda \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi,\ \Psi) \in \mathcal{X}_{\Gamma\vdash\Delta}$,

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} :: \Xi \rangle_{\Psi} \leq/\leqslant \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^{\Gamma}_{\Delta} :: \Xi \rangle_{\Psi}.$$

In this case, we say that $\mathcal{C}$ and $\mathcal{D}$ are observationally $\mathcal{S}$-similar. We call $(\lesssim_{\mathcal{S}})^c$ **observational $\mathcal{S}$-precongruence**. ◂

Observational $\mathcal{S}$-simulation is a preorder by proposition 6.1.4. We define observational $\mathcal{S}$-equivalence analogously:

Definition 7.0.3. Let $\mathcal{S} = (\mathcal{X}, \leqslant)$ be an observation system and $\leqslant$ a preorder. **Observational $\mathcal{S}$-equivalence** is the type-indexed relation $\doteq_{\mathcal{S}}$ on configurations such that $\Gamma \vdash \mathcal{C} \doteq_{\mathcal{S}} \mathcal{D} :: \Delta$ if and only

if for all $(\Lambda \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi, C) \in \mathcal{X}_{\Gamma\vdash\Delta}$,

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} :: \Xi\rangle_C \doteq/\leqslant \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^{\Gamma}_{\Delta} :: \Xi\rangle_C.$$

In this case, we say that $\mathcal{C}$ and $\mathcal{D}$ are observationally $\mathcal{S}$-equivalent. We call $(\doteq_{\mathcal{S}})^c$ **observational $\mathcal{S}$-congruence.** ◂

Observational $\mathcal{S}$-equivalence is an equivalence relation by proposition 6.1.6. Apart from instances in which we want to emphasize the symmetry of a result, we will not consider observational $\mathcal{S}$-equivalence. This does not restrict the applicability of our results in light of proposition 7.0.4, and the general fact that $\leqslant \cap \leqslant^{\mathrm{op}}$ is an equivalence relation whenever $\leqslant$ is a preorder.

Proposition 7.0.4. *Let $\mathcal{S} = (\mathcal{X}, \leqslant)$ be an observation system. $\mathcal{S}$-simulation and $\mathcal{S}$-equivalence and their (pre)congruences are related as follows:*

(1) $\Gamma \vdash \mathcal{C} \doteq_{\mathcal{S}} \mathcal{D} :: \Xi$ *if and only if* $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}} \mathcal{D} :: \Xi$ *and* $\Gamma \vdash \mathcal{D} \lesssim_{\mathcal{S}} \mathcal{C} :: \Xi$*;*
(2) $\Gamma \vdash \mathcal{C}\ (\doteq_{\mathcal{S}})^c\ \mathcal{D} :: \Xi$ *if and only if* $\Gamma \vdash \mathcal{C}\ (\lesssim_{\mathcal{S}})^c\ \mathcal{D} :: \Xi$ *and* $\Gamma \vdash \mathcal{D}\ (\lesssim_{\mathcal{S}})^c\ \mathcal{C} :: \Xi$*.*

Though $\mathcal{S}$-simulation and $\mathcal{S}$-equivalence are defined using contexts, it is important to note that, unlike contextual-equivalence-style relations, they are stable under language extension. Indeed, extending the process language with new language constructs does not affect $\mathcal{X}$'s ability to discriminate between pre-existing programs. However, $\mathcal{S}$-precongruence and $\mathcal{S}$-congruence need not be stable under language extension. This is because subjecting previously precongruent configurations to new language constructs could have effects that are discernible by $\mathcal{X}$.

There are three natural[1] families of observation systems, and each reflects a different outlook on program testing. The first is an "external" notion of experimentation, where we imagine experiments as black boxes into which we place configurations, and where the result of the experiment is reported on its exterior channels. Because values in Polarized SILL are unobservable (they are all of function or quoted process type), we do not differentiate between values observed on experiments' exterior channels. This gives:

Definition 7.0.5. The **external observation system** E is given by $(\mathcal{X}^E, \mathfrak{U})$, where $\mathfrak{U}$ is the universal relation and $\mathcal{X}^E$ is the family

$$\mathcal{X}^E_{\Gamma\vdash\Delta} = \{(\mathcal{E}[\cdot]^{\Gamma}_{\Delta}, \Lambda\Xi) \mid \Lambda \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi\}.$$ ◂

Alternatively, we could adopt an "internal" view of experimentation, where experiments question their subjects, and we observe their answers. This approach is reminiscent of the process equivalence Darondeau [Dar82] gave to a calculus inspired by CCS. This internal view is well-suited to synchronous settings like Classical Processes [Wad14], where processes cannot communicate unless we give them communication partners. Accordingly, it is the approach Atkey [Atk17] took when defining process equivalence for CP.

Definition 7.0.6. The **internal observation system** I is given by $(\mathcal{X}^I, \mathfrak{U})$, where $\mathfrak{U}$ is the universal relation and $\mathcal{X}^I$ is the family

$$\mathcal{X}^I_{\Gamma\vdash\Delta} = \{(\mathcal{E}[\cdot]^{\Gamma}_{\Delta}, \Gamma\Delta) \mid \Lambda \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi\}.$$ ◂

The final approach takes a "total" view on experimentation, where we observe communications on all channels in the experimental context. We use it strictly as a technical tool for relating observation systems. We will see that total observational simulation implies both internal and external observational simulations.

Definition 7.0.7. The **(strict) total observation system** T is given by $(\mathcal{X}^T, =)$, where $\mathcal{X}^T$ is the family

$$\mathcal{X}^T_{\Gamma\vdash\Delta} = \{(\mathcal{E}[\cdot]^{\Gamma}_{\Delta}, \Lambda\,\mathrm{I}\,\Xi) \mid \Lambda \shortmid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi\}.$$ ◂

[1] We use "natural" in the same sense as in "natural transformation", where the choice of observed channels C is uniform across all contexts.

We respectively call $\lesssim_E$, $\lesssim_I$, and $\lesssim_T$ **external**, **internal**, and **total observational simulation.**

Our observation system framework lets us prove general properties that hold for simulations induced by the above observation systems. For example, we can use the fact that experiments are written in the same language as configurations to give an easy check that observational simulations are precogruences. We say that an experiment set $\mathcal{X}$ is **closed under composition with contexts** if for all contexts $\Lambda \vdash \mathcal{C}[\cdot]^\Gamma_\Delta :: \Xi$ and all experiments $(\mathcal{E}[\cdot]^\Lambda_\Xi, C) \in \mathcal{X}$, there exists a $C' \supseteq C$ such that $(\mathcal{E}[\mathcal{C}[\cdot]^\Gamma_\Delta]^\Lambda_\Xi, C') \in \mathcal{X}$.

Proposition 7.0.8. *Let $\mathcal{S} = (\mathcal{X}, \leqslant)$ be an observation system. If $\mathcal{X}$ is closed under composition with contexts, then $\lesssim_{\mathcal{S}}$ is a precongruence, i.e., $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}} \mathcal{D} :: \Delta$ if and only if $\Gamma \vdash \mathcal{C} \ (\lesssim_{\mathcal{S}})^c \ \mathcal{D} :: \Delta$.*

Proof. Assume that $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}} \mathcal{D} :: \Delta$. Let $\Lambda \shortmid \mathcal{F}[\cdot]^\Gamma_\Delta \vdash \Xi ::$ be an arbitrary context and let $(\Phi \vdash \mathcal{E}[\cdot]^\Lambda_\Xi :: \Psi, C) \in \mathcal{X}$ be an arbitrary experiment. We must show that

$$\langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{C}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \rangle_C \lesssim/\leqslant \langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{D}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \rangle_C.$$

By closure under composition with contexts, $(\Phi \vdash \mathcal{E}[\mathcal{F}[\cdot]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi, C') \in \mathcal{X}$ for some $C' \supseteq C$. This implies

$$\langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{C}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \rangle_{C'} \lesssim/\leqslant \langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{D}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \rangle_{C'}.$$

The result is immediate from the fact that $C' \supseteq C$. □

Corollary 7.0.9. *Total observational simulation $\lesssim_T$ and external observational simulation $\lesssim_E$ are precongruences.*

Recall simply branched contexts from definition 5.7.9. It is sometimes sufficient to consider only simply branched experiments, i.e., experiments where the context is simply branched. This result is reminiscent of Milner's "context lemma" [Mil77]. We begin with the following lemma:

Lemma 7.0.10. *Let $(\Lambda \shortmid \mathrm{I} \vdash \mathcal{E}[\cdot]^\Gamma_\Delta :: \Xi, C)$ be an arbitrary experiment. There exists a simply branched context $\Lambda \shortmid \mathrm{I}, \Xi \vdash \mathcal{B}[\cdot]^\Gamma_\Delta :: a : \mathbf{1}$ such that for all $\Gamma \vdash \mathcal{C} :: \Delta$,*

$$\langle \Lambda \shortmid \mathrm{I}, \Xi \vdash \mathcal{B}[\mathcal{C}]^\Gamma_\Delta :: a : \mathbf{1} \rangle_C = \langle \Lambda \shortmid \mathrm{I} \vdash \mathcal{E}[\mathcal{C}]^\Gamma_\Delta :: \Xi \rangle_C,$$

$$\langle \Lambda \shortmid \mathrm{I}, \Xi \vdash \mathcal{B}[\mathcal{C}]^\Gamma_\Delta :: a : \mathbf{1} \rangle_a = (a : \bot).$$

Proof. Recall from example 5.3.3 that there exists a divergent process $\cdot \ ; \ \Xi \vdash \Omega :: c : \mathbf{1}$. Let $\Lambda \shortmid \mathrm{I}, \Xi \vdash \mathcal{B}[\cdot]^\Gamma_\Delta :: a : \mathbf{1}$ be given by the composition $\mathcal{E}[\cdot]^\Gamma_\Delta, \mathsf{proc}(c, \Omega)$. Let $\Gamma \vdash \mathcal{C} :: \Delta$ be arbitrary. Let T be an arbitrary fair trace of $\Lambda \shortmid \mathrm{I} \vdash \mathcal{E}[\mathcal{C}]^\Gamma_\Delta :: \Xi$. Let T' be the trace of $\Lambda \shortmid \mathrm{I}, \Xi \vdash \mathcal{B}[\mathcal{C}]^\Gamma_\Delta :: a : \mathbf{1}$ given by interleaving each step of T with an application of rule (73) to $\mathsf{proc}(c, \Omega)$. It is fair. The two traces are union equivalent, so they have the same observed communications on all channels. In particular, no message is sent on a, so $T' \rightsquigarrow a \ \varepsilon \ \mathbf{1} \ / \ \bot$. □

Proposition 7.0.11. *Let $\mathcal{S} = (\mathcal{X}, \leqslant)$ range over observation systems I and T. Let $\mathcal{S}_B = (\mathcal{X}^B, \leqslant)$ be its restriction to simply branched contexts, where $\mathcal{X}^B = \{(\mathcal{E}, C) \in \mathcal{X} \mid \mathcal{E} \text{ is simply branched}\}$. Then $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}} \mathcal{D} :: \Delta$ if and only if $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}_B} \mathcal{D} :: \Delta$.*

Proof. Sufficiency is immediate for both properties: every simply branched experiment is an experiment. To see necessity, assume that $\Gamma \vdash \mathcal{C} \lesssim_{\mathcal{S}_B} \mathcal{D} :: \Delta$. Let $(\Lambda \vdash \mathcal{E}[\cdot]^\Gamma_\Delta :: \Xi, C) \in \mathcal{X}$ be an arbitrary experiment. We must show that

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^\Gamma_\Delta :: \Xi \rangle_C \lesssim/\leqslant \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^\Gamma_\Delta :: \Xi \rangle_C.$$

By lemma 7.0.10, there exists a simply branched experiment $(\Lambda \vdash \mathcal{B}[\cdot]^\Gamma_\Delta :: a : \mathbf{1}, D) \in \mathcal{X}$ with $C \subseteq D$. In either case,

$$\langle \Lambda \vdash \mathcal{B}[\mathcal{C}]^\Gamma_\Delta :: a : \mathbf{1} \rangle_D \lesssim/\leqslant \langle \Lambda \vdash \mathcal{B}[\mathcal{D}]^\Gamma_\Delta :: a : \mathbf{1} \rangle_D$$

by assumption. Because $C \subseteq D$,

$$\langle \Lambda \vdash \mathcal{B}[\mathcal{C}]^\Gamma_\Delta :: a : \mathbf{1} \rangle_C \lesssim/\leqslant \langle \Lambda \vdash \mathcal{B}[\mathcal{D}]^\Gamma_\Delta :: a : \mathbf{1} \rangle_C.$$

By the lemma,

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^\Gamma_\Delta :: \Xi \rangle_C \lesssim/\leqslant \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^\Gamma_\Delta :: \Xi \rangle_C.$$

□

Recall simply branched contextual interiors from definition 5.7.3. Our first "context lemma" (cf. [Mil77, pp. 6–7]) states that, in the cases of external and total simulations, it is sufficient to quantify over simply branched contexts to show that configurations are $\mathcal{S}$-precongruent:

Proposition 7.0.12. *Let $\mathcal{S} = (\mathcal{X}, \leqslant)$ range over observation systems E and T. Then $\Gamma \vdash \mathcal{C} \ (\leqslant_{\mathcal{S}})^c \ \mathcal{D} :: a : A$ if and only if $\Gamma \vdash \mathcal{C} \ (\leqslant_{\mathcal{S}})^b \ \mathcal{D} :: a : A$.*

Proof. Sufficiency is immediate: every simply branched context is a context. To see necessity, assume that $\Gamma \vdash \mathcal{C} \ (\leqslant_{\mathcal{S}})^b \ \mathcal{D} :: a : A$, i.e., that

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{B}[\mathcal{C}]^{\Gamma}_{a:A}]^{\Phi}_{b:B} :: \Xi \rangle_C \leqslant\!/\!\leqslant \langle \Lambda \vdash \mathcal{E}[\mathcal{B}[\mathcal{D}]^{\Gamma}_{a:A}]^{\Phi}_{b:B} :: \Xi \rangle_C \tag{90}$$

for all simply branched contexts $\Phi \vdash \mathcal{B}[\cdot]^{\Gamma}_{a:A} :: b : B$ and experiments $(\Lambda \vdash \mathcal{E}[\cdot]^{\Phi}_{b:B} :: \Xi, C) \in \mathcal{X}$. Let $\Phi \mid \mathrm{I}_F \vdash \mathcal{F}[\cdot]^{\Gamma}_{a:A} :: \Psi$ be an arbitrary context, and let $(\Lambda \mid \mathrm{I}_E \vdash \mathcal{E}[\cdot]^{\Phi}_{\Psi} :: \Xi, C) \in \mathcal{X}$ be an arbitrary experiment. We must show that

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{F}[\mathcal{C}]^{\Gamma}_{a:A}]^{\Phi}_{\Psi} :: \Xi \rangle_C \leqslant\!/\!\leqslant \langle \Lambda \vdash \mathcal{E}[\mathcal{F}[\mathcal{D}]^{\Gamma}_{a:A}]^{\Phi}_{\Psi} :: \Xi \rangle_C. \tag{91}$$

The context $\mathcal{E}[\mathcal{F}[\cdot]^{\Gamma}_{a:A}]^{\Phi}_{\Psi}$ appears as the experiments $(\mathcal{E}[\mathcal{F}[\cdot]^{\Gamma}_{a:A}]^{\Phi}_{\Psi}, \Lambda\Xi) \in \mathcal{X}^E$ and $(\mathcal{E}[\mathcal{F}[\cdot]^{\Gamma}_{a:A}]^{\Phi}_{\Psi}, \Lambda \mathrm{I}_F \mathrm{I}_E \Xi) \in \mathcal{X}^T$. Instantiating (90) with $\mathcal{B} = [\cdot]^{\Gamma}_{a:A}$ gives the result. □

In this dissertation, we focus on external observational simulation. It is better behaved than internal observational simulation (it is a precongruence). It is also, in some sense, easier to work with. This is because we can use the fact that we never observe input on external channels. As a result, the observed communications are simpler: they are never bidirectional.

In section 7.1, we show that total observational equivalence is closed under execution. We show in section 7.2 that internal observational precongruence implies external observational precongruence. In section 7.3, we develop external observational simulation. We relate it to weak barbed precongruence. Figure 7.1 of section 7.4 summarizes the relationships between these different relations on configurations. We show in section 7.5 how to relate relations on configurations and relations on processes. This will give certain precongruences for processes.

7.1. Total Observations for Configurations

Total observational equivalence is useful for showing properties that hold of all observational $\mathcal{S}$-simulations and $\mathcal{S}$-equivalences. This is because total observational equivalence is the finest notion of equivalence based on observed communications:

Proposition 7.1.1. *Let $\mathcal{S}_i = (\mathcal{X}_i, \leqslant_i)$ for $i = 1, 2$ be observation systems. Then $\Gamma \vdash \mathcal{C} \leqslant_{\mathcal{S}_1} \mathcal{D} :: \Delta$ implies $\Gamma \vdash \mathcal{C} \leqslant_{\mathcal{S}_2} \mathcal{D} :: \Delta$ whenever both:*

1. *for all $(\Lambda \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi, C) \in \mathcal{X}_2$, there exists $C' \supseteq C$ such that $(\Lambda \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi, C') \in \mathcal{X}_1$, and*
2. *$\leqslant_1 \subseteq \leqslant_2$.*

Proof. Assume that $\Gamma \vdash \mathcal{C} \leqslant_{\mathcal{S}_1} \mathcal{D} :: \Delta$. Let $(\Lambda \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi, C) \in \mathcal{X}_2$ be arbitrary. We must show that

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} :: \Xi \rangle_C \leqslant\!/\!\leqslant_2 \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^{\Gamma}_{\Delta} :: \Xi \rangle_C.$$

By assumption, $(\Lambda \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Xi, C') \in \mathcal{X}_1$ for some $C' \supseteq C$. This implies that

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} :: \Xi \rangle_{C'} \leqslant\!/\!\leqslant_1 \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^{\Gamma}_{\Delta} :: \Xi \rangle_{C'}.$$

Because $C \subseteq C'$,

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} :: \Xi \rangle_C \leqslant\!/\!\leqslant_1 \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^{\Gamma}_{\Delta} :: \Xi \rangle_C.$$

By monotonicity (proposition 6.1.4),

$$\langle \Lambda \vdash \mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} :: \Xi \rangle_C \leqslant\!/\!\leqslant_2 \langle \Lambda \vdash \mathcal{E}[\mathcal{D}]^{\Gamma}_{\Delta} :: \Xi \rangle_C.$$ □

Corollary 7.1.2. *Let $T = (\mathcal{X}^T, =)$ be given by definition 7.0.7. For all observation systems $\mathcal{S} = (\mathcal{X}, \leqslant)$,*

- *if* $\Gamma \vdash \mathcal{C} \leq_T \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \leq_{\mathcal{S}} \mathcal{D} :: \Delta$;
- *if* $\Gamma \vdash \mathcal{C} \leq_{(\mathcal{X}^T, \leqslant)} \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \leq_{\mathcal{S}} \mathcal{D} :: \Delta$.

Total observational equivalence is closed under multiset rewriting:

Proposition 7.1.3. *If* $\Gamma \mid \mathcal{C} \vdash \Delta ::$ *and* $\mathcal{C} \to \mathcal{C}'$, *then* $\Gamma \vdash \mathcal{C} \doteq_T \mathcal{C}' :: \Delta$.

Proof. Consider an arbitrary experiment $(\Phi \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^{\Gamma}_{\Delta} :: \Lambda, \Phi\,\mathrm{I}\,\Lambda)$, and let T be a fair trace of $\mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta}$. By fairness and preservation, we can assume without loss of generality that the first step of T is $\mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta} \to \mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta}$. If every message fact in $\mathcal{C}$ appears in $\mathcal{C}'$, then we are done. Indeed, the tail T' of T is a fair trace of $\mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta}$ by the fair tail property (proposition 3.3.1). Both T and T' have the same sets of message facts, so they induce the same observations for each channel in Φ, I, Δ.

Now assume that $\mathcal{C} \to \mathcal{C}'$ consumes a message fact, i.e., that there is some message fact $\mathsf{msg}(c, m) \in \mathcal{C}$ that is not in $\mathcal{C}'$. We must show that it is not observable from any channel d in Φ, I, Λ in T, i.e., that it does not appear in any derivation of $T \rightsquigarrow u \;\varepsilon\; A \;/\; d$ for d in Φ, I, Λ.

A case analysis on the rules defining $T \rightsquigarrow u \;\varepsilon\; A \;/\; d$ shows that if $T \rightsquigarrow v \;\varepsilon\; B \;/\; a$ appears as a premise of a rule, then

(1) the rule is due to a $\mathsf{msg}(b, m)$ with $a \in \mathsf{fc}(\mathsf{msg}(b, m))$, and
(2) the conclusion of the rule is of the form $T \rightsquigarrow w \;\varepsilon\; C \;/\; c$, where $c = \mathsf{cc}(\mathsf{msg}(b, m))$.

Suppose to the contrary that $\mathsf{msg}(c, m)$ is observable from some d in Φ, I, Λ in T. We proceed by induction on the height h of $\mathsf{msg}(c, m)$ in the derivation of observed communication. Set $a = \mathsf{cc}(\mathsf{msg}(c, m))$.

Case $h = 1$: Then $\mathsf{msg}(c, m)$ is observable because $\mathsf{cc}(\mathsf{msg}(c, m)) = d$ is in Φ, I, Λ. We have $\mathsf{msg}(c, m) \notin \mathcal{C}'$ only if $\mathsf{msg}(c, m)$ was in the active portion of the rule used to make the step. However, by lemma 5.9.5, this implies that $d \in \mathsf{ic_s}(\mathsf{proc}(b, P))$ for some $\mathsf{proc}(b, P) \in \mathcal{C}$. This implies that d is an internal channel of $\mathcal{C}$ by lemma 5.6.11, which in turn implies that d is not in Φ, I, Γ, Δ, Λ. This is a contradiction.

Case $h = h' + 1$: Assume the result for h'. Then $\mathsf{msg}(c, m)$ appears at height h in the derivation, and there is a $\mathsf{msg}(b, m')$ at height h' in the derivation such that $a \in \mathsf{fc}(\mathsf{msg}(b, m'))$. Because $\mathsf{msg}(c, m) \in \mathcal{C}$ but $\mathsf{msg}(c, m) \notin \mathcal{C}'$, we know that $\mathsf{msg}(c, m)$ was in the active multiset of the rule used to make the step $\mathcal{C} \to \mathcal{C}'$. By lemma 5.9.5, a was an internal channel and it does not appear free on the right side of the rule. By preservation, it follows that a is not free in $\mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta}$. It follows that a cannot appear free in $\mathsf{msg}(b, m')$, a contradiction.

To see that a cannot appear free in $\mathsf{msg}(b, m')$, we consider two cases:
Subcase $\mathsf{msg}(b, m') \in \mathcal{C}$: A case analysis on the rules shows that we also have $\mathsf{msg}(b, m') \in \mathcal{C}'$, a contradiction of $a \notin \mathsf{fc}(\mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta})$.
Subcase $\mathsf{msg}(b, m') \notin \mathcal{C}$: Then $\mathsf{msg}(b, m')$ must appear in some $\mathcal{C}''$ such that $\mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta} \to^* \mathcal{C}''$. But each free channel in $\mathsf{msg}(b, m') \in \mathcal{C}''$ is either already in $\mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta}$, or it is freshly generated, so not in $\mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta}$ or $\mathcal{E}[\mathcal{C}']^{\Gamma}_{\Delta}$. Both of these possibilities contradict the assumption that $a \in \mathsf{fc}(\mathsf{msg}(b, m'))$. □

The following proposition shows that forwarding has no observable effect on communications, and that it acts only to rename channels:

Proposition 7.1.4. *For all* $\Gamma \vdash \mathcal{C} :: \Delta, c : C$ *and* $\Gamma, c : A \vdash \mathcal{A} :: \Delta$, *respectively,*

(1) if C is positive, then $\Gamma \vdash [d/c]\mathcal{C} \doteq_T \mathcal{C}, \mathsf{proc}(d, c \to d) :: \Delta, d : C$;
(2) if A is positive, then $\Gamma, d : A \vdash [d/c]\mathcal{A} \doteq_T \mathsf{proc}(c, d \to c), \mathcal{A} :: \Delta$;
(3) if A is negative, then $\Gamma, d : A \vdash [d/c]\mathcal{A} \doteq_T \mathsf{proc}(c, d \leftarrow c), \mathcal{A} :: \Delta$;
(4) if C is negative, then $\Gamma \vdash [d/c]\mathcal{C} \doteq_T \mathcal{C}, \mathsf{proc}(d, c \leftarrow d) :: \Delta, d : C$.

Proof. Assume first that $\Gamma \vdash \mathcal{C} :: \Delta, c : C$ and that C is positive. Let $(\mathcal{E}[\cdot]^{\Gamma}_{\Delta, c:C}, D)$ be an arbitrary experiment. Then $\mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta, c:C} \to^* \mathcal{E}'[\mathsf{msg}(c, m)]^{\Gamma'}_{\Delta', c:C}$ for some $\mathcal{E}'$ if and only if

$$([d/c]\mathcal{E})\,[[d/c]\mathcal{C}]^{\Gamma}_{\Delta, d:C} \to^* ([d/c]\mathcal{E}')[\mathsf{msg}(d, [d/c]m)]^{\Gamma'}_{\Delta', d:C},$$

and an induction shows that this holds if and only if

$$([d/c]\mathcal{E})\,[\mathcal{C}, \mathsf{proc}(d, c \to d)]^{\Gamma}_{\Delta,d:C} \to^* ([d/c]\mathcal{E}')\,[\mathsf{msg}(c, m), \mathsf{proc}(d, c \to d)]^{\Gamma'}_{\Delta',d:C}.$$

But this last multiset in turn steps to $([d/c]\mathcal{E}')[\mathsf{msg}(d, [d/c]m)]^{\Gamma'}_{\Delta',d:C}$ by rule (64):

$$([d/c]\mathcal{E}')\,[\mathsf{msg}(c, m), \mathsf{proc}(d, c \to d)]^{\Gamma'}_{\Delta',d:C} \to ([d/c]\mathcal{E}')[\mathsf{msg}(d, [d/c]m)]^{\Gamma'}_{\Delta',d:C}.$$

So if this collection of logical equivalences hold, we are done by proposition 7.1.3 and the fact that $\doteq_T$ is reflexive.

If $\mathsf{msg}(c, m)$ appears in no fair trace of $\mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta,c:C}$, then an induction shows that every trace of $\mathcal{E}[\mathcal{C}]^{\Gamma}_{\Delta,c:C}$ is a trace of $([d/c]\mathcal{E})\,[\mathcal{C}, \mathsf{proc}(d, c \to d)]^{\Gamma}_{\Delta,d:C}$ (modulo the presence of the forwarding process), and that they have the same sets of message facts. So they induce the same observations on all channels and we are done.

The remaining cases are analogous. □

7.2. Internal Observations for Configurations

Atkey [Atk17, p. 79] states without proof that his internal-style observational equivalence is a congruence. Internal observational equivalence is not a congruence in our setting because of value transmission, and our choice to compare functional values using the universal relation $\mathfrak{U}$. As described above, we use $\mathfrak{U}$ because of philosophical objections to inspecting values of function type. Unfortunately, it is these values of function type that cause observational equivalence to not be a congruence. We conjecture that replacing $\mathfrak{U}$ by a suitable refinement would cause internal observational equivalence to be a congruence.

Proposition 7.2.1. *Internal observational simulation (equivalence) is not a precongruence (congruence).*

Proof. We construct an explicit counter-example. Let the processes P and Q respectively be:

$$\cdot\,;\cdot \vdash \_ \leftarrow \mathsf{output}\ c\ (\lambda x : \tau.x);\ \mathsf{close}\ c :: c : (\tau \to \tau) \wedge \mathbf{1},$$
$$\cdot\,;\cdot \vdash \_ \leftarrow \mathsf{output}\ c\ (\lambda x : \tau.\mathsf{fix}\ y.y);\ \mathsf{close}\ c :: c : (\tau \to \tau) \wedge \mathbf{1}.$$

Then for all experiments $(\Lambda \vdash \mathcal{E}[\cdot]_{c:(\tau\to\tau)\wedge\mathbf{1}} :: \Xi, c) \in \mathcal{X}^I$,

$$\begin{aligned}
&\langle \Lambda \vdash \mathcal{E}[\mathsf{proc}(c, P)]_{c:(\tau\to\tau)\wedge\mathbf{1}} :: \Xi\rangle_c \\
&= (c : (\mathsf{val}\ \lambda x : \tau.x, \mathsf{close})) \\
&\doteq/\mathfrak{U}\ (c : (\mathsf{val}\ \lambda x : \tau.\mathsf{fix}\ y.y, \mathsf{close})) \\
&= \langle \Lambda \vdash \mathcal{E}[\mathsf{proc}(c, Q)]_{c:(\tau\to\tau)\wedge\mathbf{1}} :: \Xi\rangle_c.
\end{aligned}$$

So $\cdot \vdash \mathsf{proc}(c, P) \doteq_I \mathsf{proc}(c, Q) :: c : (\tau \to \tau) \wedge \mathbf{1}$. By proposition 7.0.4, $\cdot \vdash \mathsf{proc}(c, P) \leq_I \mathsf{proc}(c, Q) :: c : (\tau \to \tau) \wedge \mathbf{1}$. Take τ to be $\rho \to \rho$ for some ρ, and consider the process R given by

$$\cdot\,; c : (\tau \to \tau) \wedge \mathbf{1} \vdash x \leftarrow \mathsf{input}\ c;\ \_ \leftarrow \mathsf{output}\ b\ (x(\lambda z : \rho.z));\ c \to b :: b : \tau \wedge \mathbf{1}.$$

Set $\mathcal{C}[\cdot]_{c:(\tau\to\tau)\wedge\mathbf{1}} = [\cdot]_{c:(\tau\to\tau)\wedge\mathbf{1}}, \mathsf{proc}(b, R)$. Then the experiment $([\cdot]_{b:\tau\wedge\mathbf{1}}, b) \in \mathcal{X}^I$ can differentiate $\mathcal{C}[\mathsf{proc}(c, P)]^{\cdot}_{c:(\tau\to\tau)\wedge\mathbf{1}}$ and $\mathcal{C}[\mathsf{proc}(c, Q)]^{\cdot}_{c:(\tau\to\tau)\wedge\mathbf{1}}$. Intuitively, this is because $(\lambda x : \tau.x)(\lambda z : \rho.z)$ will converge in the first case, but $(\lambda x : \tau.\mathsf{fix}\ y.y)(\lambda z : \rho.z)$ will diverge in the second. Explicitly,

$$\begin{aligned}
&\langle \cdot \vdash \mathcal{C}[\mathsf{proc}(c, P)]^{\cdot}_{c:(\tau\to\tau)\wedge\mathbf{1}} :: b : \tau \wedge \mathbf{1}\rangle_b \\
&= (b : (\mathsf{val}\ (\lambda z : \rho.z), \mathsf{close})) \\
&\not\doteq/\mathfrak{U}\ (b : \bot) \\
&= \langle \cdot \vdash \mathcal{C}[\mathsf{proc}(c, Q)]^{\cdot}_{c:(\tau\to\tau)\wedge\mathbf{1}} :: b : \tau \wedge \mathbf{1}\rangle_b.
\end{aligned}$$

So $\leq_I$ is not a precongruence. It follows from proposition 7.0.4 that $\doteq_I$ is not a congruence. □

Proposition 7.2.2. *Let $\mathcal{X}^I$ and $\mathcal{X}^T$ be given by definition 7.0.6 and definition 7.0.7, respectively. Then for all $\leqslant$, if $\Gamma \vdash \mathcal{C}\ \left(\leqslant_{(\mathcal{X}^I,\leqslant)}\right)^c\ \mathcal{D} :: \Delta$, then $\Gamma \vdash \mathcal{C}\ \left(\leqslant_{(\mathcal{X}^T,\leqslant)}\right)^c\ \mathcal{D} :: \Delta$.*

Proof. Let $\Lambda \vdash \mathcal{F}[\cdot]^\Gamma_\Delta :: \Xi$ be an arbitrary context and $(\Phi \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^\Lambda_\Xi :: \Psi, \Phi\,\mathrm{I}\,\Psi) \in \mathcal{X}^T$ an arbitrary experiment. We must show that

$$\left\langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{C}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \right\rangle_{\Phi\,\mathrm{I}\,\Psi} \leqslant/\leqslant \left\langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{D}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \right\rangle_{\Phi\,\mathrm{I}\,\Psi}.$$

This is the case if and only if for all $c : C \in \Phi, \mathrm{I}, \Xi$,

$$\left\langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{C}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \right\rangle_{\Phi\,\mathrm{I}\,\Psi}(c) \leqslant/\leqslant \left\langle \Phi \vdash \mathcal{E}[\mathcal{F}[\mathcal{D}]^\Gamma_\Delta]^\Lambda_\Xi :: \Psi \right\rangle_{\Phi\,\mathrm{I}\,\Psi}(c). \tag{92}$$

Fix some arbitrary such $c : C$. Induction on $\Phi \mid \mathrm{I} \vdash \mathcal{E}[\cdot]^\Lambda_\Xi :: \Psi$ gives a decomposition of $\mathcal{E}$ as a composition of contexts $\mathcal{E}[\cdot]^\Lambda_\Xi = \mathcal{E}'[\mathcal{E}''[\cdot]^\Lambda_\Xi]^{\Lambda'}_{\Xi'}$ such that $c : C \in \Lambda', \Xi'$. Observe that the composition $\mathcal{E}''[\mathcal{F}[\cdot]^\Gamma_\Delta]^\Lambda_\Xi$ is again a context, and that $(\mathcal{E}'[\cdot]^{\Lambda'}_{\Xi'}, \Lambda'\Xi') \in \mathcal{X}^I$. Then by assumption,

$$\left\langle \Phi \vdash \mathcal{E}'[\left(\mathcal{E}''[\mathcal{F}[\mathcal{C}]^\Gamma_\Delta]^\Lambda_\Xi\right)]^{\Lambda'}_{\Xi'} :: \Psi \right\rangle_{\Lambda'\Xi'} \leqslant/\leqslant \left\langle \Phi \vdash \mathcal{E}'[\left(\mathcal{E}''[\mathcal{F}[\mathcal{D}]^\Gamma_\Delta]^\Lambda_\Xi\right)]^{\Lambda'}_{\Xi'} :: \Psi \right\rangle_{\Lambda'\Xi'}.$$

Because $c : C \in \Lambda', \Xi'$, this implies (92) and we are done. □

Combining proposition 7.2.2 and corollary 7.1.2, we conclude:

Corollary 7.2.3. *If $\mathcal{C} \vdash \mathcal{D}\ (\leqslant_I)^c\ \Delta ::$, then $\Gamma \vdash \mathcal{C}\ (\leqslant_E)^c\ \mathcal{D} :: \Delta$.*

7.3. External Observations for Configurations

We show that external observational precongruence coincides with weak barbed precongruence. We first show some general properties about observations on external channels.

Proposition 7.3.1. *Assume that $\Gamma \vdash \mathcal{C} :: \Delta$. We observe no communication on its input channels, i.e., $\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle(c) = \bot$ for all $c \in \mathsf{ic}(\mathcal{C})$.*

Proof. Let $c \in \mathsf{ic}(\mathcal{C})$ be arbitrary, let T be a fair trace, and let $\mathcal{T}$ be the union of all facts appearing in T. Suppose to the contrary that some $\mathsf{msg}(d, m) \in \mathcal{T}$ has c as its carrier channel. Then by remark 5.6.5, c is an output channel of $\mathsf{msg}(d, m)$. By preservation, the subformula property (proposition 5.6.8) and definition 5.6.1, c must also be an output channel of $\Gamma \vdash \mathcal{C} :: \Delta$. But the sets of input and output channels are disjoint, so this is a contradiction. It follows that $\mathcal{T}$ has c as its carrier, so $v_c = \bot$ by (O-⊥). □

Definition 7.3.2. Assume $\Gamma \vdash \mathcal{C} :: \Delta$, and consider a trace T of $\mathcal{C}$. A message fact $\mathsf{msg}(a, m)$ is **externally observable in** T if it is observable from some $c \in \check{\Gamma}, \check{\Delta}$ in T. ◂

Definition 7.3.3. Two configurations $\Gamma \vdash \mathcal{C} :: \Delta$ and $\Gamma \vdash \mathcal{C}' :: \Delta$ have the same externally observable message facts if for some fair traces T and T' of $\mathcal{C}$ and $\mathcal{C}'$, respectively, for all channels $c \in \check{\Gamma}, \check{\Delta}$, the sets of messages observable from c in T and in T' are equal. ◂

Proposition 7.3.4. *If $\Gamma \vdash \mathcal{C} :: \Delta$ and $\Gamma \vdash \mathcal{D} :: \Delta$ have the same sets of externally observable message facts, then $\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle = \langle \Gamma \vdash \mathcal{D} :: \Delta \rangle$.*

Proof. This is an immediate corollary of proposition 6.2.15. □

7.3.1. Barbed Simulation and Precongruence. Barbed bisimulations and congruences [MS92; San92] are the canonical notion of equivalence for process calculi. A barb is an observation predicate $\downarrow$ defined on terms in a calculus that specifies the most basic behavioural observable: the ability to perform an observable action. When defining the barb predicate,

> the global observer [...] can also recognize the production of an observable action, but in this case he cannot see neither the identity of the action produced nor the state reached. [MS92, p. 691]

Concretely, we follow Sangiorgi [San92, § 3.2] and define barbs on a per-channel basis: the predicate $(\cdot)\downarrow_a$ specifies the ability to perform an observable action on channel a.

This minimalist approach to defining barbs contrasts with some recent approaches [YHB07; Ton15; KMP19] whose barbs distinguish between different kinds of actions. For example, Yoshida, Honda, and Berger [YHB07] found it necessary to observe which label was sent to ensure that barbed bisimulation was a congruence. By using a different barb for each kind of typed communication, Toninho [Ton15, § 6.2] was able to give a binary logical relation that was consistent by construction with barbed equivalence.

We prefer the minimalist approach for its conceptual simplicity and generality: it is calculus agnostic. Instead of modifying the concept of a barb to ensure that barbed bisimulation is a congruence, we follow the original approach and extract "barbed congruences" from barbed bisimulations using contextual interiors (cf. [MS92, Definition 8; San92, Definition 3.2.6]).

Definition 7.3.5. A **barb** is the channel-indexed predicate $(\cdot)\downarrow_a$ on processes and configurations inductively defined by:

- $\mathsf{close}\ a \downarrow_a$;
- $a.k;\ P \downarrow_a$;
- $\mathsf{send}\ a\ \mathsf{shift};\ P \downarrow_a$;
- $\_ \leftarrow \mathsf{output}\ a\ M;\ P \downarrow_a$ if $M \Downarrow v$ for some v;
- $\mathsf{send}\ a\ b;\ P \downarrow_a$;
- $\mathsf{send}\ a\ \mathsf{unfold};\ P \downarrow_a$;
- $\mathsf{proc}(c, P)\downarrow_a$ whenever $P \downarrow_a$;
- $\mathsf{msg}(a, m)\downarrow_a$ whenever $m \downarrow_a$;
- $(\mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m^{-}_{b,c}))\downarrow_a$ and $(\mathsf{msg}(a, m^{+}), \mathsf{proc}(b, a \to b))\downarrow_b$; and
- $(\mathcal{C}[\mathcal{D}]^{\Gamma}_{\Delta})\downarrow_a$ whenever $\mathcal{D}\downarrow_a$.

A **weak barb** is the channel-indexed predicate $(\cdot)\Downarrow_a$ on configurations defined by the composition of relations $\to^*\ (\cdot)\downarrow_a$. We write $(\cdot)\not\Downarrow_a$ for the negation of $(\cdot)\Downarrow_a$. ◂

Write $\to^r$ for the reflexive closure of $\to$.

Proposition 7.3.6. *For all configurations $\mathcal{C}$, $\mathcal{C}\downarrow_a$ if and only if $\mathcal{C} \to^r \mathcal{C}'$, $\mathsf{msg}(c, m)$ for some $\mathsf{msg}(c, m)$ with $\mathsf{cc}(\mathsf{msg}(c, m)) = a$.*

Proof. Sufficiency follows by a case analysis on why $\mathcal{C}\downarrow_a$. To see necessity, assume first that $\mathsf{msg}(c, m) \in \mathcal{C}$. Then a case analysis on m gives the result. If $\mathsf{msg}(c, m) \notin \mathcal{C}$, then a case analysis on the (non-reflexive) step gives the result. □

Corollary 7.3.7. *For all $\Gamma \vdash \mathcal{C} :: \Delta$ and $a \in \mathsf{fc}(\mathcal{C})$, $\mathcal{C}\Downarrow_a$ if and only if $\langle\Gamma \vdash \mathcal{C} :: \Delta\rangle_a(a) \neq \bot$.*

The barbed simulation game requires the simulating configuration to match the simulated configuration's barbs:

Definition 7.3.8. A typed relation $\mathfrak{R}$ on configurations is a **(weak) barbed simulation** if $\Delta \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{D} :: \Psi$ implies

(1) if $\mathcal{C} \to \mathcal{C}'$, then $\mathcal{D} \to^* \mathcal{D}'$ with $\Delta \vdash \mathcal{C}'\ \mathfrak{R}\ \mathcal{D}' :: \Psi$; and
(2) for all channels $a : A \in \Delta, \Psi$, if $\mathcal{C}\Downarrow_a$, then $\mathcal{D}\Downarrow_a$.

(Weak) barbed similarity, $\lesssim$, is the largest barbed simulation. Two configurations $\Delta \vdash \mathcal{C} :: \Psi$ and $\Delta \vdash \mathcal{D} :: \Psi$ are **(weak) barbed similar**, $\Delta \vdash \mathcal{C} \lesssim \mathcal{D} :: \Psi$, if $\Delta \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{D} :: \Psi$ for some barbed simulation $\mathfrak{R}$. ◂

We can define barbed bisimulation from barbed simulation in the usual manner:

Definition 7.3.9. A typed relation $\mathfrak{R}$ on configurations is a **(weak) barbed bisimulation** if both $\mathfrak{R}$ and $\mathfrak{R}^{-1}$ are barbed simulations. **(Weak) barbed bisimilarity**, $\approx$, is the largest barbed bisimulation. Two configurations $\Delta \vdash \mathcal{C} :: \Psi$ and $\Delta \vdash \mathcal{D} :: \Psi$ are **(weak) barbed bisimilar**, $\Delta \vdash \mathcal{C} \approx \mathcal{D} :: \Psi$, if $\Delta \vdash \mathcal{C}\ \mathfrak{R}\ \mathcal{D} :: \Psi$ for some barbed bisimulation $\mathfrak{R}$. ◂

Barbed bisimulation is an equivalence relation. We do not develop its theory any further.

Example 7.3.10. The following two configurations are barbed bisimilar:

$$\cdot \vdash \mathsf{proc}(b, a \leftarrow \mathsf{close}\ a;\ (\mathsf{wait}\ a;\ \mathsf{close}\ b)) :: b : \mathbf{1} \tag{93}$$

$$\cdot \vdash \mathsf{proc}(b, \mathsf{close}\ b) :: b : \mathbf{1} \tag{94}$$

The unique execution of (93) is:

$$\mathsf{proc}(b, a \leftarrow \mathsf{close}\ a;\ (\mathsf{wait}\ a;\ \mathsf{close}\ b)) \tag{95}$$

$$\to \mathsf{proc}(a, \mathsf{close}\ a), \mathsf{proc}(b, \mathsf{wait}\ a;\ \mathsf{close}\ b) \tag{96}$$

$$\to \mathsf{msg}(a, \mathsf{close}\ a), \mathsf{proc}(b, \mathsf{wait}\ a;\ \mathsf{close}\ b) \tag{97}$$

$$\to \mathsf{proc}(b, \mathsf{close}\ b) \tag{98}$$

$$\to \mathsf{msg}(b, \mathsf{close}\ b), \tag{99}$$

while the unique execution of (94) is:

$$\mathsf{proc}(b, \mathsf{close}\ b) \tag{100}$$

$$\to \mathsf{msg}(b, \mathsf{close}\ b). \tag{101}$$

Where the numbers refer to the configurations in the above executions, the following relation is a barbed bisimulation:

$$\mathfrak{R} = \{((95), (100)), ((96), (100)), ((97), (100)), ((98), (100)), ((99), (101))\}$$

Indeed, it ensures that the two configurations remain related throughout the stepping game. It also satisfies the requirement that related configurations have the same barbs for channels in their interfaces: in each pair, both configurations satisfy the weak barb $(\cdot) \Downarrow_b$. ◂

Lemma 7.3.11. *If $\Gamma \vdash \mathcal{C} :: \Delta$ and $\mathcal{C} \to \mathcal{C}'$, then for all $c \in \check{\Gamma}, \check{\Delta}$, we have $\mathcal{C} \Downarrow_c$ if and only if $\mathcal{C}' \Downarrow_c$.*

Proof. This is a consequence of proposition 7.1.3 and corollary 7.3.7. □

We can characterize barbed similarity using proposition 7.3.12.

Proposition 7.3.12. *Two configurations are barbed similar, $\Delta \vdash \mathcal{C} \lessapprox \mathcal{D} :: \Psi$, if and only if for all $c \in \check{\Delta}, \check{\Psi}$, if $\mathcal{C} \Downarrow_c$, then $\mathcal{D} \Downarrow_c$.*

Proof. Sufficiency is obvious. To see necessity, let $\mathfrak{R}$ be the relation given by $\{(\mathcal{C}', \mathcal{D}') \mid \mathcal{C} \to^* \mathcal{C}' \wedge \mathcal{D} \to^* \mathcal{D}'\}$. It is a barbed bisimulation. Indeed, it is closed under stepping by construction. Moreover, if $\mathcal{C}'\ \mathfrak{R}\ \mathcal{D}'$ and $\mathcal{C}' \Downarrow_c$ for some $c \in \check{\Delta}, \check{\Psi}$, then $\mathcal{D}' \Downarrow_c$. To see that this is so, observe that if $\mathcal{C}' \Downarrow_c$, then $\mathcal{C} \Downarrow_c$, so $\mathcal{D} \Downarrow_c$ by assumption. Then $\mathcal{D}' \Downarrow_c$ by lemma 7.3.11. □

Proposition 7.3.13. *Barbed similarity not a precongruence relation on configurations.*

Proof. Recall the process Ω from example 5.3.3. Consider the processes P, Q, and R respectively given by:

$$\cdot\ ;\ a : \oplus\{l : \mathbf{1}, r : \mathbf{1}\} \vdash \mathsf{case}\ a\ \{l \Rightarrow \mathsf{wait}\ a;\ \mathsf{close}\ c \mid r \Rightarrow \Omega\} :: c : \mathbf{1},$$

$$\cdot\ ;\ a : \oplus\{l : \mathbf{1}, r : \mathbf{1}\} \vdash \mathsf{case}\ a\ \{l \Rightarrow \Omega \mid r \Rightarrow \mathsf{wait}\ a;\ \mathsf{close}\ c\} :: c : \mathbf{1},$$

$$\cdot\ ;\ \cdot \vdash a.l;\ \mathsf{close}\ a :: a : \oplus\{l : \mathbf{1}, r : \mathbf{1}\}.$$

We have the following pairs of barbed bisimilar configurations:

$$a : \oplus\{l : \mathbf{1}, r : \mathbf{1}\} \vdash \mathsf{proc}(c, P) \approx \mathsf{proc}(c, Q) :: c : \mathbf{1}$$

$$\cdot \vdash \mathsf{proc}(a, R) \approx \mathsf{proc}(a, R) :: (a : \oplus\{l : \mathbf{1}, r : \mathbf{1}\})$$

However, barbed similarity is not contextual (so not a congruence relation), for

$$\cdot \vdash \mathsf{proc}(a, R), \mathsf{proc}(c, P) \not\lessapprox \mathsf{proc}(a, R), \mathsf{proc}(c, Q) :: c : \mathbf{1}.$$

Indeed, the left side steps to the configuration $\mathsf{proc}(c, \mathsf{close}\ c)$ which satisfies the barb $(\cdot) \Downarrow_c$, while right side cannot step to a configuration satisfying this barb. □

Definition 7.3.14. (Weak) barbed precongruence $\lesssim^c$ is the contextual interior of barbed similarity, i.e., $\Gamma \vdash \mathcal{C} \lesssim^c \mathcal{D} :: \Xi$ if and only if if $\Delta \vdash \mathcal{E}[\mathcal{C}]^\Gamma_\Xi \lessapprox \mathcal{E}[\mathcal{C}]^\Gamma_\Xi :: \Lambda$ for all contexts $\Delta \vdash \mathcal{E}[\cdot]^\Gamma_\Xi :: \Lambda$. ◂

7.3.2. Relating Barbed Simulations and External-style Observational Simulations. We relate barbed simulation $\lessapprox$ and external observational simulation $\leq_E$. In fact, we show a more general result. Let $E = (\mathcal{X}^E, \mathfrak{U})$ be given by definition 7.0.5, and let $\leqslant$ be a preorder such that $(\mathcal{X}^E, \leqslant)$ is an observation system. We relate $\lessapprox$ and $\leq_{(\mathcal{X}^E,\leqslant)}$. The result for $\leq_E$ follows as a special case.

Proposition 7.3.15. *If* $\Gamma \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \lessapprox \mathcal{D} :: \Delta$.

Proof. Assume that $\Gamma \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{D} :: \Delta$. Let $c \in \check{\Gamma}, \check{\Delta}$ be arbitrary. By corollary 7.3.7, $\mathcal{C} \Downarrow_c$ if and only if $\langle \Gamma \vdash \mathcal{C} :: \Xi \rangle(c) \neq \bot$. In this case, $\langle \Gamma \vdash \mathcal{D} :: \Delta \rangle(c) \neq \bot$, which holds if and only if $\mathcal{D} \Downarrow_c$. We conclude that $\Gamma \vdash \mathcal{C} \lessapprox \mathcal{D} :: \Delta$ by proposition 7.3.12. □

Corollary 7.3.16. *If* $\Gamma \vdash \mathcal{C} \left(\leq_{(\mathcal{X}^E,\leqslant)}\right)^c \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \lesssim^c \mathcal{D} :: \Delta$.

The converse of proposition 7.3.15 is false. This is because barbs do not distinguish between sent messages, but only identify that a message was sent. Concretely,

$$\cdot \vdash \mathsf{proc}(c, c.0;\ \mathsf{close}\ c) \lessapprox \mathsf{proc}(c, c.1;\ \mathsf{close}\ c) :: c : \oplus\{0 : \mathbf{1}, 1 : \mathbf{1}\},$$

but

$$\begin{aligned}
&\langle \cdot \vdash \mathsf{proc}(c, c.0;\ \mathsf{close}\ c) :: c : \oplus\{0 : \mathbf{1}, 1 : \mathbf{1}\} \rangle \\
&= (c : (0, \mathsf{close})) \\
&\not\leqslant (c : (1, \mathsf{close})) \\
&= \langle \cdot \vdash \mathsf{proc}(c, c.1;\ \mathsf{close}\ c) :: c : \oplus\{0 : \mathbf{1}, 1 : \mathbf{1}\} \rangle.
\end{aligned}$$

Despite this, the converse of corollary 7.3.16 holds under certain reasonable hypotheses. Assume that $\Gamma \vdash \mathcal{C} \lesssim^c \mathcal{D} :: \Xi$. We must show that $\Lambda \vdash \mathcal{C} \left(\leq_{(\mathcal{X}^E,\leqslant)}\right)^c \mathcal{D} :: \Xi$. By proposition 7.0.8, it is sufficient to show that $\Lambda \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{D} :: \Xi$. We reduce this problem to the following:

Problem 7.3.17. Consider an observed communications $\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle = (c : v_c)_c$. Can we construct an experiment context[2] $\widehat{\Gamma} \vdash \mathcal{E}[\cdot]^\Gamma_\Delta :: \widehat{\Delta}$ (for some $\widehat{\Gamma}$ and $\widehat{\Delta}$ determined from Γ and Δ) such that for all $\Gamma \vdash \mathcal{D} :: \Delta$, $\mathcal{E}[\mathcal{D}]^\Gamma_\Delta \Downarrow_{\widehat{c}}$ for all $\widehat{c} : \widehat{A} \in \widehat{\Gamma}, \widehat{\Delta}$ if and only if $\Gamma \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{D} :: \Delta$? ◂

Indeed, given a solution to problem 7.3.17, we can show that $\Lambda \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{D} :: \Xi$ as follows. By reflexivity, $\Lambda \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{C} :: \Xi$. By construction, $\mathcal{E}[\mathcal{C}]^\Gamma_\Delta \Downarrow_{\widehat{c}}$ for all $\widehat{c} : \widehat{A} \in \widehat{\Gamma}, \widehat{\Delta}$. By assumption, $\widehat{\Gamma} \vdash \mathcal{E}[\mathcal{C}]^\Gamma_\Delta \lessapprox \mathcal{E}[\mathcal{D}]^\Gamma_\Delta :: \widehat{\Xi}$. So by proposition 7.3.12, $\mathcal{E}[\mathcal{D}]^\Gamma_\Delta \Downarrow_{\widehat{c}}$ for all $\widehat{c} : \widehat{A} \in \widehat{\Gamma}, \widehat{\Delta}$. Then $\Gamma \vdash \mathcal{C} \leq_{(\mathcal{X}^E,\leqslant)} \mathcal{D} :: \Delta$ by construction of $\mathcal{E}[\cdot]^\Gamma_\Delta$.

The answer to problem 7.3.17 is "no, but almost". A finite number of experiment contexts is insufficient in the presence of recursion and channel transmission (cf. [Hen83, p. 38]). We will instead construct a families of contexts, one for each height n approximation of the communications in $(c : v_c)_c$. The general result will then follow by proposition 6.1.12.

For brevity, we write $\leq$ for $\leq/\leqslant$ in the remainder of this section.

Before giving the full construction of experiment contexts, we illustrate the approach by a sequence of examples. In these examples, we only consider configurations of the form $\cdot \vdash \mathcal{C} :: a : A$. Given a channel name a, let $\widehat{a}$ be a globally fresh channel name. Let $\mathbb{Y}^+$ be the "positive answer type" $\oplus\{\mathsf{y} : \mathbf{o}^+\}$, where $\mathbf{o}^+$ is the positive empty type $\oplus\{\}$. We will construct non-empty sets $\mathcal{E}(n, v \varepsilon A)$ of processes of type $\cdot ; a : A \vdash E :: \widehat{a} : \mathbb{Y}^+$. They will satisfy the following weakened form of proposition 7.3.22:

Proposition. *Fix some* $v \varepsilon A$. *Let* $\cdot \vdash \mathcal{C} :: a : A$ *be arbitrary, and set* $w = \langle \cdot \vdash \mathcal{C} :: a : A \rangle(a)$. *For all* n, $\lfloor v \rfloor_{n+1} \leq w \varepsilon A$ *if and only if for all* $E \in \mathcal{E}(n, v \varepsilon A)$, $\langle \cdot \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, E) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathsf{y}, \bot)$.

[2] Though they serve similar purposes, these should not be confused with the experiments of observation systems.

In particular, by proposition 7.3.6, $\langle \cdot \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, E) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot)$ if and only if $(\mathcal{C}, \mathsf{proc}(\widehat{a}, E)) \Downarrow_{\widehat{a}}$. The set $\mathcal{E}(n, v \in A)$ checks that $\lfloor v \rfloor_{n+1} \leq w \in A$ instead of $\lfloor v \rfloor_n \leq w \in A$ because it is always the case that $\lfloor w \rfloor_0 \leq v \in A$, and doing so simplifies the definition. These sets of processes induce sets of experiment contexts in the obvious way. Let Y_c be the process $c.\mathtt{y};\ \Omega$, where Ω is given at each type by example 5.3.3.

Example 7.3.18. Set $v = (k, \mathsf{close})$ and $A = \oplus\{k : \mathbf{1}, l : \mathbf{1}\}$. Set $w = \langle \cdot \vdash \mathcal{C} :: a : \oplus\{k : \mathbf{1}, l : \mathbf{1}\} \rangle(a)$ for some configuration $\mathcal{C}$.

Recall that $\lfloor v \rfloor_1 = (k, \bot)$, and observe that $\lfloor (k, \mathsf{close}) \rfloor_1 \leq w \in A$ if and only if $\mathcal{C}$ sent the label k on a, so if and only if

$$\langle \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{case}\ a\ \{k \Rightarrow Y_{\widehat{a}} \mid \_ \Rightarrow \Omega\}) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot).$$

Take $\mathcal{E}(\mathtt{o}, v \in A) = \{\mathsf{case}\ a\ \{k \Rightarrow Y_{\widehat{c}} \mid \_ \Rightarrow \Omega\}\}$.

Now observe that $\lfloor v \rfloor_2 \leq w \in A$ if and only if

$$\langle \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{case}\ a\ \{k \Rightarrow \mathsf{wait}\ c;\ Y_{\widehat{c}} \mid \_ \Rightarrow \Omega\}) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot).$$

Take $\mathcal{E}(1, v \in A) = \{\mathsf{case}\ a\ \{k \Rightarrow \mathsf{wait}\ a;\ Y_{\widehat{c}} \mid \_ \Rightarrow \Omega\}\}$. Because $\lfloor v \rfloor_n = \lfloor v \rfloor_2$ for $n \geq 2$, take $\mathcal{E}(n, v \in A) = \mathcal{E}(1, v \in A)$ for all $n \geq 1$. ◀

Next, we illustrate why $\mathcal{E}(n, v \in A)$ must be a set.

Example 7.3.19. Set $v = ((\mathtt{l}, \mathsf{close}), (\mathtt{r}, \bot))$ and $A = (\oplus\{\mathtt{l} : \mathbf{1}\}) \otimes (\oplus\{\mathtt{r} : \mathbf{1}\})$. Set $w = \langle \vdash \mathcal{C} :: a : A \rangle(a)$ for some configuration $\mathcal{C}$. Clearly, $\lfloor v \rfloor_1 = (\bot, \bot) \leq w \in A$ if and only if

$$\langle \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, a \leftarrow \mathsf{recv}\ c;\ Y_{\widehat{c}}) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot).$$

Our task becomes harder when we consider $\lfloor v \rfloor_2 = ((\mathtt{l}, \bot), (\mathtt{r}, \bot))$: we must somehow inspect the communications on a and also those on c, and return a result on $\widehat{a} : \mathbb{Y}^+$. We do so by using two experiment contexts. Indeed, $\lfloor v \rfloor_2 \leq w \in A$ if and only if both

$$\langle \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, a \leftarrow \mathsf{recv}\ c;\ \mathsf{case}\ c\ \{\mathtt{l} \Rightarrow Y_{\widehat{a}}\}) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot),$$
$$\langle \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, a \leftarrow \mathsf{recv}\ c;\ \mathsf{case}\ a\ \{\mathtt{r} \Rightarrow Y_{\widehat{a}}\}) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot).$$

Accordingly, we take

$$\mathcal{E}(1, v \in A) = \{a \leftarrow \mathsf{recv}\ c;\ \mathsf{case}\ c\ \{\mathtt{l} \Rightarrow Y_{\widehat{a}}\}, a \leftarrow \mathsf{recv}\ c;\ \mathsf{case}\ a\ \{\mathtt{r} \Rightarrow Y_{\widehat{a}}\}\}$$ ◀

Having illustrated the approach, we define the family $\mathcal{E}_R(n, i, r, v \in A)$ of experiment processes by induction on n and recursion on $v \in A$, where i is the "input channel" whose communications we are examining (a in the above examples), and r is the "results channel" ($\widehat{a}$ in the above examples). We can lift these experiment processes to testing contexts in the obvious manner. The family $\mathcal{E}_R(n, i, r, v \in A)$ checks that communications w on i satisfy $\lfloor v \rfloor_{n+1} \leq w \in A$.

We maintain the invariant that if $E \in \mathcal{E}_R(n, i, r, v \in A)$, then $\cdot ; i : A \vdash E :: r : \mathbb{Y}^+$ and E can be weakened[3] to $\cdot ; \Delta, i : A \vdash E :: r : \mathbb{Y}^+$ for all Δ not mentioning i or r. In particular, the processes in $\mathcal{E}_R(n, i, r, v \in A)$ always listen from left and report results on the right; we will consider the symmetric case $\mathcal{E}_L(n, i, r, v \in A)$ later. We also maintain the invariant that $E \in \mathcal{E}_R(n, i, r, v \in A)$ if

[3]Though the type system is linear, the present of unbounded recursion allows us to ignore channels in non-terminating processes.

and only if $[i'/i]E \in \mathcal{E}_R(n, i', r, v \varepsilon A)$ for $i \neq r$ and $i' \neq r$.

$$\begin{aligned}
\mathcal{E}_R(n, i, r, \bot \varepsilon A) &= \{Y_r\} \\
\mathcal{E}_R(n, i, r, \mathsf{close} \varepsilon \mathbf{1}) &= \{\mathsf{wait}\ i;\ Y_r\} \\
\mathcal{E}_R(\mathsf{o}, i, r, (k, _) \varepsilon \oplus\{l : A_l\}_{l \in L}) &= \{\mathsf{case}\ i\ \{k \Rightarrow Y_r \mid _ \Rightarrow \Omega\}\} \\
\mathcal{E}_R(n+1, i, r, (k, v) \varepsilon \oplus\{l : A_l\}_{l \in L}) &= \{\mathsf{case}\ i\ \{k \Rightarrow E \mid _ \Rightarrow \Omega\} \mid E \in \mathcal{E}_R(n, i, r, v \varepsilon A_k)\} \\
\mathcal{E}_R(\mathsf{o}, i, r, (\mathsf{unfold}, _) \varepsilon \rho\alpha.A) &= \{\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ Y_r\} \\
\mathcal{E}_R(n+1, i, r, (\mathsf{unfold}, v) \varepsilon \rho\alpha.A) &= \{\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ E \mid E \in \mathcal{E}_R(n, i, r, v \varepsilon [\rho\alpha.A/\alpha]A)\} \\
\mathcal{E}_R(\mathsf{o}, i, r, (\mathsf{shift}, _) \varepsilon \uparrow A) &= \{\mathsf{shift} \leftarrow \mathsf{recv}\ i;\ Y_r\} \\
\mathcal{E}_R(n+1, i, r, (\mathsf{shift}, v) \varepsilon \uparrow A) &= \{\mathsf{shift} \leftarrow \mathsf{recv}\ i;\ E \mid E \in \mathcal{E}_R(n, i, r, v \varepsilon A)\} \\
\mathcal{E}_R(\mathsf{o}, i, r, (_, _) \varepsilon A \otimes B) &= \{_ \leftarrow \mathsf{recv}\ i;\ Y_r\} \\
\mathcal{E}_R(n+1, i, r, (u, v) \varepsilon A \otimes B) &= \{_ \leftarrow \mathsf{recv}\ i;\ E \mid E \in \mathcal{E}_R(n, i, r, v \varepsilon B)\} \cup \\
&\quad \cup \{a \leftarrow \mathsf{recv}\ i;\ E \mid E \in \mathcal{E}_R(n, a, r, u \varepsilon A)\}
\end{aligned}$$

Conspicuously absent are negative protocols. Because any provided channel with a negative protocol is an input channel, we can only observe $\bot$ on that channel by proposition 7.3.1. Accordingly, we define:

$$\mathcal{E}_R(n, i, r, v \varepsilon A^-) = \begin{cases} \{Y_r\} & v = \bot \\ \{\Omega\} & v \neq \bot \end{cases}$$

Also absent are sets of experiment processes for the protocol $\tau \wedge A$. We use an oracle process to check if two transmitted values are related by $\leqslant$.

Definition 7.3.20. Let P be a predicate on functional values of type τ. An **oracle process for** P is a process $\cdot\,;\cdot \vdash O :: c : \tau \supset \uparrow \oplus \{\mathtt{tt} : \mathbf{1}, \mathtt{ff} : \mathbf{1}\}$ that receives a functional value w and a shift message[4] over c, and sends $\mathtt{tt}$ if $P(w)$, and $\mathtt{ff}$ otherwise; and closes the channel in both cases. Explicitly,

$$\begin{aligned}
&\mathsf{proc}(c, O), \mathsf{msg}(d, _ \leftarrow \mathsf{output}\ c\ w;\ c \leftarrow d), \mathsf{msg}(e, \mathsf{send}\ d\ \mathsf{shift};\ d \to e) \to^* \\
&\to^* \exists f. \begin{cases} \mathsf{msg}(f, \mathsf{close}\ f), \mathsf{msg}(e, e.\mathsf{tt};\ f \to e) & P(w) \\ \mathsf{msg}(f, \mathsf{close}\ f), \mathsf{msg}(e, e.\mathsf{ff};\ f \to e) & \text{otherwise} \end{cases}
\end{aligned}$$

◀

Assumption 7.3.21. Assume that for all τ and values $\cdot \Vdash v : \tau$, there is an oracle $O^{\leqslant}_{v:\tau}$ for the predicate $\cdot \Vdash v \leqslant (-) : \tau$.

Using the oracle, we define:

$$\begin{aligned}
\mathcal{E}_R(\mathsf{o}, i, r, (\mathsf{val}\ f, _) \varepsilon \tau \wedge A) &= \{c \leftarrow O_{f:\tau};\ x \leftarrow \mathsf{input}\ i;\ _ \leftarrow \mathsf{output}\ c\ x;\ \mathsf{send}\ c\ \mathsf{shift}; \\
&\qquad \mathsf{case}\ c\ \{\mathtt{tt} \Rightarrow \mathsf{wait}\ c;\ Y_r \mid \mathtt{ff} \Rightarrow \Omega\}\} \\
\mathcal{E}_R(n+1, i, r, (\mathsf{val}\ f, u) \varepsilon \tau \wedge A) &= \{c \leftarrow O_{f:\tau};\ x \leftarrow \mathsf{input}\ i;\ _ \leftarrow \mathsf{output}\ c\ x;\ \mathsf{send}\ c\ \mathsf{shift}; \\
&\qquad \mathsf{case}\ c\ \{\mathtt{tt} \Rightarrow \mathsf{wait}\ c;\ E \mid \mathtt{ff} \Rightarrow \Omega\} \mid E \in \mathcal{E}_R(n, i, r, u \varepsilon A)\}
\end{aligned}$$

Proposition 7.3.22. *Let $\Gamma \vdash \mathcal{C} :: a : A$ and $v \varepsilon A$ be arbitrary. Set $w = \langle \Gamma \vdash \mathcal{C} :: \Delta, a : A \rangle(a)$ and let $\widehat{a}$ be globally fresh. Then for all n, $\lfloor v \rfloor_{n+1} \leqslant w \varepsilon A$ if and only if for all $E \in \mathcal{E}_R(n, a, \widehat{a}, v \varepsilon A)$,*

$$\langle \Gamma \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, E) :: \Delta, \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathsf{y}, \bot).$$

Proof. By induction on n. Assume first that $n = \mathsf{o}$. Then we proceed by case analysis on $v \varepsilon A$. We give the representative cases; the rest will follow by analogy.

Case $\bot \varepsilon A$**:** The result is immediate.

Case $v \varepsilon A^-$**:** Then $w = \bot$ by proposition 7.3.1. If $v = \bot$, then the result is immediate. If $v \neq \bot$, then it is not the case that $v \leqslant w \varepsilon A$, and the result also follows from the definition of the divergent process Ω.

[4] Because types are polarized, the up shift is required to ensure that the type is well-formed.

Case $\mathsf{close} \;\varepsilon\; \mathbf{1}$: Then $\lfloor \mathsf{close} \rfloor_1 = \mathsf{close}$. By inversion, $\mathsf{close} \le w \;\varepsilon\; A$ if and only if $w = \mathsf{close}$. This is the case if and only if $\mathsf{msg}(a, \mathsf{close}\; a)$ appears in a fair trace of $\mathcal{C}$. The only element of $\mathcal{E}_R(n, a, \widehat{a}, \mathsf{close} \;\varepsilon\; \mathbf{1})$ is $\{\mathsf{wait}\; a;\; Y_{\widehat{a}}\}$. The fact $\mathsf{msg}(a, \mathsf{close}\; a)$ appears in a fair trace of $\mathcal{C}$ if and only if every fair trace of $\mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{wait}\; a;\; Y_{\widehat{a}})$ has an instantiation

$$\mathsf{msg}(a, \mathsf{close}\; a), \mathsf{proc}(\widehat{a}, \mathsf{wait}\; a;\; Y_{\widehat{a}}) \to \mathsf{proc}(\widehat{a}, Y_{\widehat{a}})$$

of rule (67). By fairness, $Y_{\widehat{a}}$ produces the observation $(\mathtt{y}, \bot)$ on $\widehat{a}$. So $\mathsf{msg}(a, \mathsf{close}\; a)$ appears in a fair trace of $\mathcal{C}$ if and only if $\langle \Gamma \vdash \mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{wait}\; a;\; Y_{\widehat{a}}) :: \widehat{a} : \mathbb{Y}^+ \rangle(\widehat{a}) = (\mathtt{y}, \bot)$. This gives the result.

Case $(k, u) \;\varepsilon\; \oplus\{l : A_l\}_{l \in L}$: Then $\lfloor (k, u) \rfloor_1 = (k, \bot)$. By inversion, $(k, \bot) \le w \;\varepsilon\; A$ if and only if $w = (k, w')$ for some w'. This is the case if and only if $\mathsf{msg}(a, a.k;\; \_ \to a)$ appears in a fair trace of $\mathcal{C}$. The only element of $\mathcal{E}_R(\mathtt{o}, a, \widehat{a}, (k, u) \;\varepsilon\; A$ is $\mathsf{case}\; a\; \{k \Rightarrow Y_{\widehat{a}} \mid \_ \Rightarrow \Omega\}$. The aforementioned fact appears in a fair trace of $\mathcal{C}$ if and only if every fair trace of $\mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{case}\; a\; \{k \Rightarrow Y_{\widehat{a}} \mid \_ \Rightarrow \Omega\})$ has an instantiation

$$\mathsf{msg}(a, a.k;\; \_ \to a), \mathsf{proc}(\widehat{a}, \mathsf{case}\; a\; \{k \Rightarrow Y_{\widehat{a}} \mid \_ \Rightarrow \Omega\}) \to \mathsf{proc}(\widehat{a}, Y_{\widehat{a}})$$

of rule (79). The remainder is analogous to the previous case.

Case $(\mathsf{val}\; f, u) \;\varepsilon\; \tau \wedge A$: Then $\lfloor (\mathsf{val}\; f, u) \rfloor_1 = (\mathsf{val}\; f, \bot)$. By inversion, $(\mathsf{val}\; f, \bot) \le w \;\varepsilon\; \tau \wedge B$ if and only if $w = (\mathsf{val}\; g, w')$ for some g and w' and $\cdot \Vdash f \leqslant g : \tau$. This is the case if and only if $\mathsf{msg}(a, \_ \leftarrow \mathsf{output}\; a\; g;\; \_ \to a)$ appears in a fair trace of $\mathcal{C}$ with g satisfying the above relation. The only element of $\mathcal{E}_R(\mathtt{o}, a, \widehat{a}, (\mathsf{val}\; f, u) \;\varepsilon\; A)$ is

$$c \leftarrow O_{f:\tau};\; x \leftarrow \mathsf{input}\; a;\; \_ \leftarrow \mathsf{output}\; c\; x;\; \mathsf{send}\; c\; \mathsf{shift};\; \mathsf{case}\; c\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; c;\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\}.$$

Every fair trace of $\mathcal{C}$ has an instantiation

$$\begin{aligned} &\mathsf{proc}(\widehat{a}, c \leftarrow O_{f:\tau};\; x \leftarrow \mathsf{input}\; a; \\ &\qquad \_ \leftarrow \mathsf{output}\; c\; x;\; \mathsf{send}\; c\; \mathsf{shift};\; \mathsf{case}\; c\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; c;\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\}) \to \\ &\to \mathsf{proc}(c', [c'/c]O_{f:\tau}), \mathsf{proc}(\widehat{a}, x \leftarrow \mathsf{input}\; a; \\ &\qquad \_ \leftarrow \mathsf{output}\; c'\; x;\; \mathsf{send}\; c'\; \mathsf{shift};\; \mathsf{case}\; c'\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; c';\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\}) \end{aligned}$$

of rule (66). The aforementioned message fact appears in a fair trace of $\mathcal{C}$ if and only if the trace contains the following instantiation of rule (75):

$$\begin{aligned} &(\mathsf{msg}(a, \_ \leftarrow \mathsf{output}\; a\; g;\; \_ \to a), \mathsf{proc}(\widehat{a}, x \leftarrow \mathsf{input}\; a; \\ &\qquad \_ \leftarrow \mathsf{output}\; c'\; x;\; \mathsf{send}\; c'\; \mathsf{shift};\; \mathsf{case}\; c'\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; c';\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\})) \to \\ &\to (\mathsf{proc}(\widehat{a}, \_ \leftarrow \mathsf{output}\; c'\; g;\; \mathsf{send}\; c'\; \mathsf{shift};\; \mathsf{case}\; c'\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; c';\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\}))\,, \end{aligned}$$

and of rules (5.B) and (77) (not necessarily in immediate succession of each other):

$$\begin{aligned} &(\mathsf{proc}(\widehat{a}, \_ \leftarrow \mathsf{output}\; c'\; g;\; \mathsf{send}\; c'\; \mathsf{shift};\; \mathsf{case}\; c'\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; c';\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\})) \to^* \\ &\qquad \to^* (\mathsf{msg}(d, \_ \leftarrow \mathsf{output}\; c'\; g;\; c' \leftarrow d), \mathsf{msg}(e, \mathsf{send}\; d\; \mathsf{shift};\; d \to e), \\ &\qquad\qquad \mathsf{proc}(\widehat{a}, \mathsf{case}\; e\; \{\mathtt{tt} \Rightarrow \mathsf{wait}\; e;\; Y_r \mid \mathtt{ff} \Rightarrow \Omega\}))\,. \end{aligned}$$

By fairness and the definition of the oracle, the above hold if and only if the oracle takes the steps

$$\begin{aligned} &\big(\mathsf{proc}(c', [c'/c]O_{f:\tau}), \mathsf{msg}(d, \_ \leftarrow \mathsf{output}\; c'\; g;\; c' \leftarrow d), \mathsf{msg}(e, \mathsf{send}\; d\; \mathsf{shift};\; d \to e)\big) \to^* \\ &\qquad \to^* \mathsf{msg}(h, \mathsf{close}\; h), \mathsf{msg}(c', c'.\mathtt{tt};\; h \to c'). \end{aligned}$$

From here, the proof is analogous to the previous cases.

Now assume that the result holds for some n. We show the inductive step $n + 1$, again by case analysis on $v \;\varepsilon\; A$. We give the representative case; the rest follow by analogy with this case or with base cases.

Case $(k, u) \;\varepsilon\; \oplus\{l : A_l\}_{l \in L}$: The elements of $\mathcal{E}_R(n+1, a, \widehat{a}, (k, u) \;\varepsilon\; \oplus\{l : A_l\}_{l \in L})$ are of the form $\mathsf{case}\; a\; \{k \Rightarrow E \mid \_ \Rightarrow \Omega\}$ for $E \in \mathcal{E}_R(n, a, \widehat{a}, u \;\varepsilon\; A_k)$. Recall that $\lfloor (k, u) \rfloor_{n+2} = (k, \lfloor u \rfloor_{n+1})$. By inversion, $(k, \lfloor u \rfloor_{n+1}) \le w \;\varepsilon\; \oplus\{l : A_l\}_{l \in L}$ if and only if $w = (k, w')$ and $\lfloor u \rfloor_{n+1} \le w' \;\varepsilon\; A_k$. But $w = (k, w')$ if and only if there is an observable message $\mathsf{msg}(a, a.k;\; d \to a)$ in a fair trace of $\mathcal{C}$

for some channel d with $T \rightsquigarrow w' \varepsilon A_k \,/\, d$. There is a message $\mathsf{msg}(a, a.k;\ d \to a)$ in a fair trace of $\mathcal{C}$ if and only if there is such a message in a fair trace of $\mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{case}\ a\ \{k \Rightarrow E \mid \_ \Rightarrow \Omega\})$. Fairness implies that a fair trace of $\mathcal{C}, \mathsf{proc}(\widehat{a}, \mathsf{case}\ a\ \{k \Rightarrow E \mid \_ \Rightarrow \Omega\})$ has an instantiation

$$\mathsf{msg}(a, a.k;\ d \to a), \mathsf{proc}(\widehat{a}, \mathsf{case}\ a\ \{k \Rightarrow E \mid \_ \Rightarrow \Omega\}) \to \mathsf{proc}(\widehat{a}, [d/a]E)$$

of rule (79) if and only if $\mathsf{msg}(a, a.k;\ d \to a)$ appears in the trace. If this is the case, then we apply the induction hypothesis: $T \rightsquigarrow (\mathrm{y}, \bot) \varepsilon \mathbb{Y}^+ \,/\, \widehat{a}$ for all $[d/a]E \in \mathcal{E}_R(n, d, \widehat{a}, u \varepsilon A_k)$ if and only if $\lfloor u \rfloor_{n+1} \leq w' \varepsilon A_k$, where we recall that w' is given by $T \rightsquigarrow w' \varepsilon A_k \,/\, d$. It follows that

$$\langle \mathcal{C}, \mathsf{proc}(\widehat{a}, T) \rangle(\widehat{a}) = (\mathrm{y}, \bot),$$

for all processes $T \in \mathcal{E}_R(n+1, a, \widehat{a}, (k, u) \varepsilon \oplus\{l : A_l\}_{l \in L}$ if and only if both $w = (k, w')$ and $\lfloor u \rfloor_{n+1} \leq w' \varepsilon A_k$, i.e., if and only if $\lfloor (k, u) \rfloor_{n+2}) \leq w \varepsilon \oplus\{l : A_l\}_{l \in L}$. □

Let $\mathbf{o}^-$ be the negative empty type $\&\{\}$ and let $\mathbb{Y}^-$ be the "negative answer type" $\&\{\mathrm{y} : \mathbf{o}^-\}$. We can dualize the definition of $\mathcal{E}_R$ to get a family $\mathcal{E}_L$ of processes that listens on the right and reports on the left. In fact, the processes carry over unchanged:[5]

$$\begin{aligned}
\mathcal{E}_L(n, i, r, \bot \varepsilon A) &= \mathcal{E}_R(n, i, r, \bot \varepsilon A) \\
\mathcal{E}_L(n, i, r, (k, v) \varepsilon \&\{l : A_l\}_{l \in L}) &= \mathcal{E}_R(n, i, r, (k, v) \varepsilon \oplus\{l : A_l\}_{l \in L}) \\
\mathcal{E}_L(n, i, r, (\mathsf{unfold}, v) \varepsilon \rho\alpha.A) &= \mathcal{E}_R(n, i, r, (\mathsf{unfold}, v) \varepsilon \rho\alpha.A) \\
\mathcal{E}_L(n, i, r, (\mathsf{shift}, v) \varepsilon \downarrow A) &= \mathcal{E}_R(n, i, r, (\mathsf{shift}, v) \varepsilon \uparrow A) \\
\mathcal{E}_L(n, i, r, (u, v) \varepsilon A \multimap B) &= \mathcal{E}_R(n, r, (u, v) \varepsilon A \otimes B) \\
\mathcal{E}_L(n, i, r, v \varepsilon A^+) &= \mathcal{E}_R(n, i, r, v \varepsilon B^-) \\
\mathcal{E}_L(n, i, r, (\mathsf{val}\ f, u) \varepsilon \tau \wedge A) &= \mathcal{E}_R(n, i, r, (\mathsf{val}\ f, u) \varepsilon \tau \supset A).
\end{aligned}$$

The proof of the following proposition is analogous to the proof of proposition 7.3.22.

Proposition 7.3.23. *Let $\Gamma, a : A \vdash \mathcal{C} :: \Delta$ and $v \varepsilon A$ be arbitrary. Set $w = \langle \Gamma, a : A \vdash \mathcal{C} :: \Delta \rangle(a)$ and let $\widehat{a}$ be globally fresh. Then for all n, $\lfloor v \rfloor_{n+1} \leq w \varepsilon A$ if and only if for all $E \in \mathcal{E}_L(n, a, \widehat{a}, w \varepsilon A)$,*

$$\langle \Gamma, \widehat{a} : \mathbb{Y}^- \vdash \mathsf{proc}(\widehat{a}, E), \mathcal{C} :: \Delta \rangle(\widehat{a}) = (\mathrm{y}, \bot).$$

We combine propositions 7.3.22 and 7.3.23 to build families of experiment contexts. Given n observations $\overline{v_i \varepsilon A_i / a_i}$ and m observations $\overline{w_j \varepsilon C_j / c_j}$ with $n \geq 0$ and $m \geq 1$, define the set of configuration contexts:

$$\begin{aligned}
&\mathcal{E}\left(n, \overline{v_i \varepsilon A_i / a_i}, \overline{w_j \varepsilon C_j / c_j}\right) \\
&= \Big\{ \mathsf{proc}(a_1, L_1), \ldots, \mathsf{proc}(a_n, L_n), [\cdot]^{\overline{a_i : Ai}}_{\overline{c_j : C_j}}, \mathsf{proc}(\widehat{c_1}, R_1), \ldots, \mathsf{proc}(\widehat{c_m}, R_m) \mid \\
&\qquad \mid L_i \in \mathcal{E}_L\left(n, a_i, \widehat{a_i}, v_i \varepsilon A_i\right), R_j \in \mathcal{E}_R\left(n, c_j, \widehat{c_j}, w_j \varepsilon C_j\right) \Big\}.
\end{aligned}$$

Proposition 7.3.24. *Let $\Gamma \vdash \mathcal{C} :: \Delta$ and $\Gamma \vdash \mathcal{D} :: \Delta$ be arbitrary. If $\tau \wedge A$ is a subphrase of a type in Δ or $\tau \supset A$ is a subphrase of a type in Γ, then for each value $\cdot \Vdash v : \tau$ assume the existence of an oracle process $O^{\leqslant}_{v:\tau}$ satisfying assumption 7.3.21. Let v_i and w_j be given by:*

$$\begin{aligned}
\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle_\Gamma &= (a_i : v_i)_{a_i : A_i \in \Gamma}, \\
\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle_\Delta &= (c_j : w_j)_{c_j : C_j \in \Delta}.
\end{aligned}$$

Then

$$\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle_{\Gamma,\Delta} \lesseqgtr \langle \Gamma \vdash \mathcal{D} :: \Delta \rangle_{\Gamma,\Delta}$$

if and only if for all n and $\mathcal{F}[\cdot]^\Gamma_\Delta \in \mathcal{E}(n, \overline{v_i \varepsilon A_i / a_i}, \overline{w_j \varepsilon C_j / c_j})$,

$$\left\langle \overline{\widehat{a_i} : \mathbb{Y}^-} \vdash \mathcal{F}[\mathcal{D}]^\Gamma_\Delta :: \overline{\widehat{c_j} : \mathbb{Y}^+} \right\rangle(b) = (\mathrm{y}, \bot)$$

for all $b \in \overline{\widehat{a_i}}, \overline{\widehat{c_j}}$.

[5] Here, we are taking an extrinsic [Rey98, § 15.4] or "Curry-style" view of process typing.

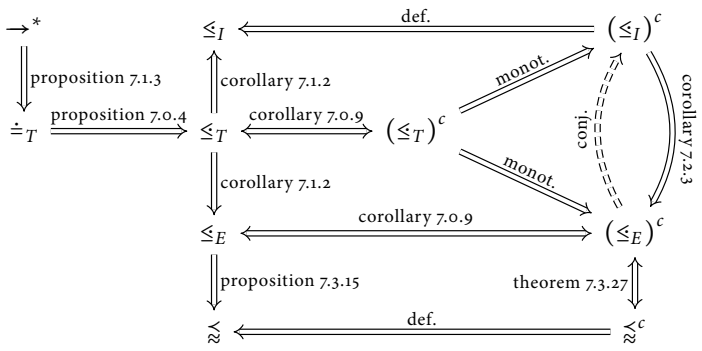


FIGURE 7.1. Relationship between relations of section 7.3

Proof. By definition,

$$\langle \Gamma \vdash \mathcal{C} :: \Delta \rangle_{\Gamma,\Delta} \leq\!\!/\!\!\leqslant \langle \Gamma \vdash \mathcal{D} :: \Delta \rangle_{\Gamma,\Delta}$$

if and only if

$$(a_i : v_i, c_j : w_j)_{a_i:A_i \in \Gamma, c_j:C_j \in \Delta} \leq\!\!/\!\!\leqslant \langle \Gamma \vdash \mathcal{D} :: \Delta \rangle_{\Gamma,\Delta}.$$

By proposition 6.1.12, this is the case if and only if for all n,

$$(a_i : \lfloor v_i \rfloor_n, c_j : \lfloor w_j \rfloor_n)_{a_i:A_i \in \Gamma, c_j:C_j \in \Delta} \leq\!\!/\!\!\leqslant \langle \Gamma \vdash \mathcal{D} :: \Delta \rangle_{\Gamma,\Delta}.$$

The result then follows by fairness and propositions 7.3.22 and 7.3.23. □

Proposition 7.3.24 and the analysis following problem 7.3.17 imply:

COROLLARY 7.3.25. *For each value* $\cdot \Vdash v : \tau$ *assume the existence of an oracle process* $O^{\leqslant}_{v:\tau}$ *satisfying assumption 7.3.21. If* $\Gamma \vdash \mathcal{C} \lessapprox^c \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \leq_{(\mathcal{X}^E, \leqslant)} \mathcal{D} :: \Delta$.

COROLLARY 7.3.26. *If* $\Gamma \vdash \mathcal{C} \lessapprox^c \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \leq_E \mathcal{D} :: \Delta$.

Proof. By corollary 7.3.25. The oracle $O^{\mathfrak{U}}_{v:\tau}$ assumed by corollary 7.3.25 is given by:

$$\cdot ; \cdot \vdash \_ \leftarrow \mathsf{input}\ c;\ \mathsf{shift} \leftarrow \mathsf{recv}\ c;\ c.\mathtt{tt};\ \mathsf{close}\ c :: c : \tau \supset \uparrow \oplus \{\mathtt{tt} : \mathbf{1}, \mathtt{ff} : \mathbf{1}\}.$$ □

Combining corollary 7.3.26 and proposition 7.3.15 gives:

THEOREM 7.3.27. *For all* $\Gamma \vdash \mathcal{C} :: \Delta$ *and* $\Gamma \vdash \mathcal{D} :: \Delta$, $\Gamma \vdash \mathcal{C} \lessapprox^c \mathcal{D} :: \Delta$ *if and only if* $\Gamma \vdash \mathcal{C} \leq_E \mathcal{D} :: \Delta$.

7.4. Summary of Relations

Figure 7.1 summarizes the main results for relations on configurations. Double arrows denote implications. Dashed arrows denote conjectured implications. Missing arrows (when not implied by transitivity) indicate falsehoods. We recall that:

- $\to^*$ is the reflexive, transitive closure of $\to$;
- $\leq_I$ is internal observational simulation;
- $(\leq_I)^c$ is internal observational precongruence;
- $\doteq_T$ is total observational equivalence;
- $\leq_T$ is total observational simulation;
- $(\leq_T)^c$ is total observational precongruence;
- $\leq_E$ is external observational simulation;
- $(\leq_E)^c$ is external observational precongruence;
- $\lessapprox$ is barbed simulation;
- $\lessapprox^c$ is barbed precongruence.

7.5. Precongruences for Processes

In this section, we relate relations on configurations to relations on processes. Recall that to show that two configurations are total or external precongruent, it is sufficient by proposition 7.0.12 to consider only simply branched contexts. In section 7.5.1, we show that simply branched configuration contexts closely mirror "observation contexts" for processes. In section 7.5.2, we show how to lift relations from configurations to processes.

7.5.1. Relating Simply Branched Contexts and Observation Contexts. Observation contexts characterize the idea of processes experimenting on processes through communication:

Definition 7.5.1. An **observation context** is a typed context derived using exactly one instance of the axiom (P-Hole), plus zero or more instances of the derived rules (Hole-Cut-L) and (Hole-Cut-R),

$$\frac{\cdot\,;\,\Delta_1 \vdash O[\cdot]^{\Delta}_{a:A} :: b : B \quad \cdot\,;\,\Delta_2, b : B \vdash P :: c : C}{\cdot\,;\,\Delta_1, \Delta_2 \vdash b \leftarrow O[\cdot]^{\Delta}_{a:A};\ P :: c : C}\ (\textsc{Hole-Cut-L})$$

$$\frac{\cdot\,;\,\Delta_1 \vdash P :: b : B \quad \cdot\,;\,b : B, \Delta_2 \vdash O[\cdot]^{\Delta}_{a:A} :: c : C}{\cdot\,;\,\Delta_1, \Delta_2 \vdash b \leftarrow P;\ O[\cdot]^{\Delta}_{a:A} :: c : C}\ (\textsc{Hole-Cut-R})$$

such that the context satisfies the grammar:

$$O[\cdot]^{\Delta}_{a:A} ::= [\cdot]^{\Delta}_{a:A} \mid b \leftarrow O[\cdot]^{\Delta}_{a:A};\ P \mid b \leftarrow P;\ O[\cdot]^{\Delta}_{a:A}. \qquad \blacktriangleleft$$

Recall from definition 5.7.6 the definition of a contextual relation on processes. "Observational contextuality" weakens this notion from arbitrary contexts to observation contexts:

Definition 7.5.2. A typed-indexed relation $\mathfrak{R}$ on processes is an **observationally contextual** if $\cdot\,;\,\Delta \vdash P\ \mathfrak{R}\ Q :: a : A$ implies $\cdot\ ;\ \Lambda \vdash O[P]^{\Delta}_{a:A}\ \mathfrak{R}\ O[Q]^{\Delta}_{a:A} :: b : B$ for all observation contexts $\cdot\,;\,\Lambda \vdash O[\cdot]^{\Delta}_{a:A} :: b : B$. $\blacktriangleleft$

Definition 7.5.3. The **observationally contextual interior** of a typed relation $\mathfrak{R}$ on processes is the greatest observationally contextual typed relation $\mathfrak{R}^{\mathcal{O}}$ contained in $\mathfrak{R}$. $\blacktriangleleft$

There is an obvious translation from observation contexts to configuration contexts, where we inductively map (P-Hole) to (Conf-H), and (Hole-Cut-L) and (Hole-Cut-R) to (Conf-C):

Proposition 7.5.4. *Let $\cdot\ ;\ \Lambda \vdash O[\cdot]^{\Delta}_{a:A} :: c : C$ be an observation context. There exists a configuration context $\Lambda \mid \mathrm{I} \vdash \mathcal{O}[\cdot]^{\Delta}_{a:A} :: c : C$ such that $\mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) \to^* \mathcal{O}[\mathsf{proc}(a, Q)]^{\Delta}_{a:A}$ for all processes $\cdot\,;\,\Delta \vdash Q :: a : A$.*

Proof. By induction on the derivation of the observation context.

Case (P-Hole)**:** Let $\mathcal{O}$ be given by (Conf-H). The step is given by reflexivity.

Case (Hole-Cut-L)**:** The observation context is $\cdot\,;\,\Delta_1, \Delta_2 \vdash b \leftarrow O[\cdot]^{\Delta}_{a:A};\ P :: c : C$, and it is formed by:

$$\frac{\cdot\,;\,\Delta_1 \vdash O[\cdot]^{\Delta}_{a:A} :: b : B \quad \cdot\,;\,\Delta_2, b : B \vdash P :: c : C}{\cdot\,;\,\Delta_1, \Delta_2 \vdash b \leftarrow O[\cdot]^{\Delta}_{a:A};\ P :: c : C}\ (\textsc{Hole-Cut-L})$$

By the induction hypothesis, there exists a configuration context $\Delta_1 \mid \mathrm{I}' \vdash \mathcal{O}'[\cdot]^{\Delta}_{a:A} :: b : B$ such that $\mathsf{proc}(b, O[Q]^{\Delta}_{a:A}) \to^* \mathcal{O}'[\mathsf{proc}(a, Q)]^{\Delta}_{a:A}$. This implies that

$$\begin{aligned}
&\mathsf{proc}(c, b \leftarrow O[Q]^{\Delta}_{a:A};\ P)\\
&\to \mathsf{proc}(b, O[Q]^{\Delta}_{a:A}), \mathsf{proc}(c, P)\\
&\to^* \mathcal{O}'[\mathsf{proc}(a, Q)]^{\Delta}_{a:A}, \mathsf{proc}(c, P).
\end{aligned}$$

Let $\mathcal{O}$ be given by:

$$\frac{\Delta_1 \mid \Gamma' \vdash \mathcal{O}'[\cdot]^{\Delta}_{a:A} :: b : B \quad \dfrac{\cdot\,;\,\Delta_2, b : B \vdash P :: c : C}{\Delta_2, b : B \mid \cdot \vdash \mathsf{proc}(c, P) :: c : C}\ (\textsc{Conf-P})}{\Delta_1, \Delta_2 \mid \Gamma', b : B \vdash \mathcal{O}'[\cdot]^{\Delta}_{a:A}, \mathsf{proc}(c, P) :: c : C}\ (\textsc{Conf-C})$$

Plugging $\mathsf{proc}(a, Q)$ into $\mathcal{O}$ gives the configuration

$$\Delta_1, \Delta_2 \mid \Gamma', b : B \vdash \mathcal{O}'[\mathsf{proc}(a, Q)]^{\Delta}_{a:A}, \mathsf{proc}(c, P) :: c : C$$

that we recognize as the result of the above sequence of rewrite steps. We conclude that

$$\mathsf{proc}(c, b \leftarrow O[Q]^{\Delta}_{a:A};\ P) \to^* \mathcal{O}[\mathsf{proc}(a, Q)]^{\Delta}_{a:A}.$$

Case (Hole-Cut-R)**:** This case is symmetric to the previous case. □

The translation in the opposite direction is more subtle. To translate configuration contexts to observation contexts, one must be able to translate configurations to processes. Naïvely, one would hope that:

Falsehood. *If* $\Gamma \mid \Gamma \vdash \mathcal{C} :: c : C$*, then there exists a process* $\cdot\,;\,\Gamma \vdash P :: c : C$ *such that* $\mathsf{proc}(c, P) \to^* \mathcal{C}$.

This is often impossible when configurations contain message facts. Consider, for example, the configuration $d : A \mid \cdot \vdash \mathsf{msg}(c, \mathsf{send}\ c\ k;\ d \to c) :: c : \oplus\{k : A\}$. The only plausible solutions are variations on the theme $\cdot\,;\, d : A \vdash c.k;\ d \to c :: d : c : \oplus\{k : A\}$. However,

$$\mathsf{proc}(c, c.k;\ d \to c) \to \mathsf{proc}(d', d \to d'), \mathsf{msg}(c, \mathsf{send}\ c\ k;\ d' \to c)$$

and there is no way to get rid of the forwarding process fact. Instead, we settle for:

Proposition 7.5.5. *If* $\Delta \vdash \mathcal{P} :: c : A$*, then there exists a process* $\cdot\,;\,\Delta \vdash P :: c : A$ *such that* $\Delta \vdash \mathcal{P} \doteq_T \mathsf{proc}(a, P) :: c : A$.

Proof. By proposition 5.6.21, $\Delta \vdash \mathcal{P} :: c : A$ has a simply-branched derivation. We proceed by induction on this derivation. We give only the illustrative cases.

Case (Conf-M)**:** We proceed by case analysis on the particular message fact in

$$\frac{\cdot\,;\,\Delta \vdash m :: c : A}{\Sigma \parallel \Delta \mid \cdot \vdash \mathsf{msg}(c, m) :: (c : A)}\ (\textsc{Conf-M})$$

Subcase $m = \mathsf{close}\ c$**:** Take $P = \mathsf{close}\ c$ and apply rule (68) and proposition 7.1.3.
Subcase $m = c.k;\ d \to c$**:** Take $P = m$ and apply rule (78) and proposition 7.1.3 to get:

$$d : A_k \vdash \mathsf{proc}(c, c.k;\ d \to c) \doteq_T \mathsf{proc}(e, d \to e), \mathsf{msg}(c, c.k;\ e \to c) :: c : \oplus\{l : A_l\}_{l \in L}.$$

By proposition 7.1.4,

$$d : A_k \vdash \mathsf{msg}(c, c.k;\ d \to c) \doteq_T \mathsf{proc}(e, d \to e), \mathsf{msg}(c, c.k;\ e \to c) :: c : \oplus\{l : A_l\}_{l \in L}.$$

We conclude the result by transitivity and symmetry.

Case (Conf-P)**:** Immediate.

Case (Conf-C)**:** By assumption, both branches of the rule

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \Gamma_1 \vdash \mathcal{C} :: \Phi\Pi \quad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \Gamma_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \Gamma_1\Pi\Gamma_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\textsc{Conf-C})$$

are simply branched, and $\Pi = b : B$ contains a single channel. By the induction hypothesis, there exist processes C and D such that

$$\Gamma \vdash \mathcal{C} \doteq_T \mathsf{proc}(b, C) :: b : B$$
$$b : B, \Lambda \vdash \mathcal{D} \doteq_T \mathsf{proc}(c, D) :: c : A$$

Take the process $\cdot\,;\,\Gamma\Lambda \vdash b \leftarrow C;\ D :: c : A$. Then

$$\mathsf{proc}(a, b \leftarrow D;\ C) \to \mathsf{proc}(b, C), \mathsf{proc}(c, D),$$

so $\Gamma\Lambda \vdash \mathsf{proc}(a, b \leftarrow C;\ D) \doteq_T \mathsf{proc}(b, C), \mathsf{proc}(c, D) :: c : A$ by proposition 7.1.3. Because $\doteq_T$ is a congruence,

$$\Gamma\Lambda \vdash \mathsf{proc}(a, b \leftarrow C;\ D) \doteq_T \mathcal{C}, \mathcal{D} :: c : A$$

as desired. □

Proposition 7.5.6 extends proposition 7.5.4 to give the correspondence between observation contexts for processes and simply branched configuration contexts.

Proposition 7.5.6.

(1) For all configuration contexts $\Lambda \vdash \mathcal{O}[\cdot]^{\Delta}_{a:A} :: c : C$, there exists an observation context $\cdot\ ; \Lambda \vdash O[\cdot]^{\Delta}_{a:A} :: c : C$ such that for all $\cdot\ ; \Delta \vdash Q :: a : A$,

$$\Lambda \vdash \mathcal{O}[\mathsf{proc}(a, Q)]^{\Delta}_{a:A} \doteq_T \mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) :: c : C.$$

(2) For all observation contexts $\cdot\ ; \Lambda \vdash O[\cdot]^{\Delta}_{a:A} :: c : C$, there exists a configuration context $\Lambda \vdash \mathcal{O}[\cdot]^{\Delta}_{a:A} :: c : C$ such that for all $\cdot\ ; \Delta \vdash Q :: a : A$,

$$\Lambda \vdash \mathcal{O}[\mathsf{proc}(a, Q)]^{\Delta}_{a:A} \doteq_T \mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) :: c : C.$$

Proof. We show the first part of the proposition. Let $\Lambda \vdash \mathcal{O}[\cdot]^{\Delta}_{a:A} :: c : C$ be arbitrary. By proposition 5.6.21, it has a simply-branched derivation. We proceed by induction on this derivation to construct $\cdot\ ; \Lambda \vdash O[\cdot]^{\Delta}_{a:A} :: c : C$. The possible cases are:

Case (Conf-H)**:** If $\mathcal{O}$ is a hole, then let $O = [\cdot]^{\Delta}_{a:A}$.

Case (Conf-C)**:** Then the context is formed by an instance of

$$\frac{\Sigma, \check{\Pi} \parallel \Gamma \mid \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi \qquad \check{\Pi}, \Sigma' \parallel \Pi\Lambda \mid \mathrm{I}_2 \vdash \mathcal{D} :: \Xi}{\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \mid \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi}\ (\text{Conf-C})$$

By simple-branching, $\Pi = b : B$ contains a single channel. If the hole is in the left branch, i.e., if $\mathcal{C} = \mathcal{O}'[\cdot]^{\Delta}_{a:A}$, then by the induction hypothesis, there exists an observation context $\cdot\ ; \Lambda \vdash O'[\cdot]^{\Delta}_{a:A} :: b : B$ such that for all $\cdot\ ; \Delta \vdash Q :: a : A$,

$$\Lambda \vdash \mathcal{O}'[\mathsf{proc}(a, Q)]^{\Delta}_{a:A} \doteq_T \mathsf{proc}(c, O'[Q]^{\Delta}_{a:A}) :: b : B.$$

Let D be given for $\mathcal{D}$ by proposition 7.5.5 such that $b : B, \Lambda \vdash \mathcal{D} \doteq_T \mathsf{proc}(c, D) :: c : A$. Take $\cdot\ ; \Lambda \vdash O[\cdot]^{\Delta}_{a:A} :: c : C$ to be given by $b \leftarrow O'[\cdot]^{\Delta}_{a:A};\ D$. Then for all $\cdot\ ; \Delta \vdash Q :: a : A$,

$$\mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) \to \mathsf{proc}(b, O'[Q]^{\Delta}_{a:A}), \mathsf{proc}(c, D),$$

so by proposition 7.1.3,

$$\Lambda \vdash \mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) \doteq_T \mathsf{proc}(b, O'[Q]^{\Delta}_{a:A}), \mathsf{proc}(c, D) :: c : C.$$

But $\doteq_T$ is a congruence, so

$$\Lambda \vdash \mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) \doteq_T \mathcal{O}'[\mathsf{proc}(a, Q)]^{\Delta}_{a:A}, \mathcal{D} :: c : C,$$

i.e., $\Lambda \vdash \mathsf{proc}(c, O[Q]^{\Delta}_{a:A}) \doteq_T \mathcal{O}[\mathsf{proc}(a, Q)]^{\Delta}_{a:A} :: c : C$. The result follows by symmetry. The case for when the hole is in the right branch is analogous.

The second part of the proposition is immediate by propositions 7.1.3 and 7.5.4. □

7.5.2. Relating Precongruences on Configurations and Processes. We show how to lift relations $\preccurlyeq$ on configurations to relations on open processes. We frequently assume that $\doteq_T \subseteq \preccurlyeq$. Recall from fig. 7.1 that this assumption is satisfied by all observational preorders $\preccurlyeq$ that we have considered thus far.

Definition 7.5.7. A **closing substitution** is a substitution (a context morphism) $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Psi$ such that $\sigma(x)$ val for all $x : \tau \in \Psi$. ◂

We lift type-indexed relations on configurations to type-indexed relations on open processes using an approach reminiscent of Howe's "open extensions" [How96, Definition 2.2]:

Definition 7.5.8. Let $\mathfrak{R}$ be a type-indexed relation on configurations. Write $\Psi\ ;\ \Delta \vdash P\ \lceil\mathfrak{R}\rceil\ Q :: c : C$ if $\Delta \vdash \mathsf{proc}(c, [\sigma]P)\ \mathfrak{R}\ \mathsf{proc}(c, [\sigma]Q) :: c : C$ for all closing substitutions $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Psi$. ◀

Recall the definition of simply branched contextual interior $\mathfrak{R}^b$ of a relation from definition 5.7.11. The simply branched contextual interior and observationally contextual interiors coincide:

Proposition 7.5.9. *Let $\preccurlyeq$ be a transitive type-indexed relation on configurations such that $\doteq_T \subseteq \preccurlyeq$. The following are equivalent:*

(1) $\Psi\ ;\ \Delta \vdash P\ \left\lceil\preccurlyeq^b\right\rceil\ Q :: c : C$;

(2) $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{\mathcal{O}}\ Q :: c : C$.

Proof. Remark that, because $\doteq_T$ is a congruence, $\doteq_T \subseteq \preccurlyeq$ implies $\doteq_T \subseteq \preccurlyeq^b$. Observe that $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{\mathcal{O}}\ Q :: c : C$ if and only if both

(i) $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil\ Q :: c : C$; and

(ii) $\cdot\ ;\ \Gamma \vdash O[[\sigma]P]^{\Delta}_{c:C}\ \lceil\preccurlyeq\rceil\ O[[\sigma]Q]^{\Delta}_{c:C} :: b : B$ for all closing substitutions $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Psi$ and all observation contexts $\cdot\ ;\ \Gamma \vdash O[\cdot]^{\Delta}_{c:C} :: b : B$.

To see that item 1 implies item 2, assume that $\Psi\ ;\ \Delta \vdash P\ \left\lceil\preccurlyeq^b\right\rceil\ Q :: c : C$. This implies that $\cdot\ ;\ \Delta \vdash [\sigma]P\ \left\lceil\preccurlyeq^b\right\rceil\ [\sigma]Q :: c : C$ for all $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Psi$. Let $\cdot\ ;\ \Gamma \vdash O[\cdot]^{\Delta}_{c:C} :: b : B$ be an arbitrary observation context, and let $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Psi$ be an arbitrary closing substitution. We must show that

$$\cdot\ ;\ \Gamma \vdash O[[\sigma]P]^{\Delta}_{c:C}\ \lceil\preccurlyeq\rceil\ O[[\sigma]Q]^{\Delta}_{c:C} :: b : B. \tag{102}$$

By proposition 7.5.6, there exists a simply branched context $\Gamma \vdash \mathcal{O}[\cdot]^{\Delta}_{c:C} :: b : B$ such that

$$\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]P)]^{\Delta}_{c:C} \doteq_T \mathsf{proc}(b, O[[\sigma]P]^{\Delta}_{c:C}) :: b : B,$$
$$\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]Q)]^{\Delta}_{c:C} \doteq_T \mathsf{proc}(b, O[[\sigma]Q]^{\Delta}_{c:C}) :: b : B.$$

Because $\Psi\ ;\ \Delta \vdash P\ \left\lceil\preccurlyeq^b\right\rceil\ Q :: c : C$ and $\preccurlyeq^b$ is simply branched contextual,

$$\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]P)]^{\Delta}_{c:C} \preccurlyeq^b \mathcal{O}[\mathsf{proc}(c, [\sigma]Q)]^{\Delta}_{c:C} :: b : B.$$

By assumption, the symmetric relation $\doteq_T$ is contained in $\preccurlyeq^b$. By transitivity of $\preccurlyeq^b$,

$$\Gamma \vdash \mathsf{proc}(b, O[[\sigma]P]^{\Delta}_{c:C}) \preccurlyeq^b \mathsf{proc}(b, O[[\sigma]Q]^{\Delta}_{c:C}) :: b : B.$$

But $\preccurlyeq^b \subseteq \preccurlyeq$, so we conclude eq. (102).

To see that item 2 implies item 1, assume that $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{\mathcal{O}}\ Q :: c : C$, and let $\Gamma \vdash \mathcal{O}[\cdot]^{\Delta}_{c:C} :: b : B$ and $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Psi$ be arbitrary. We must show that $\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]P)]^{\Delta}_{c:C} \preccurlyeq \mathcal{O}[\mathsf{proc}(c, [\sigma]Q)]^{\Delta}_{c:C} :: b : B$. By proposition 7.5.6, there exists an observation context $\Gamma \vdash O[\cdot]^{\Delta}_{c:C} :: b : B$ such that

$$\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]P)]^{\Delta}_{c:C} \doteq_T \mathsf{proc}(b, O[[\sigma]P]^{\Delta}_{c:C}) :: b : B,$$
$$\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]Q)]^{\Delta}_{c:C} \doteq_T \mathsf{proc}(b, O[[\sigma]Q]^{\Delta}_{c:C}) :: b : B.$$

The symmetric relation $\doteq_T$ is contained in $\preccurlyeq$, so

$$\Gamma \vdash \mathcal{O}[\mathsf{proc}(c, [\sigma]P)]^{\Delta}_{c:C} \preccurlyeq \mathsf{proc}(b, O[[\sigma]P]^{\Delta}_{c:C}) :: b : B,$$
$$\Gamma \vdash \mathsf{proc}(b, O[[\sigma]Q]^{\Delta}_{c:C}) \preccurlyeq \mathcal{O}[\mathsf{proc}(c, [\sigma]Q)]^{\Delta}_{c:C} :: b : B.$$

By assumption, $\Gamma \vdash \mathsf{proc}(b, O[[\sigma]P]^{\Delta}_{c:C}) \preccurlyeq \mathsf{proc}(b, O[[\sigma]Q]^{\Delta}_{c:C}) :: b : B$. We are done by transitivity of $\preccurlyeq$. □

We would like to strengthen the correspondence to give a full precongruence:

Conjecture 7.5.10. *Let $\preccurlyeq$ be a transitive type-indexed relation on configurations such that $\doteq_T \subseteq \preccurlyeq$. Then $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{\mathcal{O}}\ Q :: a : A$ if and only if $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{c}\ Q :: a : A$.*

A proof of conjecture 7.5.10 is elusive because of the subtle interplay between the process and functional layers. We have made preliminary attempts to generalize Howe's method [How96] to prove this result, but do not present these attempts here. This generalization is non-trivial because a single relation on processes is insufficient: we also need a relation on terms. We must show that these two relations agree with each other, and that all constructions in Howe's method preserve this agreement.

Despite these difficulties, we can still significantly generalize proposition 7.5.9 to handle contexts whose hole does not cross the boundary between processes and functional programs. We call these contexts "pure process contexts":

Definition 7.5.11. A **pure process context** $\Psi\ ;\ \Delta \vdash C_p[\cdot]^{\Gamma;\Lambda}_{b:B} :: a : A$ is a process context with exactly one hole such that its instance of (P-Hole) does not appear in a subderivation of (E-{}). ◂

Definition 7.5.12. A typed-indexed relation $\mathfrak{R}$ on processes is an **purely process contextual** if $\Psi\ ;\ \Delta \vdash P\ \mathfrak{R}\ Q :: a : A$ implies $\Phi\ ;\ \Lambda \vdash C_p[P]^{\Psi;\Delta}_{a:A}\ \mathfrak{R}\ C_p[Q]^{\Delta}_{a:A} :: b : B$ for all pure process contexts $\Phi\ ;\ \Lambda \vdash C_p[\cdot]^{\Psi;\Delta}_{a:A} :: b : B$. The **purely process contextual interior** of a typed relation $\mathfrak{R}$ on processes is the greatest purely process contextual typed relation $\mathfrak{R}^p$ contained in $\mathfrak{R}$. ◂

Theorem 7.5.13. *Let $\preccurlyeq$ be a transitive type-indexed relation on configurations such that $\doteq_T \subseteq \preccurlyeq$. Then $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{\mathcal{O}}\ Q :: a : A$ if and only if $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{p}\ Q :: a : A$.*

Proof. Necessity is immediate, so we show sufficiency. Assume that $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq\rceil^{\mathcal{O}}\ Q :: c : C$. By proposition 7.5.9, this implies for all simply branched contexts $\Lambda \vdash \mathcal{B}[\cdot]^{\Delta}_{c:C} :: b : B$ and all closing substitutions $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Phi$ that

$$\Lambda \vdash \mathcal{B}[\mathsf{proc}(c, [\sigma]P)]^{\Delta}_{c:C} \preccurlyeq \mathcal{B}[\mathsf{proc}(c, [\sigma]Q)]^{\Delta}_{c:C} :: b : B.$$

We show the stronger property that $\Psi\ ;\ \Delta \vdash P\ \lceil\preccurlyeq^b\rceil^{p}\ Q :: a : A$. This means that we must show for all pure process contexts $\Gamma\ ;\ \Phi \vdash C[\cdot]^{\Psi;\Delta}_{a:A} :: b : B$ and all $\sigma :_{\mathsf{f}} \cdot \rightsquigarrow \Gamma$ that

$$\Phi \vdash \mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big)) \preccurlyeq^b \mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big)) :: b : B.$$

This in turn requires that we show for all simply branched contexts $\Lambda \vdash \mathcal{B}[\cdot]^{\Phi}_{b:B} :: d : D$ that:

$$\Lambda \vdash \mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \preccurlyeq \mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} :: d : D$$

Let $\Lambda \vdash \mathcal{B}[\cdot]^{\Phi}_{b:B} :: d : D$ be an arbitrary simply branched context. We proceed by induction on $\cdot\ ;\ \Phi \vdash C[\cdot]^{\Psi;\Delta}_{a:A} :: b : B$, and give only the illustrative cases.

Case (P-Hole)**:** The result is immediate by assumption.

Case (Fwd$^+$)**:** This case is impossible because there is no hole.

Case (Cut)**:** Then C is either $e \leftarrow C'[\cdot]^{\Psi;\Delta}_{a:A};\ R$ or $e \leftarrow L;\ C'[\cdot]^{\Psi;\Delta}_{a:A}$ for some C' and L or R. Assume that we fall in the first case. Let e' be globally fresh. Then by rule (66),

$$\begin{aligned}&\mathcal{B}[\mathsf{proc}(b, [\sigma]C[P]^{\Psi;\Delta}_{a:A})]^{\Phi}_{b:B} \rightarrow \\ &\quad\rightarrow \mathcal{B}[\mathsf{proc}(e', [e'/e]\left([\sigma]\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\right)), \mathsf{proc}(b, [e'/e]\left([\sigma]R\right))]^{\Phi}_{b:B}.\end{aligned}$$

By proposition 7.1.3,

$$\begin{aligned}&\Lambda \vdash \mathcal{B}[\mathsf{proc}(b, [\sigma]C[P]^{\Psi;\Delta}_{a:A})]^{\Phi}_{b:B} \\ &\quad\doteq_T \mathcal{B}[\mathsf{proc}(e', [e'/e]\left([\sigma]\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\right)), \mathsf{proc}(b, [e'/e]\left([\sigma]R\right))]^{\Phi}_{b:B} :: d : D.\end{aligned}$$

We recognize the right side as $\mathcal{B}'[\mathsf{proc}(e', [e'/e](C'[P]^{\Delta}_{a:A}))]$ where $\mathcal{B}'$ is the simply branched context $\mathcal{B}[[\cdot]^{\Delta}_{a:A}, \mathsf{proc}(b, [e'/e]R)]^{\Phi}_{b:B}$. Analogously,

$$\Lambda \vdash \mathcal{B}[\mathsf{proc}(b, [\sigma]C[Q]^{\Psi;\Delta}_{a:A})]^{\Phi}_{b:B} \doteq_T \mathcal{B}'[\mathsf{proc}(e', [e'/e]\left([\sigma]\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\right))] :: d : D.$$

By the induction hypothesis and the fact that $\preccurlyeq$ is type-indexed, so closed under renamings of channels,

$$\Lambda \vdash \mathcal{B}'[\mathsf{proc}(e', [e'/e]\left([\sigma]\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\right))] \preccurlyeq \mathcal{B}'[\mathsf{proc}(e', [e'/e]\left([\sigma]\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\right))] :: d : D.$$

We are done by transitivity, the assumption that $\doteq_T \subseteq \preccurlyeq$, and symmetry of $\doteq_T$. The case of $e \leftarrow L;\ C'[\cdot]^{\Psi;\Delta}_{a:A}$ for some C' and L is analogous.

Case ($\oplus$L)**:** Then C is of the form $\mathsf{case}\ e\ \{l \Rightarrow P_l\}_{l \in L}$ where $P_k = C'[\cdot]^{\Psi;\Delta}_{a:A}$ for some unique $k \in L$. We observe that

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{msg}(e, e.l;\ e' \to e), \mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi'}_{b:B}$$

if and only if

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{msg}(e, e.l;\ e' \to e), \mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi'}_{b:B}.$$

If this is the case and $l \neq k$, then both

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,([\sigma]P_l))]^{\Phi'}_{b:B},$$
$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,([\sigma]P_l))]^{\Phi'}_{b:B}.$$

We are done by proposition 7.1.3, transitivity, and the inclusion $\doteq_T \subseteq \preccurlyeq$. If $l = k$, then

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma]\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B},$$
$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma]\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B}.$$

By the induction hypothesis and the fact that $\preccurlyeq$ is type-indexed, so closed under renamings of channels,

$$\Lambda \vdash \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma]\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B} \preccurlyeq \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma]\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B} :: d : D.$$

We are done by proposition 7.1.3, transitivity, the assumption that $\doteq_T \subseteq \preccurlyeq$, and symmetry of $\doteq_T$.

Finally, assume that in no fair trace do we get a message fact $\mathsf{msg}(e, e.l;\ e' \to e)$. Then by case analysis on the rules, no rule ever applies to $\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))$ or $\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))$. It follows that $\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B}$ and $\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B}$ have the same traces (modulo the unused process fact), so the same observable messages and observed communications. This completes the case.

Case ($\wedge$L)**:** Then C is of the form $x \leftarrow \mathsf{input}\ e;\ C'[\cdot]^{\Psi;\Delta}_{a:A}$. We observe that

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{msg}(e, \_ \leftarrow \mathsf{output}\ e\ v;\ e' \to e), \mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi'}_{b:B}$$

if and only if

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{msg}(e, \_ \leftarrow \mathsf{output}\ e\ v;\ e' \to e), \mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi'}_{b:B}.$$

If this is the case, then both

$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[P]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{proc}(b, [e', v/e, x]\,\big([\sigma]\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B}, \tag{103}$$
$$\mathcal{B}[\mathsf{proc}(b, [\sigma]\big(C[Q]^{\Psi;\Delta}_{a:A}\big))]^{\Phi}_{b:B} \to \mathcal{B}'[\mathsf{proc}(b, [e', v/e, x]\,\big([\sigma]\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B}. \tag{104}$$

We remark that the composition $[v/x] \circ \sigma$ determines a closing substitution $\sigma' :_{\mathsf{f}} \cdot \rightsquigarrow \Gamma, x : \tau$ for $C'[\cdot]^{\Psi;\Delta}_{a:A}$. So the right sides of (103) and (104) are respectively equal to:

$$\mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma']\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B},$$
$$\mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma']\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B}.$$

By the induction hypothesis and the fact that $\preccurlyeq$ is type-indexed, so closed under renamings of channels,

$$\Lambda \vdash \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma']\big(C'[P]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B} \preccurlyeq \mathcal{B}'[\mathsf{proc}(b, [e'/e]\,\big([\sigma']\big(C'[Q]^{\Psi;\Delta}_{a:A}\big)\big))]^{\Phi'}_{b:B} :: d : D.$$

We are done by proposition 7.1.3, transitivity, the assumption that $\doteq_T \subseteq \preccurlyeq$, and symmetry of $\doteq_T$. Finally, assume that in no fair trace do we get a message fact $\mathsf{msg}(e, \_ \leftarrow \mathsf{output}\ e\ v;\ d \to e)$. Then the analysis is the same as in the previous case.

All other cases are analogous to the above. Explicitly, (FWD$^-$), (**1**R), and (E-{}) are analogous to (FWD$^+$). All of the cases in which the hole sends a message are analogous to (CUT). All of the cases in which the hole receives a message are analogous to ($\oplus$L) or ($\wedge$L), depending on whether or not the message carries a functional value. □

We summarize our results for the precongruences of fig. 7.1:

COROLLARY 7.5.14. *Let $\preccurlyeq$ be a transitive, type-indexed precongruence on configurations such that $\doteq_T \subseteq \preccurlyeq$. The following are equivalent:*

(1) $\Psi\,;\Delta \vdash P \lceil\preccurlyeq\rceil Q :: a : A$*;*
(2) $\Psi\,;\Delta \vdash P \lceil\preccurlyeq\rceil^{\mathcal{O}} Q :: b : B$*;*
(3) $\Psi\,;\Delta \vdash P \lceil\preccurlyeq\rceil^{p} Q :: b : B$*.*

Proof. Observe that $\left((\preccurlyeq)^c\right)^b = \preccurlyeq^c$. The result follows from proposition 7.5.9 and theorem 7.5.13. □

CHAPTER 8

Denotational Approaches to Equivalence

From the outset, denotational semantics are a promising approach for reasoning about Polarized SILL and its programs. Indeed, denotational semantics are *compositional* by construction. This means that we can reason about parts of a program at a time, instead of having to reason about whole programs at once. They also induce a semantic equivalence, and as described in chapter 1, program equivalences underlie many techniques for reasoning about programs. Moreover, Polarized SILL has a functional layer and recursive types and programs, and denotational semantics have historically excelled at reasoning about these features in a variety of settings. Finally, a recurring observation in programming languages research is that beautiful and elegant techniques work best, and it is our opinion that denotational semantics are a mathematically elegant approach to programming languages semantics.

There are several challenges in giving Polarized SILL a denotational semantics. We illustrate these using the bit flipping process `flip` from example 5.3.10. Fair executions ensure that processes have deterministic input-output behaviour. This suggests that `flip` denotes a function $[\![\texttt{flip}]\!]$ from bit streams on `b` to bit streams on `f`. This processes-as-functions interpretation raises many questions. The process providing the bit stream on b could get stuck and only send a finite prefix of this bit stream. How should $[\![\texttt{flip}]\!]$ handle these finite prefixes? Computationally, $[\![\texttt{flip}]\!]$ should be monotone: a longer input prefix should result in no less output. It should also be continuous: $[\![\texttt{flip}]\!]$ should not be able to observe an entire infinitely-long bit stream before sending output. This suggests that $[\![\texttt{flip}]\!]$ denotes a continuous function between dcpos of bit streams. These questions and answers have been known for close to fifty years: Kahn [Kah74] answered them when giving a semantics to dataflow networks.

Real challenges arise when we realize that bit streams and `flip` are not representative of many SILL protocols and processes: they do not involve bidirectional communication. The questions are then: if monotonicity and continuity capture important computational properties, can we use still continuous functions to model processes with bidirectional communications? If so, what should be the functions' domains and codomains? A natural idea is to decompose bidirectional session-typed communications into pairs of unidirectional communications, and then to define functions on these decomposed communications. But how do we decompose bidirectional communications in a principled way, so that we do not lose any information? And how do we ensure that the denotations of processes respect this decomposition, i.e., that they do not produce output that, according to the decomposition, is inconsistent with their input? Finally, what does it mean to compose communicating processes in this setting?

We show that a domain-theoretic denotational semantics elucidates the structure of higher-order session-typed languages with recursion. We make the following contributions:

(1) **A new style of denotational semantics called *CYO semantics*.** CYO semantics are a general denotational framework for processes and bidirectional communication. In CYO semantics, communication protocols denote decompositions of bidirectional communications into unidirectional communications. Processes denote continuous functions from unidirectional communications (their inputs) to completed bidirectional communications. The semantic framework is designed such that processes and communication decompositions form a coherent whole. We give an overview of CYO semantics in section 8.1, and we give the details in section 8.2.

(2) **An order-theoretic analysis of polarized session types.** The decomposition of bidirectional communication into "inputs" and "outputs" is linked to polarity. We show that this decomposition is given by a natural family of embeddings.
(3) **A denotational semantics for Polarized SILL.** We give Polarized SILL a CYO semantics in section 8.3. We interpret session types as dI-domains, and processes and terms as stable functions. These interpretations validate expected equivalences. We show that the semantics is well-defined in section 8.4, that it satisfies expected semantic properties in section 8.5, and that it is sound in section 8.6.

We hope this work will help bridge the gap between two research communities and bodies of work. For readers familiar with session types, we hope they can take away the high-level ideas of their semantic interpretation in the presence of (nonlinear) functions and arbitrary recursion and how it might be used to reason about process equivalence. A particular phenomenon not usually addressed is that processes may fail to communicate along a given channel in the presence of recursively defined types and processes. This phenomenon is easily addressed domain-theoretically: because processes denote continuous (so monotone) functions, they uniformly treat complete and incomplete communications.

For readers familiar with denotational semantics, we hope they can take away the ideas behind its application to bidirectional, session-typed communication in the presence of recursion. The key insights here, when compared to the denotational semantics of functional languages, are that (session) types denote decompositions of complete communications into pairs of unidirectional communications instead of denoting domains of values, and that program (process) composition is given by a trace operator instead of by function composition.

8.1. Overview of the Semantics

We first give an overview of our semantics for processes. We do so through a sequence of false starts, where each successive attempt will capture an essential feature of our semantics. Then, we give an overview of our semantics for the functional layer.

Our starting point is Kahn's semantics [Kah74] for dataflow networks. In dataflow networks, processes are computational agents that communicated over *unidirectional* channels. These channels carry sequences (streams) of values, e.g., natural numbers. It is assumed that these channels are the only means processes have to communicate. It is also assumed that if a message is sent, then it is transmitted within an unpredictable but finite amount of time. In Kahn's semantics, communication channels denote dcpos of prefix-ordered sequences of values. Processes denote continuous functions on the dcpos of input channels to the dcpos of output channels. Kahn used a least fixed point construction to capture process composition.

This approach guarantees several desirable semantic properties. First, processes are monotone: giving a process more input will result in no less output. Second, continuity ensures that processes cannot wait until they have received all of their input before they start computing.

In contrast to processes in dataflow networks, session-typed processes communicate on *bidirectional* channels. At first glance, this poses no difficulty: we can imagine each bidirectional channel as being a pair of bidirectional channels, with one channel for each direction. Using the terminology of section 5.1, one of the channels carries communications in the *positive* direction, while the other carries communications in the *negative* direction. If we write $[\![A]\!]^+$ and $[\![A]\!]^-$ for the pointed[1] dcpos of communications that respectively flow in the positive and negative direction on a channel of type A, a process $\Psi \,;\, a_1 : A_1, \ldots, a_n : A_n \vdash P :: a_o : A_o$ denotes a continuous function[2]

$$[\![\Psi \,;\, a_1 : A_1, \ldots, a_n : A_n \vdash P :: a_o : A_o]\!] : \left(\prod_{i=1}^{n} [\![A_i]\!]^+\right) \times [\![A_o]\!]^- \to \left(\prod_{i=1}^{n} [\![A_i]\!]^-\right) \times [\![A_o]\!]^+. \quad (105)$$

[1] The bottom element represents the absence of communication.
[2] We ignore the presence of Ψ and of the functional layer for the time being.

In this setting, process composition is exactly as it was in Kahn's semantics. Indeed, we can interpret (Cut) as:

$$[\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!](\delta_1^+, \delta_2^+, c^-) = (\delta_1^-, \delta_2^-, c^+)$$

where δ_1^-, δ_2^-, a^-, a^+, and c^+ form the least solution[3] to the equations

$$(\delta_1^-, a^+) = [\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!](\delta_1^+, a^-),$$
$$(\delta_2^-, a^-, c^+) = [\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!](\delta_2^+, a^+, c^-).$$

We recognize this fixed point as the trace of the interpretations of P and Q:

$$[\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]$$
$$= \mathrm{Tr}^{a^+ \times a^-}\left([\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!] \times [\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]\right).$$

Informally, the trace operator $\mathrm{Tr}^{a^+ \times a^-}$ fixes and then hides the internal communications between P and Q on $[\![a : A]\!]^+ \times [\![a : A]\!]^-$.

We remark that the process interpretation (105) exists within a "wave"-style [Abr96] geometry of interaction (GoI) construction [AHS02, Definition 2.6]. Indeed, the objects of the GoI construction $\mathcal{G}(\mathbf{DCPO}_\perp)$ are pairs (A^+, A^-) of objects A^+ and A^- of $\mathbf{DCPO}_\perp$. Morphisms $f : (A^+, A^-) \to (B^+, B^-)$ of $\mathcal{G}(\mathbf{DCPO}_\perp)$ are morphisms $\hat{f} : A^+ \times B^- \to A^- \times B^+$ of $\mathbf{DCPO}_\perp$. Given a morphism $g : (B^+, B^-) \to (C^+, C^-)$, the composition $g \circ f$ is defined by $\mathrm{Tr}^{B^- \times B^+}_{A^+ \times C^-, A^- \times C^+}(\hat{g} \times \hat{f})$. The interpretation of process composition is then exactly the composition $[\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]u \circ [\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!]u$ in $\mathcal{G}(\mathbf{DCPO}_\perp)$.

Though this approach seems promising and intuitive reasonable, it raises several questions. We address these in turn.

Question 8.1.1. What does it mean to decompose communications satisfying A into their positive and negative "aspects", i.e., into dcpos $[\![A]\!]^+$ and $[\![A]\!]^-$, and to do so in a principled way? ◂

To answer question 8.1.1, we define a third pointed dcpo, $[\![A]\!]$, of bidirectional communications satisfying A. Informally, we treat this dcpo as the ground truth of what it means to be a communication satisfying A. A decomposition of A into its polarized aspects is then given by a (continuous) embedding $\langle A \rangle : [\![A]\!] \to [\![A]\!]^+ \times [\![A]\!]^-$. This embedding ensures that there exists a faithful copy of the bidirectional communications $[\![A]\!]$ in the dcpo of decomposed communications $[\![A]\!]^+ \times [\![A]\!]^-$. Its projection associates to each $(a^+, a^-) \in [\![A]\!]^+ \times [\![A]\!]^-$ the largest bidirectional communication $a \in [\![A]\!]$ whose decomposition $\langle A \rangle(a)$ is consistent with (a^+, a^-).

To help build intuition for this semantics of processes and communication decompositions, we make an analogy between communications and interactive surveys. Interactive surveys are questionnaires that may, based on an answer to a given question, instruct you to skip certain questions. Imagine that a session type A specifies an interactive survey, and let $[\![A]\!]$ be the dcpo of partially or fully completed surveys under a prefix-ordering. The embedding $\langle A \rangle : [\![A]\!] \to [\![A]\!]^+ \times [\![A]\!]^-$ decomposes surveys $a \in [\![A]\!]$ into pairs (a^+, a^-), where a^- is the collection of questions answered and a^+ is the collection of answers.[4] Consider a process $\Psi \,;\, \cdot \vdash P :: s : A$ that completes a survey over the channel s. It denotes a continuous function $[\![A]\!]^- \to [\![A]\!]^+$ from sequences of survey questions to survey answers.

Question 8.1.2. Consider a sequence of survey questions a^-, and set $a^+ = [\![\Psi \,;\, \cdot \vdash P :: s : A]\!](a^-)$. How do we semantically ensure that P's answers a^+ correspond to the questions a^- it received? ◂

We cannot insist that $\langle A \rangle(a) = (a^+, a^-)$ for some a, for the process P may not have answered all the questions. Indeed, the process P could have gotten stuck in an infinite loop and only consumed part of its input a^-. However, it is semantically reasonable to require that there be a least

[3] By the Kleene fixed-point characterization of traces, corollary 2.3.3, we can think of this least solution as the limit of a sequence of finite approximations, where each approximation represents one additional exchange between processes.

[4] The choice to treat questions as negative and answers as positive was arbitrary. The symmetric choice is equally valid.

$a_o^- \sqsubseteq a^-$ such that $[\![\Psi \,;\, \cdot \vdash P :: s : A]\!](a_o^-) = a^+$: this a_o^- corresponds to the prefix of the questions a^- that P actually answered. Moreover, it is semantically reasonable to require that there exist an $a_o \in [\![A]\!]$ such that $\langle A \rangle(a_o) = (a^+, a_o^-)$: it is the partially completed survey consisting of the questions a_o^- that P answered, along with P's answers a^+.

Our answer to question 8.1.2 comes in two parts: first, a change to the semantics, and second, a property we call *junk-freedom*.

We start by revising our semantics so that processes denote continuous functions from partial communications to bidirectional communications. Explicitly, a process $\Psi \,;\, a_1 : A_1, \ldots, a_n : A_n \vdash P :: a_o : A_o$ now denotes a continuous function of type

$$[\![\Psi \,;\, a_1 : A_1, \ldots, a_n : A_n \vdash P :: a_o : A_o]\!] : \left(\prod_{i=1}^{n} [\![A_i]\!]^+\right) \times [\![A_o]\!]^- \to \prod_{i=o}^{n} [\![A_i]\!]. \tag{106}$$

We can recover our previous semantics, (105), by composing this new denotation (106) with the appropriate embeddings and projections. As before, process composition is defined using a trace operator, i.e., it is given by a least fixed point.

Next, we require that processes denote *junk-free* functions. To make this rigorous, set $\langle A \rangle^- = \pi_2 \circ \langle A \rangle : [\![A]\!] \to [\![A]\!]^-$ and $p = [\![\Psi \,;\, \cdot \vdash P :: s : A]\!] : [\![A]\!]^- \to [\![A]\!]$. Write $f \restriction A'$ for the restriction of f a function $f : A \to B$ to a subset $A' \subseteq A$, and write $f^\circ : A \to \mathrm{im}(f)$ for the corestriction of f to its image. We say that a function $p : [\![A]\!]^- \to [\![A]\!]$ is junk-free (relative to $\langle A \rangle$) if $(\langle A \rangle^- \restriction \mathrm{im}(p), p^\circ)$ is an e-p-pair of monotone maps.[5] Junk-freedom captures several desirable semantic facts:

(1) Bidirectional communications in the image of p agree with the input p used to generate them, or alternatively, the questions in a survey completed by p are a prefix of the ones p received as input. Indeed, if a^- is a sequence of questions, then the questions completed by p are $a_o^- = (\langle A \rangle^- \circ p)(a^-)$,[6] and the definition of e-p-pair ensures that $a_o^- \sqsubseteq a^-$.

(2) Bidirectional communications in the image of p are uniquely determined by a minimal piece of input. This property follows from the fact that projection preserve existing infima by proposition 2.2.19 and that, as we will see, dcpos of communications will be bounded complete.

By using functions of type (106), we can also easily state another desirable semantic property: *completeness*. Completeness means that if $p(a^-) = a$ and $\langle A \rangle^+(a) = a^+$, then $\langle A \rangle^p(a^+, a^-) = a$. Intuitively, this means that a contains the first question that p left unanswered, if it exists. In particular, it means that if p answers no questions, a contains the first question in a^-. This lets us differentiate between settings where p could have answered a question had it been presented with one, and settings where p could not have answered such a question.

Before going any further in our analysis of denotations of processes, we must investigate the dcpos on which they are defined, i.e., the denotations of session-types:

Question 8.1.3. Which variety of dcpo best reflects semantic properties of session-typed communications? ◂

We claim that pointed dI-domains (definition 2.2.35) are an ideal choice for our semantics. First, the interpretation of (CVar) forces us to use bounded-complete domains.[7] Second, we believe it important for domains of communications to satisfy the I-property. Indeed, a compact communication, which we can intuit as a finite prefix of a communication, should be approximated by only finitely many other compact communications. Third, if we retain our intuition from section 6.1 that session-typed communications are trees of messages, then elements in domains of communications should satisfy the d-property. To illustrate this fact, consider complete communications x, y, z such that $y \uparrow z$, i.e., such that y and z are consistent. Under the prefix ordering, the infimum of trees is given by their intersection (their largest common prefix), while their supremum is given by

[5] We do not require that they form an e-p-pair of continuous morphisms. We also remark that, though the image $\mathrm{im}(f)$ of f need not be a dcpo, it is a poset.

[6] Observe also that $p(a_o^-) = p(a^-)$ by proposition 2.2.19.

[7] We refer the reader to section 8.3.5 for a discussion of this fact.

their union (the least tree of which they are all a prefix). The assumption $y \uparrow z$ implies that y and z are both prefixes of some larger tree, and by bounded-completeness, their union $y \sqcup z$ exists. The tree $x \sqcap (y \sqcup z)$ is then the largest prefix of both x and $y \sqcup z$. This prefix is given by the union of the largest prefixes $x \sqcap y$ of x and y and $x \sqcap z$ of x and z. That is, $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. Finally, we require our domains to be pointed to allow for empty communications and to ensure the existence of least fixed points.

Question 8.1.4. Which semantic universe or categorical structures best capture Polarized SILL's processes and configurations? ◂

Processes compose in a tree-like structure. Semantically, we expect process composition to be associative and partially commutative. To make these facts semantically explicit, we interpret processes as morphisms in a multicategory. This multicategory is contained in a *CYO pluricategory*[8,9] $\mathbf{CYO}(\mathbf{Stab}_\perp)$ over the category $\mathbf{Stab}_\perp$ of pointed dI-domains and stable maps. The objects of $\mathbf{CYO}(\mathbf{Stab}_\perp)$ are embeddings $a : A \to A^+ \times A^-$ between dI-domains. These embeddings are subject to an additional condition—being "well-woven"—that is necessary and sufficient for it to have an identity morphism in $\mathbf{CYO}(\mathbf{Stab}_\perp)$. Given objects $a_i : A_i \to A_i^+ \times A_i^-$ and $b_j : B_j \to B_j^+ \times B_j^-$, a morphism $a_1, \ldots, a_n \to b_1, \ldots, b_m$ in $\mathbf{CYO}(\mathbf{Stab}_\perp)$ is a function

$$\left(\prod_{i=1}^{n} A_i^+\right) \times \left(\prod_{j=1}^{m} B_j^-\right) \to \left(\prod_{i=1}^{n} A_i\right) \times \left(\prod_{j=1}^{m} B_j\right) \tag{107}$$

in $\mathbf{Stab}_\perp$ closed under composition with the identity morphisms of $\mathbf{CYO}(\mathbf{Stab}_\perp)$. Composition is given by a trace operator. Processes $\Psi \,;\, a_1 : A_1, \ldots, a_n : A_n \vdash P :: a_o : A_o$ now denote junk-free, complete, frugal[10] morphisms

$$[\![\Psi \,;\, a_1 : A_1, \ldots, a_n : A_n \vdash P :: a_o : A_o]\!] : \langle A_1 \rangle, \ldots, \langle A_n \rangle \to \langle A_o \rangle \tag{108}$$

in $\mathbf{CYO}(\mathbf{Stab}_\perp)$, where $\langle A_i \rangle : [\![A_i]\!] \to [\![A_i]\!]^+ \times [\![A_i]\!]^-$ is the denotation of the closed session type A_i. In general, we write $\langle a_1 : A_1, \ldots, a_n : A_n \rangle$ for the object $\langle A_1 \rangle, \ldots, \langle A_n \rangle$.

Until this point, we have only considered the denotations of closed session types. To be able to define the semantics of recursive session types, we must also give a semantic account of open session types $\Xi \vdash A\ \mathsf{type_s}$. To do so, we generalize from a single embedding to a natural family of well-woven embeddings[11]

$$\langle \Xi \vdash A\ \mathsf{type_s} \rangle : [\![\Xi \vdash A\ \mathsf{type_s}]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type_s}]\!]^+ \times [\![\Xi \vdash A\ \mathsf{type_s}]\!]^- : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!},$$

where $[\![\Xi]\!] = \prod_{\alpha \in \Xi} \mathbf{Stab}_{\perp!}$. We abuse notation and write $\langle \Xi \vdash A\ \mathsf{type_s} \rangle^p$ for the corresponding family of projections. This family will not, in general, be natural. The family $\langle \Xi \vdash A\ \mathsf{type_s} \rangle$ determines a 2-cell in the 2-category **CFP** defined in section 4.5.2. In particular, the functors $[\![\Xi \vdash A\ \mathsf{type_s}]\!]$ (giving bidirectional communications), $[\![\Xi \vdash A\ \mathsf{type_s}]\!]^+$ (giving positive communications), and $[\![\Xi \vdash A\ \mathsf{type_s}]\!]^-$ (giving negative communications) are locally continuous. In particular, whenever A is closed, the family contains a single well-woven embedding

$$\langle A \rangle : [\![A]\!] \to [\![A]\!]^+ \times [\![A]\!]^-,$$

and this embedding is an object of $\mathbf{CYO}(\mathbf{Stab}_\perp)$.

The semantics of the functional layer follows the standard approach [Cro93; Gun92; Rey98; Sto77; Ten95]. In particular, terms denote continuous functions between pointed dcpos. The semantics of value transmission implies that these dcpos should be dI-domains, and that terms

[8] CYO pluricategories are named in honour of Choose Your Own Adventure book series, a kind of interactive fiction similar to the interactive surveys described above. They are studied in section 8.2.

[9] We use pluricategories so that we can interpret processes and configurations in the same semantic universe.

[10] Frugality is discussed in section 8.2. Jointly, frugality, junk-freedom, and completeness are sufficient conditions for functions of type eq. (107) to be morphisms in a CYO pluricategory.

[11] These embeddings are not rigid, i.e., they are embeddings relative to the pointwise order. This poses no difficulties in the treatment of recursive types. Indeed, though recursive session types are constructed using ω-colimits, these non-rigid embeddings are not links of the corresponding ω-chain.

should denote stable functions. However, the open status of conjecture 8.2.25 complicates this otherwise pleasant account. In particular, it is unknown whether dcpos of junk-free, complete, frugal functions are dI-domains. This in turn means that it is unknown whether the dcpo of quoted processes—the denotation of (T{})—is a dI-domain. We escape this issue by assuming that we do not transmit quoted processes, an assumption further justified in section 8.1.1. Even though we cannot send or receive quoted processes, we must nevertheless be able to work with them in the functional layer. This leads to two denotational semantics of the functional layer, where the first is a special case of the second. For convenience and conciseness, we call types and terms that do not use the functional layer *purely functional*:

Definition 8.1.5. A functional type $\Xi \vdash \tau\ \mathsf{type}_s$ is **purely functional** if its derivation does not use (T{}); it is **impure** otherwise. A functional term $\Psi \Vdash \tau :$ is **purely functional** if all types appearing in its derivation are purely functional. ◂

The first semantics is for purely functional types and terms. In this case, a type $\Xi \vdash \tau\ \mathsf{type}_s$ denotes a constant functor $[\![\Xi \vdash \tau\ \mathsf{type}_s]\!] : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}$. We use constant functions because we assume as a simplifying assumption that functional types are closed (see assumption 8.1.7). A term $\Psi \Vdash M : \tau$ denotes a stable continuous function

$$[\![\Psi \Vdash M : \tau]\!] : [\![\Psi]\!] \to [\![\tau]\!]$$

where $[\![\Psi]\!] = \prod_{x:\tau \in \Psi} [\![\tau]\!]$. Otherwise, $\Xi \vdash \tau\ \mathsf{type}_s$ denotes a constant functor $[\![\Xi \vdash \tau\ \mathsf{type}_s]\!] : \prod_{\alpha \in \Xi} \mathbf{DCPO}_\perp \to \mathbf{DCPO}_\perp$, and $[\![\Psi \Vdash M : \tau]\!]$ is only assumed to be continuous.

Remark 8.1.6. If we allow session-typed communications to denote pointed dcpos instead of pointed dI-domains, then we could drop the above bifurcation of our semantics. Indeed, we could interpret the entire functional layer in $DCPO_{\perp!}$, and the process layer in $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. Unfortunately, in doing so, we lose the semantic properties captured by dI-domains that we described following question 8.1.3.

The final iteration of our process semantics addresses the process layer's use of contexts of functional variables. It is given by analogy with the semantics of the functional layer: a process $\Psi\ ;\ \Delta \vdash P :: a : A$ denotes a continuous function

$$[\![\Psi\ ;\ \Delta \vdash P :: a : A]\!] : [\![\Psi]\!] \to \mathbf{JFC}\,[\langle \Delta \rangle \to \langle A \rangle]$$

where $\mathbf{JFC}\,[\langle \Delta \rangle \to \langle A \rangle]$ is the pointed dcpo of junk-free, continuous, frugal functions from $\langle \Delta \rangle$ to $\langle A \rangle$ in $\mathbf{CYO}(\mathbf{Stab}_\perp)$, stably ordered.

Finally, a configuration $\Gamma \vdash \mathcal{C} :: \Delta$ denotes a junk-free, complete, and frugal morphism

$$[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] : \langle \Gamma \rangle \to \langle \Delta \rangle$$

in $\mathbf{CYO}(\mathbf{Stab})$.

To summarize the above development:

- An open session type $\Xi \vdash A\ \mathsf{type}_s$ denotes a natural family of well-woven embeddings

  $$\langle \Xi \vdash A\ \mathsf{type}_s \rangle : [\![\Xi \vdash A\ \mathsf{type}_s]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type}_s]\!]^+ \times [\![\Xi \vdash A\ \mathsf{type}_s]\!]^- : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}.$$

  It is a 2-cell in the 2-category **CFP** defined in section 4.5.2.
- A purely functional type $\Xi \vdash \tau\ \mathsf{type}_s$ denotes a constant functor.

  $$[\![\Xi \vdash \tau\ \mathsf{type}_s]\!] : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}.$$

- An impure functional type $\Xi \vdash \tau\ \mathsf{type}_s$ denotes a constant functor

  $$[\![\Xi \vdash \tau\ \mathsf{type}_s]\!] : \prod_{\alpha \in \Xi} \mathbf{DCPO}_\perp \to \mathbf{DCPO}_\perp.$$

- A configuration $\Gamma \vdash \mathcal{C} :: \Delta$ denotes a junk-free, complete, frugal morphism

  $$[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] : \langle \Gamma \rangle \to \langle \Delta \rangle$$

  in $\mathbf{CYO}(\mathbf{Stab}_\perp)$.

- A process $\Psi \,;\, \Delta \vdash P :: a : A$ denotes a continuous function
$$[\![\Psi \,;\, \Delta \vdash P :: a : A]\!] : [\![\Psi]\!] \to \mathbf{JFC}\left[\langle \Delta \rangle \to \langle A \rangle\right].$$
In particular, for all $u \in [\![\Psi]\!]$,
$$[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u : \langle \Delta \rangle \to \langle A \rangle$$
is a junk-free, continuous, frugal morphism in $\mathbf{CYO}(\mathbf{Stab}_\perp)$.
- A functional term $\Psi \Vdash M : \tau$ denotes a continuous function
$$[\![\Psi \Vdash M : \tau]\!] : [\![\Psi]\!] \to [\![\tau]\!]$$
where $[\![\Psi]\!] = \prod_{x:\tau \in \Psi} [\![\tau]\!]$. Its denotation is stable if $\Psi \Vdash M : \tau$ is purely functional.

8.1.1. Simplifying Assumptions. Our semantics makes two simplifying assumptions.

Assumption 8.1.7. All types in the functional layer are closed, i.e., that whenever $\Xi \vdash \tau\ \mathsf{type}_\mathsf{s}$, then τ has no free variables.

Assumption 8.1.7 avoids complexities caused by mixed-variant functors, especially when it comes to interpreting recursive types. Techniques for solving domain equations involving mixed-variant functors are well known [AJ95, § 5.3.3], and we conjecture that extending our semantics to handle open functional types will pose no significant technical difficulty.

Assumption 8.1.8. The rule (T{}) never appears in a derivation of (C∧) or (C⊃), i.e., all functional types appearing in a derivation of (C∧) or (C⊃) are purely functional.

Assumption 8.1.8 is due to the open status of conjecture 8.2.25. Concretely, dcpos of session-typed communications are assumed to be dI-domains, but it remains unknown whether the denotations of processes form dI-domains. As a result, processes cannot (yet) be included in session-typed communications.

Our two use-dependent interpretations of the functional layer do not pose any semantic difficulties. This is because one interpretation is a special case of the other. Indeed, dI-domains are special cases of dcpos, and strict stable continuous functions are special cases of strict continuous functions. This means that we can use the more specialized interpretation in all settings where the more relaxed interpretation is allowed.

To avoid trivializing the functional layer, we extend Polarized SILL with eager natural numbers as a base type:

$$\frac{}{\Xi \vdash \mathbf{nat}\ \mathsf{type}_\mathsf{f}}\ (\text{T-}\mathbb{N}) \qquad \frac{}{\Psi \Vdash \mathsf{o} : \mathbf{nat}}\ (\text{F-Z}) \qquad \frac{\Psi \Vdash M : \mathbf{nat}}{\Psi \Vdash s(M) : \mathbf{nat}}\ (\text{F-S})$$

$$\frac{}{\mathsf{o} \Downarrow \mathsf{o}}\ (\text{EV-Zero}) \qquad \frac{M \Downarrow v}{\mathsf{s}(M) \Downarrow \mathsf{s}(v)}\ (\text{EV-Succ})$$

This provides us with suitable base type when we cannot use (T{}). In general, we expect the functional layer to be extended with whichever base types the user desires.

8.2. Choose Your Own Categories

Remark 8.2.1. Recall that a collection $d_i : D_i \to D_i^+ \times D_i^-$, $1 \le i \le n$, of embeddings determines an embedding $(d_1, \ldots, d_n) : \prod_{i=1}^n D_i \to (\prod_{i=1}^n D_i^+) \times (\prod_{i=1}^n D_i^-)$ by lemma 2.2.49.

Definition 8.2.2. Let $\mathbf{C}$ be a traced cartesian $\mathbf{O}$-category. The **CYO pluricategory** $\mathbf{CYO}(\mathbf{C})$ is the pluricategory given by the following data:

- Objects are embeddings $a = \langle a^+, a^- \rangle : A \to A^+ \times A^-$ such that
$$\mathrm{Tr}^{A^+ \times A^-}_{A,A}\left(\langle a^p, a^+ \circ a^p \rangle \times \langle a^- \circ a^p, a^p \rangle\right) = \langle a^p, a^p \rangle.$$
We often abuse notation and write A, A^+, A^-, and A^p for a, a^+, a^-, and a^p, respectively. Object lists are ranged over by capital Greek letters.

- Morphisms $f : A_1, \ldots, A_n \to B_1, \ldots, B_m$ with $n, m \geq 0$ are morphisms
$$\left(\prod_{i=1}^{n} A_i^+\right) \times \left(\prod_{j=1}^{m} B_j^-\right) \to \left(\prod_{i=1}^{n} A_i\right) \times \left(\prod_{j=1}^{m} B_j\right)$$
of $\mathbf{C}$ that are closed under composition with the identity morphisms of $\mathbf{CYO}(\mathbf{C})$.
- The identity morphism for $a : A \to A^+ \times A^-$ is $\mathsf{id}_a = \langle a^p, a^p \rangle$.
- The composition $g \circ f : \Lambda, \Phi, \Xi \to \Gamma, \Xi, \Delta$ of $f : \Phi \to \Gamma, \Pi, \Delta$ and $g : \Lambda, \Pi, \Sigma \to \Xi$ is[12]
$$\mathrm{Tr}^{\Pi^+ \times \Pi^-}\left(((\mathsf{id}_{\Phi \times \Gamma \times \Delta} \times \Pi^+) \circ f) \times ((\Pi^- \times \mathsf{id}_{\Lambda \times \Sigma \times \Xi}) \circ g)\right). \qquad \blacktriangleleft$$

Remark 8.2.3. The object corresponding to the empty list of $\mathbf{CYO}(\mathbf{C})$ is determined by the terminal object $\top$ of $\mathbf{C}$: $\varepsilon : \top \to \top \times \top$.

Proposition 8.2.4. *The data of definition 8.2.2 determines a pluricategory.*

Proof. By straightforward string diagram manipulations in the underlying category, using properties of traces. □

Given a list Δ of objects in $\mathbf{CYO}(\mathbf{C})$, we abuse notation and write Δ, Δ^p, Δ^+, and Δ^- for the associated embedding and projection given by remark 8.2.1, and their associated projections.

Remark 8.2.5. Every object $A \to A^+ \times A^-$ in $\mathbf{CYO}(\mathbf{C})$ determines a dual object $\overline{(A \to A^+ \times A^-)} = (A \to A^+ \times A^- \cong A^- \times A^+)$. As a result, every morphism $f : \Gamma \to \Delta$ can equivalently[13] be thought of as a morphism $f : \varepsilon \to \overline{\Gamma}, \Delta$ or $f : \Gamma, \overline{\Delta} \to \varepsilon$. We will use this observation below to simplify calculations below. In particular, it will be sufficient to consider only compositions of the form $g \circ f : \Phi \to \Gamma$ for $f : \Phi \to \Delta$ and $g : \Delta \to \Gamma$. This is because morphisms $f : \Phi \to \Phi_1, \Pi, \Phi_2$ and $g : \Gamma_1, \Pi, \Gamma_2 \to \Gamma$ can be thought of as morphisms $\Phi, \overline{\Phi_1}, \overline{\Phi_2} \to \Pi$ and $\Pi \to \Gamma, \overline{\Gamma_1}, \overline{\Gamma_2}$, respectively. The resulting composition $\Phi, \overline{\Phi_1}, \overline{\Phi_2} \to \Gamma, \overline{\Gamma_1}, \overline{\Gamma_2}$ is equal to the composition $g \circ f : \Gamma_1, \Phi, \Gamma_2 \to \Phi_1, \Gamma, \Phi_2$ modulo the required symmetry isomorphisms.

8.2.1. CYO Categories Over Categories of Pointed DCPOs. We study sufficient conditions for embeddings and morphisms of $\mathbf{DCPO}_\perp$ to be objects and morphisms of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. Several of these were semantically motivated in section 8.1. Whenever we speak of functions of the form $p : \Delta^+ \to \Delta$ or $p : \Gamma^+ \times \Delta^- \to \Gamma \times \Delta$, we assume that embeddings $\Delta \to \Delta^+ \times \Delta^-$ and $\Gamma \to \Gamma^+ \times \Gamma^-$ have been fixed.

We use corollary 2.3.8 to explicitly characterize sequential composition in $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. Consider morphisms $p : \Delta \to \Gamma$ and $q : \Gamma \to \Psi$, and let $(\delta^+, \psi^-) \in \Delta^+ \times \Psi^-$ be arbitrary. Then $(q \circ p)(\delta^+, \psi^-) = (\delta, \psi)$ where $(\delta, \gamma^+, \gamma^-, \psi)$ are minimum satisfying
$$p(\delta^+, \gamma^-) = (\delta, \gamma_p), \qquad \Gamma^+(\gamma_p) \sqsubseteq \gamma^+,$$
$$q(\gamma^+, \psi^-) = (\gamma_q, \psi), \qquad \Gamma^-(\gamma_q) \sqsubseteq \gamma^-.$$
In this case, we say that (γ^+, γ^-) **witness** the composition $q \circ p$.

8.2.1.1. *Objects in* $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. We characterize the embeddings that determine objects of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$, and we study their properties.

Definition 8.2.6. An embedding $e : A \to A^+ \times A^-$ is **well-woven** if for all $(a^+, a^-) \in A^+ \times A^-$, if $(\alpha^+, \alpha^-) \in A^+ \times A^-$ are chosen minimal such that both
$$(e^+ \circ e^p)(a^+, \alpha^-) \sqsubseteq \alpha^+$$
$$(e^- \circ e^p)(\alpha^+, a^-) \sqsubseteq \alpha^-,$$
then $e^p(a^+, \alpha^-) = e^p(\alpha^+, a^-)$. We call the above system of inequalities the **weaving equations** for e and (a^+, a^-). $\blacktriangleleft$

[12] We leave the symmetry isomorphisms implicit in the trace for legibility, both here and throughout.

[13] Strictly speaking, there are implicit symmetry isomorphisms permuting the products, but we safely ignore these to avoid drowning in a sea of notation and pedantry.

Proposition 8.2.7. *Let $e : A \to A^+ \times A^-$ be well-woven and let $(a^+, a^-) \in A^+ \times A^-$ be arbitrary. If $(\alpha^+, \alpha^-) \in A^+ \times A^-$ is minimal such that for some α_1 and α_2,*

$$e^p(a^+, \alpha^-) = \alpha_1, \qquad e^+(\alpha_1) \sqsubseteq \alpha^+,$$
$$e^p(\alpha^+, a^-) = \alpha_2, \qquad e^-(\alpha_2) \sqsubseteq \alpha^-,$$

then $\alpha_1 = \alpha_2$ and $e^p(a^+, a^-) = e^p(\alpha^+, \alpha^-) = \alpha_1$. In particular, $e(\alpha_1) = (\alpha^+, \alpha^-)$.

Proof. It is immediate by the definition of well-woven embedding that $\alpha_1 = \alpha_2$. Using proposition 2.2.16, we recognize (α^+, α^-) as the least fixed point of the function

$$\lambda(x^+, x^-).((e^+ \circ e^p)(a^+, x^-), (e^- \circ e^p)(x^+, a^-)).$$

We deduce that $e(\alpha_1) = (\alpha^+, \alpha^-)$. It follows by monotonicity and properties of projections that $(\alpha^+, \alpha^-) \sqsubseteq (a^+, a^-)$.

Projections preserve existing infima by proposition 2.2.19, so

$$e^p(\alpha^+, \alpha^-) = e^p(a^+ \sqcap \alpha^+, \alpha^- \sqcap a^-) = e^p(a^+, \alpha^-) \sqcap e^p(\alpha^+, a^-) = \alpha_1 \sqcap \alpha_2 = \alpha_1.$$

By continuity,

$$e^p(a^+, a^-) = e^p(a^+ \sqcup \alpha^+, \alpha^- \sqcup a^-) = e^p(a^+, \alpha^-) \sqcup e^p(\alpha^+, a^-) = \alpha_1 \sqcup \alpha_2 = \alpha_1. \qquad \square$$

Corollary 8.2.8. *An embedding $e : A \to A^+ \times A^-$ determines an object of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$ if and only if it is well-woven.*

Proof. Necessity is an immediate corollary of proposition 8.2.7 and corollary 2.3.8. To see sufficiency, assume that e is an object of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$ and let $(a^+, a^-) \in A^+ \times A^-$ be arbitrary. By assumption,

$$\mathrm{Tr}^{A^+ \times A^-}_{A,A}(\langle e^p, e^+ \circ e^p \rangle \times \langle e^- \circ e^p, e^p \rangle) = \langle e^p, e^p \rangle. \tag{109}$$

Let $(\alpha^+, \alpha^-) \in A^+ \times A^-$ be minimal such that for some α_1 and α_2,

$$e^p(a^+, \alpha^-) = \alpha_1 \qquad e^+(\alpha_1) \sqsubseteq \alpha^+$$
$$e^p(\alpha^+, a^-) = \alpha_2 \qquad e^-(\alpha_2) \sqsubseteq \alpha^-.$$

By corollary 2.3.8,

$$\mathrm{Tr}^{A^+ \times A^-}_{A,A}(\langle e^p, e^+ \circ e^p \rangle \times \langle e^- \circ e^p, e^p \rangle)(a^+, a^-) = (\alpha_1, \alpha_2).$$

By eq. (109),

$$e^p(a^+, \alpha^-) = \alpha_1 = e^p(a^+, a^-) = \alpha_2 = e^p(\alpha^+, a^-).$$

We conclude that e is well-woven. $\square$

The following technical lemma generalizes proposition 8.2.7. It will be essential to showing that processes sending and receiving channels are morphisms in our semantic domain.

Lemma 8.2.9. *Let $e : A \to A^+ \times A^-$ be well-woven and let $(a^+, a^-) \in A^+ \times A^-$ be arbitrary. The minimum solution $(\alpha_1^+, \alpha_2^+, \alpha_1^-, \alpha_2^-) \in A^+ \times A^+ \times A^- \times A^-$ such that for some α_0, α_1, and α_2*

$$\alpha_0 = e^p(a^+, \alpha_2^-) \qquad A^+(\alpha_0) \sqsubseteq \alpha_1^+$$
$$\alpha_1 = e^p(\alpha_1^+, \alpha_1^-) \qquad A^+(\alpha_1) \sqsubseteq \alpha_2^+ \qquad A^-(\alpha_1) \sqsubseteq \alpha_2^-$$
$$\alpha_2 = e^p(\alpha_2^+, a^-) \qquad A^-(\alpha_2) \sqsubseteq \alpha_1^-$$

is $(\alpha^+, \alpha^+, \alpha^-, \alpha^-)$ where $(e \circ e^p)(a^+, a^-) = (\alpha^+, \alpha^-)$.

Proof. Observe first that a solution exists: take the solution from the statement. Now consider any minimal solution $(\alpha_1^+, \alpha_2^+, \alpha_1^-, \alpha_2^-)$. Looking at the system as two systems of four equations, we deduce that $\alpha_0 = \alpha_1 = \alpha_2$ by proposition 8.2.7. By the same result and monotonicity, $e^p(a^+, \alpha_1^-) = e^p(\alpha_1^+, a^-) = \alpha_1$, so

$$e^p(a^+, a^-) = e^p(a^+ \sqcup \alpha_1^+, \alpha_1^- \sqcup a^-) = e^p(a^+, \alpha_1^-) \sqcup e^p(\alpha_1^+, a^-) = \alpha_1 \sqcup \alpha_1 = \alpha_1.$$

By the same result, $(\alpha_1^+, \alpha_2^-) = e(\alpha_0) = e(\alpha_1) = (\alpha_2^+, \alpha_1^-)$. This implies that the minimal solution has the desired form. Because it is entirely determined by the above sequence of equalities, it is minimum. □

8.2.1.2. *Morphisms in* $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. We formally define the notions of junk-freedom and completeness that were motivated in section 8.1: They jointly with a third condition, *frugality*, will be sufficient conditions for a morphism $p : \Delta^+ \to \Delta$ of $\mathbf{DCPO}_\perp$ to be a morphism $\Delta \to \varepsilon$ of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$.

Definition 8.2.10. Let $\Delta \to \Delta^+ \times \Delta^-$ be an embedding. A function $p : \Delta^+ \to \Delta$ in $\mathbf{DCPO}_\perp$ is:

- **junk-free** if $(\Delta^+ \upharpoonright \mathrm{im}(p), p^\circ)$ is an e-p-pair, where $p^\circ : \Delta^+ \to \mathrm{im}(p)$ is the corestriction of p to its image;
- **complete** if $\Delta^p \circ \langle \mathsf{id}, \Delta^- \circ p \rangle = p$;
- **frugal** if for all $\delta_0^+ \in \Delta^+$, $(\Delta \circ p)(\delta_0^+)$ is the least solution (δ^+, δ^-) to the "frugality system"

$$(\Delta^+ \circ \Delta^p)(\delta_0^+, \delta^-) \sqsubseteq \delta^+$$
$$(\Delta^- \circ p)(\delta^+) \sqsubseteq \delta^-.$$

A function $p : \Delta^+ \times \Psi^- \to \Delta \times \Psi$ is junk-free, complete, or frugal if $p : \Delta^+ \times \overline{\Psi}^+ \to \Delta \times \overline{\Psi}$ is respectively junk-free, complete, or frugal. A morphism $p : \Delta \to \Psi$ in $\mathbf{CYO}(\mathbf{DCPO}_\perp)$ is junk-free, complete, or frugal if its underlying morphism is respectively junk-free, complete, or frugal. We say that a function is **jfc** if it is junk-free, complete, and frugal. ◀

We already know examples of junk-free, complete, and frugal morphisms:

Proposition 8.2.11. *If* $a : A \to A^+ \times A^-$ *is an object of* $\mathbf{CYO}(\mathbf{DCPO}_\perp)$*, then* id_a *is junk-free, complete, and frugal.*

Proof. By definition, $\mathsf{id}_a : A \to A$ is junk-free if and only if $\mathsf{id}_a : A^+ \times A^- \to A \times A$ is junk-free, and this is the case if and only if $\mathsf{id}_a : A^+ \times \overline{A}^+ \to A \times \overline{A}$ is junk-free. So we must show that $((a^+ \times \overline{a}^+) \upharpoonright \mathrm{im}(\mathsf{id}_a), \mathsf{id}_a)$ is an e-p-pair. Observe that $\mathrm{im}(\mathsf{id}_a) = \{(\alpha, \alpha) \mid \alpha \in A\}$. Fixing an arbitrary element (α, α) of this image, we compute

$$(a^+ \times \overline{a}^+)(\alpha, \alpha) = (a^+(\alpha), a^-(\alpha)) = a(\alpha),$$

so by definition of e-p-pair,

$$(\mathsf{id}_a \circ (a \times \overline{a})^+)(\alpha, \alpha) = (\langle a^p, a^p \rangle \circ a)(\alpha) = (\alpha, \alpha).$$

Conversely, if $(\alpha^+, \alpha^-) \in A^+ \times \overline{A}^+$, then

$$((a \times \overline{a})^+ \circ \mathsf{id}_a)(\alpha^+, \alpha^-) = ((a^+ \circ a^p)(\alpha^+, \alpha^-), (a^- \circ a^p)(\alpha^+, \alpha^-)) \sqsubseteq (\alpha^+, \alpha^-)$$

by definition of e-p-pair. We conclude that id_a is junk-free.

To see that it is complete, we must show that

$$(a^p \times a^p) \circ \langle \mathsf{id}_{A \times A}, (a^- \times a^+) \circ \langle a^p, a^p \rangle \rangle = \langle a^p, a^p \rangle.$$

Then:

$$\begin{aligned} &(a^p \times a^p) \circ \langle \mathsf{id}_{A \times A}, (a^- \times a^+) \circ \langle a^p, a^p \rangle \rangle \\ &= (a^p \times a^p) \circ \langle \mathsf{id}_{A \times A}, a \circ a^p \rangle \\ &= \langle a^p, a^p \circ a \circ a^p \rangle, \end{aligned}$$

which by proposition 2.2.19:

$$\begin{aligned} &= \langle a^p, a^p \rangle \\ &= \mathsf{id}_a. \end{aligned}$$

We conclude that id_a is complete.

To show that it is frugal means to show for all $\alpha_o^+ \in A^+ \times \overline{A}^+$, that $((a, \overline{a})^p \circ \mathsf{id}_a)(\alpha_o^+)$ is the least solution $\alpha^+, \alpha^-)$ to the system

$$((a, \overline{a})^+ \circ (a, \overline{a})^p)(\alpha_o^+, \alpha^-) \sqsubseteq \alpha^+$$
$$((a, \overline{a})^- \circ \mathsf{id}_a)(\alpha^+) \sqsubseteq \alpha^-.$$

If some $\alpha_o^+ = (a_o^+, a_o^-)$, and let $\alpha^+ = (\alpha_1^+, \alpha_1^-) \in A^+ \times \overline{A}^+$ and $\alpha^- = (\alpha_2^-, \alpha_2^+) \in A^- \times \overline{A}^-$ be the least solution to the above system. The above system is equivalent to the system

$$(a^+ \circ a^p)(a_o^+, \alpha_2^-) \sqsubseteq \alpha_1^+$$
$$(a^- \circ a^p)(\alpha_2^+, a_o^-) \sqsubseteq \alpha_1^-$$
$$(a^- \circ a^p)(\alpha_1^+, \alpha_1^-) \sqsubseteq \alpha_2^-$$
$$(a^+ \circ a^p)(\alpha_1^+, \alpha_1^-) \sqsubseteq \alpha_2^+.$$

This statement has the same form as the system in the statement of lemma 8.2.9. By lemma 8.2.9, its least solution $(\alpha_1^+, \alpha_1^-, \alpha_2^+, \alpha_2^-)$ is given by $(\alpha_1^+, \alpha_1^-) = (\alpha_2^+, \alpha_2^-) = (a \circ a^p)(a_o^+, a_o^-)$. Frugality requires that we show

$$(a \circ a^p)(a_o^+, a_o^-) = (\alpha_1^+, \alpha_1^-),$$
$$(\overline{a} \circ a^p)(a_o^+, a_o^-) = (\alpha_2^-, \alpha_2^+),$$

and this is now immediate. □

Composition in CYO categories is defined using a trace operator, which hides the "complete" communications on the channels on which processes or configurations communicate. Proposition 8.2.12 states that both processes induce the same complete communications on those hidden channels.

Proposition 8.2.12. *Let $p : \Delta^+ \times \Psi^- \to \Delta \times \Psi$ and $q : \Psi^+ \times \Gamma^- \to \Psi \times \Gamma$ be junk-free, and let $(\delta^+, \gamma^-) \in \Delta^+ \times \Gamma^-$ be arbitrary. If (ψ^+, ψ^-) is minimum such that for some $(\delta, \psi_p, \psi_q, \gamma)$,*

$$p(\delta^+, \psi^-) = (\delta, \psi_p), \qquad \Psi^+(\psi_p) \sqsubseteq \psi^+,$$
$$q(\psi^+, \gamma^-) = (\psi_q, \gamma), \qquad \Psi^-(\psi_q) \sqsubseteq \psi^-,$$

then $\psi_q = \psi_p$.

Proof. We deduce that $\Psi^+(\psi_p) = \psi^+$ and $\Psi^-(\psi_q) = \psi^-$ by corollary 2.3.9. By junk-freedom, we also deduce that $\Psi^-(\psi_p) = \psi^-$ and $\Psi^+(\psi_q) = \psi^+$. It follows that $\Psi(\psi_p) = \Psi(\psi_q)$. But Ψ is an embedding and embeddings are injective, so $\psi_p = \psi_q$. □

We now turn our attention to showing that junk-free, frugal, complete functions are morphisms in $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. We begin with a pair of results characterizing witnesses for compositions of morphisms and identity morphisms in $\mathbf{CYO}(\mathbf{DCPO}_\perp)$.

Lemma 8.2.13. *Let $\Delta \to \Delta^+ \times \Delta^-$ be well-woven. Let $p : \Delta^+ \to \Delta$ and let $\delta_o^+ \in \Delta^+$ be arbitrary. Consider the least solution (δ_1^+, δ_2^-) such that for some δ_1 and δ_2,*

$$\Delta^p(\delta_o^+, \delta_2^-) = \delta_1 \qquad \Delta^+(\delta_1) \sqsubseteq \delta_1^+$$
$$p(\delta_1^+) = \delta_2 \qquad \Delta^-(\delta_2) \sqsubseteq \delta_2^-.$$

If p is complete, then $\delta_1 = \delta_2$ and $\Delta(\delta_1) = (\delta_1^+, \delta_2^-)$. If p is also frugal, then $p(\delta_o^+) = p(\delta_1^+)$.

Proof. We recognize the above system as defining the trace of a function. By corollary 2.3.9, we know that they are equalities:

$$\Delta^p(\delta_o^+, \delta_2^-) = \delta_1 \qquad \Delta^+(\delta_1) = \delta_1^+$$
$$p(\delta_1^+) = \delta_2 \qquad \Delta^-(\delta_2) = \delta_2^-.$$

Assume first that p is complete. We show that $\delta_1 = \delta_2$. Consider the least solution (a_1^+, a_2^-) to the following system:

$$\Delta^p(\delta_o^+, a_2^-) = a_1 \qquad \Delta^+(a_1) \sqsubseteq a_1^+$$
$$\Delta^p(a_1^+, \delta_2^-) = a_2 \qquad \Delta^-(a_2) \sqsubseteq a_2^-.$$

The embedding $\Delta \to \Delta^+ \times \Delta^-$ is well woven, so by proposition 8.2.7, $a_1 = a_2 = \Delta^p(\delta_o^+, \delta_2^-)$ and $\Delta(a_1) = (a_1^+, a_2^-)$. So $a_1 = a_2 = \delta_1$. We recognize the system as:

$$\Delta^p(\delta_o^+, a_2^-) = \delta_1 \qquad \Delta^+(\delta_1) \sqsubseteq \delta_1^+$$
$$\Delta^p(\delta_1^+, \delta_2^-) = \delta_1 \qquad \Delta^-(\delta_1) \sqsubseteq a_2^-.$$

But p was complete, so

$$\Delta^p(\delta_1^+, \delta_2^-) = \delta_2.$$

It follows that $\delta_1 = \delta_2$ and $a_2^- = \delta_2^-$. It follows that $\Delta(\delta_1) = (\delta_1^+, \delta_2^-)$.

Next, we show that $p(\delta_o^+) = p(\delta_1^+)$ when p is also frugal. By frugality, $(\Delta^- \circ p)(\delta_o^+) = \delta_2^-$. By completeness and this equality,

$$\begin{aligned} &p(\delta_o^+) \\ &= \Delta^p(\delta_o^+, (\Delta^- \circ p)(\delta_o^+)) \\ &= \Delta^p(\delta_o^+, \delta_2^-) \\ &= \Delta^p(\delta_o^+, (\Delta^- \circ p)(\delta_1^+)) \\ &= p(\delta_1^+). \end{aligned}$$

□

Proposition 8.2.14. *Let $\Delta \to \Delta^+ \times \Delta^-$ and $\Psi \to \Psi^+ \times \Psi^-$ be well-woven embeddings. Let $p : \Delta^+ \times \Psi^- \to \Delta \times \Psi$ be jfc, and $(\delta_o^+, \psi_o^-) \in \Delta^+ \times \Psi^-$ be arbitrary. If $p(\delta_o^+, \psi_o^-) = (\delta, \psi)$, $\Delta(\delta) = (\delta^+, \delta^-)$, and $\Psi(\psi) = (\psi^+, \psi^-)$, then*

(1) $\Delta^p(\delta_o^+, \delta^-) = \delta$ and $\Psi^p(\psi^+, \psi_o^-) = \psi$;

(2) (δ^+, δ^-) is the minimum solution (x^+, x^-) such that for some x_1 and x_2,

$$\Delta^p(\delta_o^+, x^-) = x_1, \qquad \Delta^+(x_1) \sqsubseteq x^+,$$
$$p(x^+, \psi_o^-) = (x_2, \_), \qquad \Delta^-(x_2) \sqsubseteq x^-.$$

(3) (ψ^+, ψ^-) is the minimum solution (x^+, x^-) such that for some x_1 and x_2,

$$p(\delta_o^+, x^-) = (\_, x_1), \qquad \Psi^+(x_1) \sqsubseteq x^+,$$
$$\Psi^p(x^+, \psi_o^-) = x_2, \qquad \Psi^-(x_2) \sqsubseteq x^-.$$

Proof. Item 1 is immediate by completeness of p. Indeed, completeness is exactly the claim that $(\Delta, \overline{\Psi})^p((\delta_o^+, \psi_o^-), (\delta^-, \psi^+)) = (\delta, \psi)$.

We show that (δ^+, δ^-) is the minimum solution (x^+, x^-) such that for some x_1 and x_2,

$$\Delta^p(\delta_o^+, x^-) = x_1, \qquad \Delta^+(x_1) \sqsubseteq x^+,$$
$$p(x^+, \psi_o^-) = (x_2, \_), \qquad \Delta^-(x_2) \sqsubseteq x^-.$$

By lemma 8.2.13, we know that

$$(\Delta, \overline{\Psi})^p((\delta_o^+, \psi_o^-), (\delta^-, \psi^+)) = (\delta, \psi), \qquad (\Delta, \overline{\Psi})^+(\delta, \psi) = (\delta^+, \psi^-),$$
$$p(\delta^+, \psi^-) = (\delta, \psi), \qquad (\Delta, \overline{\Psi})^-(\delta, \psi) = (\delta^-, \psi^+),$$
$$p(\delta_o^+, \psi_o^-) = (\delta, \psi), \qquad (\delta^+, \psi^-) \sqsubseteq (\delta_o^+, \psi_o^-).$$

We deduce by monotonicity that $p(\delta^+, \psi_o^-) = (\delta, \psi)$. By projecting out the desired components from four of these seven (in)equalities, we deduce that (δ^+, δ^-) is a solution (x^+, x^-) to the system

$$\Delta^p(\delta_o^+, x^-) = x_1, \qquad \Delta^+(x_1) \sqsubseteq x^+,$$
$$p(x^+, \psi_o^-) = (x_2, \_), \qquad \Delta^-(x_2) \sqsubseteq x^-. \tag{110}$$

Consider any other solution $(x^+, x^-) \sqsubseteq (\delta^+, \delta^-)$ to this system. Monotonicity implies that $((x^+, \psi^-), (x^-, \psi^+))$ is a solution to the frugality system

$$((\Delta, \overline{\Psi})^+ \circ (\Delta, \overline{\Psi})^p)((\delta^+_o, \psi^-_o), (x^-, \psi^+)) \sqsubseteq (x^+, \psi^-) \tag{111}$$

$$((\Delta, \overline{\Psi})^- \circ p)(x^+, \psi^-) \sqsubseteq (x^-, \psi^+) \tag{112}$$

for $p : \Delta^+ \times \overline{\Psi}^+ \to \Delta \times \Psi$. Indeed, eq. (111) follows by properties of products. To see eq. (112), observe that by eq. (110) and monotonicity:

$$((\Delta, \overline{\Psi})^- \circ p)(x^+, \psi^-) \sqsubseteq ((\Delta, \overline{\Psi})^- \circ p)(x^+, \psi^-_o) \sqsubseteq (x^-, \_).$$

By the above system of six (in)equalities, $x^+ \sqsubseteq \delta^+$ and monotonicity:

$$((\Delta, \overline{\Psi})^- \circ p)(x^+, \psi^-) \sqsubseteq ((\Delta, \overline{\Psi})^- \circ p)(\delta^+, \psi^-) \sqsubseteq (\delta^-, \psi^+).$$

But $x^- \sqsubseteq \delta^-$, so eq. (112) follows from these two sequences of inequalities. So $((x^+, \psi^-), (x^-, \psi^+))$ is indeed a solution to the frugality system. Recall that its least solution is $((\delta^+, \psi^-), (\delta^-, \psi^+))$, so $(x^+, x^-) = (\delta^+, \delta^-)$. We conclude that (δ^+, δ^-) is the minimum solution.

The third claim, concerning (ψ^+, ψ^-), follows by symmetry. □

Corollary 8.2.15. *If $\delta : \Delta \to \Delta^+ \times \Delta^-$ is an object of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$ and $p : \Delta^+ \to \Delta$ is continuous, complete, and frugal, then $p : \Delta \to \varepsilon$ is a morphism of $\mathbf{CYO}(\mathbf{DCPO}_\perp)$.*

Proof. The embedding δ is well-woven by corollary 8.2.8. It follows from proposition 8.2.14 that p is closed under composition with identity morphisms. □

The following proposition captures the intuition given in section 8.1 that the image of complete morphisms always includes the "the unanswered questions". Indeed, if we take δ^+_o to be the questions asked to p, then $\Delta^p(\delta^+_o, \perp)$ is the survey given by answering none of those questions. The statement says that $p(\delta^+_o)$ is at least as big as that survey.

Proposition 8.2.16. *If $p : \Delta^+ \to \Delta$ is complete and $p(\delta^+_o) = \delta$, then $\Delta^p(\delta^+_o, \perp) \sqsubseteq \delta$.*

Proof. Immediate by the completeness condition and monotonicity. □

We now consider properties of collections of morphisms in $\mathbf{CYO}(\mathbf{DCPO}_\perp)$. In particular, we show that jfc morphisms between bounded-complete dcpos form a dcpo.

Proposition 8.2.17. *Junk-free continuous functions $\Delta^+ \to \Delta$, ordered pointwise, form a dcpo. This dcpo is bounded-complete when Δ is bounded-complete and $\Delta \to \Delta^+ \times \Delta^-$ is an embedding.*

Proof. We start by showing that junk-free continuous functions are closed under directed suprema. Let M be a directed subset of $\mathbf{DCPO}_\perp[\Delta^+ \to \Delta]$ of junk-free functions, and set $F = \bigsqcup^\uparrow M$. We must show that

$$(\Delta^+ \upharpoonright \mathrm{im}(F), F)$$

is an e-p-pair.

We begin by showing that $(\Delta^+ \upharpoonright \mathrm{im}(F)) \circ F \sqsubseteq \mathsf{id}$:

$$(\Delta^+ \upharpoonright \mathrm{im}(F)) \circ F = \Delta^+ \circ F = \bigsqcup_{f \in M}{}^\uparrow \Delta^+ \circ f \sqsubseteq \bigsqcup_{f \in M}{}^\uparrow \mathsf{id} = \mathsf{id}.$$

Next, we show that $\mathsf{id} = F \circ \Delta^+ \restriction \mathrm{im}(F)$. Let $\delta^+ \in \Delta^+$ be arbitrary, and set $\delta = F(\delta^+)$. Then $\delta = \bigsqcup^\uparrow_{f \in M} f(\delta^+)$. We compute using proposition 2.2.11:

$$\begin{aligned}
&(F \circ \Delta^+)(\delta) \\
&= \bigsqcup_{f \in M}^\uparrow (f \circ \Delta^+)(\delta) \\
&= \left(\bigsqcup_{f \in M}^\uparrow f \circ \Delta^+\right)\left(\bigsqcup_{g \in M}^\uparrow g(\delta^+)\right) \\
&= \bigsqcup_{f \in M}^\uparrow (f \circ \Delta^+ \circ f)(\delta^+) \\
&= \bigsqcup_{f \in M}^\uparrow f(\delta^+) \\
&= F(\delta^+).
\end{aligned}$$

Because δ^+ was arbitrary, this establishes the result.

Next, assume that Δ is bounded-complete. We show that the collection of junk-free continuous functions $\Delta^+ \to \Delta$ is bounded complete. Let $p, p_1, p_2 : \Delta^+ \to \Delta$ be junk-free with $p_1 \sqsubseteq p$ and $p_2 \sqsubseteq p$. We must show that $p_1 \sqcup p_2 : \Delta^+ \to \Delta$ exists. It follows easily from bounded-completeness of Δ that the supremum exists in $\mathbf{DCPO}_\perp[\Delta^+ \to \Delta]$ and that $(p_1 \sqcup p_2)(\delta^+) = p_1(\delta^+) \sqcup p_2(\delta^+)$. We show that $p_1 \sqcup p_2$ is junk-free. We start by showing that $\Delta^+ \circ (p_1 \sqcup p_2) \sqsubseteq \mathsf{id}$. But this follows by monotonicity and the fact that p is junk-free:

$$\Delta^+ \circ (p_1 \sqcup p_2) \sqsubseteq \Delta^+ \circ p \sqsubseteq \mathsf{id}.$$

Next, we show that $\mathsf{id} = (p_1 \sqcup p_2) \circ \Delta^+$ when restricted to the image of $p_1 \sqcup p_2$, i.e., that

$$(p_1 \sqcup p_2) \circ \Delta^+ \circ (p_1 \sqcup p_2) = p_1 \sqcup p_2.$$

Observe first that Δ^+ preserves suprema. Indeed, $e : \Delta \to \Delta^+ \times \Delta^-$ is an embedding,[14] so a lower adjoint, and lower adjoints preserve suprema. But suprema in products, including $\Delta^+ \times \Delta^-$, are computed component-wise, so $\Delta^+ = \pi_1 \circ e$ also preserves suprema.[15] This implies that

$$\Delta^+ \circ (p_1 \sqcup p_2) = (\Delta^+ \circ p_1) \sqcup (\Delta^+ \circ p_2).$$

We deduce that

$$\begin{aligned}
&(p_1 \sqcup p_2) \circ \Delta^+ \circ (p_1 \sqcup p_2) \\
&= (p_1 \circ \Delta^+ \circ p_1) \sqcup (p_1 \circ \Delta^+ \circ p_2) \sqcup (p_2 \circ \Delta^+ \circ p_1) \sqcup (p_2 \circ \Delta^+ \circ p_2)
\end{aligned}$$

but p_1 and p_2 are junk-free, so:

$$= p_1 \sqcup (p_1 \circ \Delta^+ \circ p_2) \sqcup (p_2 \circ \Delta^+ \circ p_1) \sqcup p_2.$$

Recall that $\Delta^+ \circ p_i \sqsubseteq \mathsf{id}$ for $i = 1, 2$, so

$$(p_1 \circ \Delta^+ \circ p_2) \sqcup (p_2 \circ \Delta^+ \circ p_1) \sqsubseteq p_1 \sqcup p_2.$$

We deduce that

$$p_1 \sqcup (p_1 \circ \Delta^+ \circ p_2) \sqcup (p_2 \circ \Delta^+ \circ p_1) \sqcup p_2 = p_1 \sqcup p_2.$$

The result follows by transitivity. □

Corollary 8.2.18. *Fix an embedding $\Delta \to \Delta^+ \times \Delta^-$ between dI-domains. Stable junk-free continuous functions $\Delta^+ \to \Delta$, stably ordered, form a bounded-complete dcpo.*

[14] Warning: this does not imply that $\pi_1 \circ e : \Delta \to \Delta^+$ is an embedding.

[15] We find ourselves in the unfortunate position of punning on Δ^+ as a dcpo and Δ^+ as a morphism in the same sentence.

Proof. Consider a directed subset M of $\mathbf{Stab}\,[\Delta^+ \to \Delta]$, and assume that every function in M is junk-free. Its directed supremum $\bigsqcup^\uparrow M$ exists in $\mathbf{Stab}\,[\Delta^+ \to \Delta]$ because $\mathbf{Stab}\,[\Delta^+ \to \Delta]$ is a dcpo. The stable ordering implies the pointwise ordering, so M is also directed in $\mathbf{DCPO}_\perp\,[\Delta^+ \to \Delta]$. The directed supremum $\bigsqcup^\uparrow M$ in $\mathbf{Stab}\,[\Delta^+ \to \Delta]$ is computed pointwise, i.e., it coincides with the directed supremum of M in $\mathbf{DCPO}_\perp\,[\Delta^+ \to \Delta]$. We conclude that it is junk-free by proposition 8.2.17.

Now consider stable junk-free continuous functions $p, p_1, p_2 : \Delta^+ \to \Delta$ such that $p_i \sqsubseteq_s p$ for $i = 1, 2$. The upper bound $p_1 \sqcup p_2$ exists in $\mathbf{Stab}\,[\Delta^+ \to \Delta]$ because $\mathbf{Stab}\,[\Delta^+ \to \Delta]$ is bounded-complete. The same argument as the previous paragraph gives that it is junk-free. □

Proposition 8.2.19. *Fix an embedding $\Delta \to \Delta^+ \times \Delta^-$. Complete continuous functions $\Delta^+ \to \Delta$, ordered pointwise, form a dcpo.*

Proof. An easy consequence of continuity. Let M be a directed subset of $\mathbf{DCPO}_\perp\,[\Delta^+ \to \Delta]$ of complete functions, and set $F = \bigsqcup^\uparrow M$. We must show that

$$\Delta^p \circ \langle \mathsf{id}, \Delta^- \circ F \rangle = F.$$

Let δ_o^+ be arbitrary. We compute:

$$\begin{aligned} &\Delta^p(\delta_o^+, (\Delta^- \circ F)(\delta_o^+)) \\ &= \bigsqcup_{p \in M}^\uparrow \Delta^p(\delta_o^+, (\Delta^- \circ p)(\delta_o^+)) \end{aligned}$$

which by the assumption that p is complete:

$$\begin{aligned} &= \bigsqcup_{p \in M}^\uparrow p(\delta_o^+) \\ &= F(\delta_o^+) \\ &= \delta. \end{aligned}$$

We conclude that F is complete. □

Corollary 8.2.20. *Fix an embedding $\Delta \to \Delta^+ \times \Delta^-$ between dI-domains. Stable complete functions $\Delta^+ \to \Delta$, stably ordered, form a dcpo.*

Proof. Analogous to the proof of corollary 8.2.18. □

Proposition 8.2.21. *Frugal continuous functions, ordered pointwise, are closed under directed suprema.*

Proof. Let M be a directed subset of $\mathbf{DCPO}_\perp\,[\Delta^+ \to \Delta]$ of complete functions, and set $F = \bigsqcup^\uparrow M$. Let δ_o^+ be arbitrary. We must show that if

$$(\Delta^p \circ F)(\delta_o^+) = (\delta^+, \delta^-),$$

then (δ^+, δ^-) is the least solution (x^+, x^-) such that

$$\begin{aligned} (\Delta^+ \circ \Delta^p)(\delta_o^+, x^-) &\sqsubseteq x^+ \\ (\Delta^- \circ F)(x^+) &\sqsubseteq x^-. \end{aligned}$$

We recognize (δ^+, δ^-) as $\bigsqcap \mathfrak{M}$, where

$$\mathfrak{M} = \{(\delta_1^+, \delta_2^-) \mid (\Delta^+ \circ \Delta^p)(\delta_o^+, \delta_2^-) \sqsubseteq \delta_1^+ \wedge (\Delta^- \circ F)(\delta_1^+) \sqsubseteq \delta_2^-\}.$$

We know that $\bigsqcap \mathfrak{M}$ exists by proposition 2.2.16. Where $p \in M$, set

$$\mathfrak{T}_p = \{(\delta_1^+, \delta_2^-) \mid (\Delta^+ \circ \Delta^p)(\delta_o^+, \delta_2^-) \sqsubseteq \delta_1^+ \wedge (\Delta^- \circ p)(\delta_1^+) \sqsubseteq \delta_2^-\}.$$

We know by frugality that for each $p \in M$,

$$\bigsqcap \mathfrak{T}_p = (\Delta \circ p)(\delta_o^+).$$

Observe that by continuity

$$(\delta^+, \delta^-) = (\Delta \circ F)(\delta^+_o) = \bigsqcup_{f \in M}^{\uparrow} (\Delta \circ f)(\delta^+_o) = \bigsqcup_{f \in M}^{\uparrow} \bigsqcap \mathfrak{T}_f.$$

This implies that, to show that (δ^+, δ^-) is the least solution, it is sufficient to show that

$$\bigsqcup_{f \in M}^{\uparrow} \bigsqcap \mathfrak{T}_f \sqsubseteq \bigsqcap \mathfrak{M}.$$

To do so, it is sufficient to show that $\bigsqcap \mathfrak{T}_f \sqsubseteq \bigsqcap \mathfrak{M}$ for all $f \in M$. It is in turn sufficient to show that $\mathfrak{M} \subseteq \mathfrak{T}_f$ for all $f \in M$. We do so. Let $f \in M$ and $(d^+, d^-) \in \mathfrak{M}$ be arbitrary. By monotonicity of composition and the definition of $\mathfrak{M}$, we observe:

$$(\Delta^- \circ f)(d^+) \sqsubseteq (\Delta^- \circ F)(d^+) \sqsubseteq d^-.$$

It is immediate by the definition of $\mathfrak{M}$ that $(\Delta^+ \circ \Delta^p)(\delta^+_o, d^-) \sqsubseteq d^+$. It follows that $(d^+, d^-) \in \mathfrak{T}_f$ as desired. We conclude that F is frugal. □

Corollary 8.2.22. *Fix an embedding $\Delta \to \Delta^+ \times \Delta^-$ between dI-domains. Stable frugal functions $\Delta^+ \to \Delta$, stably ordered, form a dcpo.*

Proof. Analogous to the proof of corollary 8.2.18. □

Recall that junk-free functions are, by virtue of being upper-adjoints, always stable. Propositions 8.2.17, 8.2.19 and 8.2.21 and corollaries 8.2.18, 8.2.20 and 8.2.22 then imply:

Corollary 8.2.23. *The collection of junk-free, complete, frugal functions $\Delta \to \Psi$ in $\mathbf{CYO}(\mathbf{BC}_\perp)$ forms a dcpo under the pointwise ordering. The collection of junk-free, complete, frugal, stable functions $\Delta \to \Psi$ in $\mathbf{CYO}(\mathbf{Stab}_\perp)$ forms a dcpo $\mathbf{JFC}[\Delta \to \Psi]$ under the stable ordering.*

Proposition 8.2.24. *The dcpo $\mathbf{JFC}[\Delta \to \Psi]$ is pointed. Its bottom element is*

$$\lambda(\delta^+, \psi^-) \in \Delta^+ \times \Psi^-.(\Delta^p(\delta^+, \perp), \Psi^p(\perp, \psi^-)).$$

Proof. Let b be the function from the statement, and let $p \in \mathbf{JFC}[\Delta \to \Psi]$ be arbitrary. We must show that b is an element of $\mathbf{JFC}[\Delta \to \Psi]$ and that $b \sqsubseteq_s p$.

We show that it is junk-free. We start by showing that

$$b \circ (\Delta^+ \times \Psi^-) \circ b = b.$$

Let (δ^+, ψ^-) be arbitrary in its domain. We analyze the Δ and Ψ components separately. By proposition 2.2.19,

$$(\Delta^p \circ \Delta \circ \Delta^p)(\delta^+, \perp) = \Delta^p(\delta^+, \perp).$$

It follows that

$$\Delta^p(\Delta^+(\Delta^p(\delta^+, \perp)), \perp) = \Delta^p(\delta^+, \perp).$$

A similar analysis for Ψ gives the result. Next, we show that

$$(\Delta^+ \times \Psi^-) \circ b \sqsubseteq \mathsf{id}.$$

But this is immediate from the definition of b and the fact that Δ and Ψ are embeddings.

Next, we show that it is complete. Again, we analyze only the Δ component, and observe that the Ψ component will follow by symmetry. Let (δ^+, ψ^-) be arbitrary in the domain of b. We must show that

$$\Delta^p(\delta^+, (\Delta^- \circ \Delta^p(\delta^+, \perp))) = \Delta^p(\delta^+, \perp).$$

Observe that, by definition of e-p-pair, $(\Delta^- \circ \Delta^p(\delta^+, \perp)) = \perp$. The result is now obvious.

We turn to frugality. Let (δ^+, ψ^-) be arbitrary in the domain of b. Consider the frugality system:

$$\begin{array}{lll} \Delta^p(\delta^+, \delta_2^-) = \delta_1 & \Delta^+(\delta_1) \sqsubseteq \delta_1^+ & \Delta^-(\delta_2) \sqsubseteq \delta_2^- \\ \Psi^p(\psi_2^+, \psi^-) = \psi_1 & \Psi^-(\psi_1) \sqsubseteq \psi_1^- & \Psi^+(\psi_2) \sqsubseteq \psi_2^+ \\ b(\delta_1^+, \psi_1^-) = (\delta_2, \psi_2) & & \end{array}$$

The solution $(\delta_1^+, \delta_2^-, \psi_1^-, \psi_2^+) = (\Delta^+(\delta_1), \bot, \Psi^-(\psi_1), \bot)$ is minimum, and it is the one required for b to be frugal. So b is frugal.

Finally, we show that $b \sqsubseteq_s p$. We must show that for all $(\delta_1^+, \psi_1^-) \sqsubseteq (\delta_2^+, \psi_2^-)$,

$$b(\delta_1^+, \psi_1^-) = b(\delta_2^+, \psi_2^-) \sqcap p(\delta_1^+, \psi_1^-).$$

Setting $(\delta_p, \psi_p) = p(\delta_1^+, \psi_1^-)$, this means that we must show that:

$$\Delta^p(\delta_1^+, \bot) = \Delta^p(\delta_2^+, \bot) \sqcap \delta_p,$$
$$\Psi^p(\bot, \psi_1^-) = \Psi^p(\bot, \psi_2^-) \sqcap \psi_p.$$

Observe that Δ^p is stable and that $(\delta_2^+, \bot)$ and $(\delta_1^+, \Delta^-(\delta_p))$ are consistent. By completeness, $\delta_p = \Delta^p(\delta_1^+, \Delta^-(\delta_p))$. It follows that

$$\begin{aligned}
&\Delta^p(\delta_2^+, \bot) \sqcap \delta_p \\
&= \Delta^p(\delta_2^+, \bot) \sqcap \Delta^p(\delta_1^+, \Delta^-(\delta_p)) \\
&= \Delta^p(\delta_2^+ \sqcap \delta_1^+, \bot \sqcap \Delta^-(\delta_p)) \\
&= \Delta^p(\delta_1^+, \bot).
\end{aligned}$$

The proof for the Ψ component is analogous. □

The following conjecture has important consequences for our semantics of Polarized SILL. Indeed, if conjecture 8.2.25 is true, then we can drop assumption 8.1.8. The first difficulty in proving conjecture 8.2.25 is showing that complete and frugal functions are bounded-complete. It is also unclear that this collection of morphisms has a compact basis.

Conjecture 8.2.25. *The dcpo* $\mathbf{JFC}\left[\Delta \to \Psi\right]$ *is a dI-domain.*

8.3. Semantic Clauses

We define the denotations of judgments by induction on their derivation.

8.3.1. Manipulating channels. The forwarding processes forward communications as-is. Accordingly, they denote the identity morphisms for $\langle A \rangle$ composed with the appropriate labelling for channel names:

$$[\![\Psi\ ;\ a : A \vdash a \to b :: b : A]\!]u = \langle a : \langle A \rangle^p, b : \langle A \rangle^p \rangle \tag{113}$$
$$[\![\Psi\ ;\ a : A \vdash a \leftarrow b :: b : A]\!]u = \langle a : \langle A \rangle^p, b : \langle A \rangle^p \rangle \tag{114}$$

We will see proposition 8.5.8 that the only effect of composing an arbitrary process with a forwarding process is to rename the forwarded channel.

Process composition is given by the obvious composition in the semantic universe:

$$[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u = [\![\Psi\ ;\ a : A, \Delta_2 \vdash Q :: c : C]\!]u \circ_a [\![\Psi\ ;\ \Delta_1 \vdash P :: a : A]\!]u \tag{115}$$

Consequently, we can deduce that cut is an associative and partially commutative operation "for free".

Processes can close channels of type $\mathbf{1}$. The close message is the only communication possible on a channel of type $\mathbf{1}$. As a result, whole communications of type $\mathbf{1}$ are elements of the two element domain $\{\bot \sqsubseteq \mathsf{close}\}$. All communication on a channel of type $\mathbf{1}$ is positive. As a result, its positive aspect is equal to its canonical interpretation. Its negative aspect is the constant functor onto the

one-element terminal object.

$$[\![\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+]\!] = \mathrm{diag}_{[\![\Xi]\!]}\{\bot \subsetneq \mathsf{close}\} \tag{116}$$

$$[\![\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+]\!]^+ = \mathrm{diag}_{[\![\Xi]\!]}\{\bot \subsetneq \mathsf{close}\} \tag{117}$$

$$[\![\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+]\!]^- = \mathrm{diag}_{[\![\Xi]\!]}\top_{\mathbf{Stab}} \tag{118}$$

$$\langle\!\langle \Xi \vdash \mathbf{1}\ \mathsf{type}_s^+\rangle\!\rangle^+ = \mathsf{id} \tag{119}$$

$$\langle\!\langle \Xi \vdash \mathbf{1}\ \mathsf{type}_s^+\rangle\!\rangle^- = \top \tag{120}$$

$$\langle\!\langle \Xi \vdash \mathbf{1}\ \mathsf{type}_s^+\rangle\!\rangle^p = \pi_1 \tag{121}$$

In our asynchronous setting, close a does not wait for a client before sending the close message. We interpret ($\mathbf{1}$R) as the constant function that sends the close message:

$$[\![\Psi\ ;\ \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}]\!]u\bot = \mathsf{close} \tag{122}$$

The process wait a; P blocks until it receives the close message. All other communication is handled by P. We interpret ($\mathbf{1}$L) by:

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, \mathsf{close}, c) & \text{if } a^+ = \mathsf{close}\\ ((\langle\!\langle\Delta\rangle\!\rangle^p(\delta^+, \bot), \bot, \langle\!\langle C\rangle\!\rangle^p(\bot, c^-)) & \text{otherwise}\end{cases}\\ &\quad \text{where } (\delta, c) = [\![\Psi\ ;\ \Delta \vdash P :: c : C]\!]u(\delta^+, c^-)\end{aligned} \tag{123}$$

Our treatment of $A \otimes B$ is analogous to the one from chapter 6. We treat communications of type $A \otimes B$ as a pair of communications: one for the sent channel and one for the continuation channel. We account for the potential absence of communication by lifting.

$$[\![\Xi \vdash A \otimes B\ \mathsf{type}_s^+]\!] = ([\![\Xi \vdash A\ \mathsf{type}_s^+]\!] \times [\![\Xi \vdash B\ \mathsf{type}_s^+]\!])_\bot \tag{124}$$

$$[\![\Xi \vdash A \otimes B\ \mathsf{type}_s^+]\!]^+ = ([\![\Xi \vdash A\ \mathsf{type}_s^+]\!]^+ \times [\![\Xi \vdash B\ \mathsf{type}_s^+]\!]^+)_\bot \tag{125}$$

$$[\![\Xi \vdash A \otimes B\ \mathsf{type}_s^+]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_s^+]\!]^- \times [\![\Xi \vdash B\ \mathsf{type}_s^+]\!]^- \tag{126}$$

$$\langle\!\langle \Xi \vdash A \otimes B\ \mathsf{type}_s^+\rangle\!\rangle^+ = (-)_\bot\ (\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^+ \times \langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^+) \tag{127}$$

$$\langle\!\langle \Xi \vdash A \otimes B\ \mathsf{type}_s^+\rangle\!\rangle^- = \mathsf{down} * (\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^- \times \langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^-) \tag{128}$$

The associated family of projections is:

$$\begin{aligned}&\langle\!\langle \Xi \vdash A \otimes B\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi([(a^+, b^+)], (a^-, b^-))\\ &= [(\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(a^+, a^-), \langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(b^+, b^-))]\end{aligned} \tag{129}$$

We abuse notation to pattern match in eq. (129). This family is strict in the positive component.

Recall that the process send a b; P sends the channel b over the channel a and continues as P. The complete communications of type B that we observe are the greatest communications consistent with the positive and negative input of type B that the process receives. This behaviour is analogous to the behaviour of ($\textsc{Fwd}^+$) in eq. (113). The continuation P handles all other communication.

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A]\!]u(\delta^+, b^+, (a_B^-, a_A^-))\\ &= (\delta, b, [(b, a)])\ \text{ where } \begin{cases}[\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u(\delta^+, a_A^-) = (\delta, a)\\ \langle\!\langle B\rangle\!\rangle^p(b^+, a_B^-) = b\end{cases}\end{aligned} \tag{130}$$

The client $b \leftarrow \mathsf{recv}\ a$; Q blocks until it receives a channel on a. When it receives a positive communication $[(b_o^+, a_o^+)]$ on a, it unpacks it into the two positive communications a_o^+ and b_o^+ expected by Q. It then combines the communication Q produces on $a : A, b : B$ into a single

communication on $a : B \otimes A$.

$$\begin{aligned}
&[\![\Psi\ ;\ \Delta, a : B \otimes A \vdash b \leftarrow \mathsf{recv}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-) \\
&= \begin{cases} (\delta, [(b, a)], c) & \text{if } a^+ = [(b_{\mathsf{o}}^+, a_{\mathsf{o}}^+)] \\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise} \end{cases} \\
&\quad \text{where } [\![\Psi\ ;\ \Delta, a : A, b : B \vdash P :: c : C]\!]u(\delta^+, a_{\mathsf{o}}^+, b_{\mathsf{o}}^+, c^-) = (\delta, a, b, c)
\end{aligned} \tag{131}$$

Example 8.3.1. The process below blocks until it receives a channel a of type $\mathbf{1}$ over the channel b, at which point the type of b becomes $\mathbf{1}$. Then, the process waits for the close messages on a and b before closing c. The element $[(\mathsf{close}, \mathsf{close})] \in [\![\mathbf{1} \otimes \mathbf{1}]\!]^+ = ([\![\mathbf{1}]\!]^+ \times [\![\mathbf{1}]\!]^+)_\bot$ corresponds to receiving the channel a, the close message on a, and the close message on b. The element $[(\bot, \bot)]$ corresponds to receiving a but no close messages, while the elements $[(\mathsf{close}, \bot)]$ and $[(\bot, \mathsf{close})]$ correspond to receiving a and one close message. The element $\bot$ means that a is never received. It is clear from the denotation that the process only closes c in the first case:

$$\begin{aligned}
&[\![\cdot\ ;\ b : \mathbf{1} \otimes \mathbf{1} \vdash a \leftarrow \mathsf{recv}\ b;\ \mathsf{wait}\ a;\ \mathsf{wait}\ b;\ \mathsf{close}\ c :: c : \mathbf{1}]\!]\bot(b^+ : \beta, c^- : \bot) \\
&= \begin{cases} (b : [(\mathsf{close}, \mathsf{close})], c : \mathsf{close}) & \text{if } \beta = [(\mathsf{close}, \mathsf{close})] \\ (b : [(\mathsf{close}, \bot)], c : \bot) & \text{if } \beta = [(\mathsf{close}, \bot)] \\ (b : [(\bot, \mathsf{close})], c : \bot) & \text{if } \beta = [(\bot, \mathsf{close})] \\ (b : \bot, c : \bot) & \text{if } \beta = \bot \end{cases}
\end{aligned}$$

◀

8.3.2. Functional Programming and Value Transmission. The functional layer is the simply-typed λ-calculus with a call-by-value semantics and a fixed-point operator. Arrow types are interpreted as strict function spaces to enforce a call-by-value semantics. We lift these function spaces to be able to detect divergence, e.g., to be able to denotationally differentiate the terms $\Gamma \Vdash \lambda x : \tau.\mathsf{fix}\ y.y : \tau \to \tau$ and $\Gamma \Vdash \mathsf{fix}\ x.x : \tau \to \tau$.[16] If $\Xi \vdash \tau \to \sigma\ \mathsf{type_f}$ is purely functional, then

$$[\![\Xi \vdash \tau \to \sigma\ \mathsf{type_f}]\!] = \mathrm{diag}_{[\![\Xi]\!]}\left((\mathbf{Stab}_{\bot!}\,[[\![\Xi \vdash \tau\ \mathsf{type_f}]\!] \to [\![\Xi \vdash \sigma\ \mathsf{type_f}]\!]])_\bot\right). \tag{132}$$

Otherwise,

$$[\![\Xi \vdash \tau \to \sigma\ \mathsf{type_f}]\!] = \mathrm{diag}_{[\![\Xi]\!]}\left((\mathbf{DCPO}_{\bot!}\,[[\![\Xi \vdash \tau\ \mathsf{type_f}]\!] \to [\![\Xi \vdash \sigma\ \mathsf{type_f}]\!]])_\bot\right). \tag{133}$$

The call-by-value semantics is adapted from [Gun92, chap. 6] to use dI-domains and stable functions. We let u range over $[\![\Psi]\!] = \prod_{x:\tau\in\Psi}[\![\tau]\!]$. The environment $[u \mid x \mapsto v] \in [\![\Psi, x : \tau]\!]$ maps x to v and y to $u(y)$ for all $y \in \Psi$. The fixed-point operator (F-Fix) is interpreted using the fixed-point operator defined in section 2.3.

$$[\![\Psi, x : \tau \Vdash x : \tau]\!]u = \pi_x^{\Psi,x} u \tag{134}$$

$$[\![\Psi \Vdash \lambda x : \tau.M : \tau \to \sigma]\!]u = \mathsf{up}\,(\mathsf{strict}\,(\lambda v \in [\![\tau]\!].[\![\Psi, x : \tau \Vdash M : \sigma]\!][u \mid x \mapsto v])) \tag{135}$$

$$[\![\Psi \Vdash MN : \sigma]\!]u = \mathsf{down}\,([\![\Psi \Vdash M : \tau \to \sigma]\!]u)\,([\![\Psi \Vdash N : \tau]\!]u) \tag{136}$$

$$[\![\Psi \Vdash \mathsf{fix}\ x.M : \tau]\!]u = [\![\Psi, x : \tau \Vdash M : \tau]\!]^\dagger u \tag{137}$$

These denotations are morphisms in $\mathbf{Stab}_\bot$ whenever the term is purely functional. Otherwise, they are morphisms in $\mathbf{DCPO}_\bot$.

We interpret (T-$\mathbb{N}$) as the constant functor onto the flat domain of natural numbers. We interpret natural numbers as the corresponding element.

$$[\![\Xi \vdash \mathbf{nat}\ \mathsf{type_f}]\!] = \mathrm{diag}_{\mathbf{Stab}_{\bot!}}\,\mathbb{N}_\bot \tag{138}$$

$$[\![\Psi \Vdash \mathsf{o} : \mathbf{nat}]\!]u = \mathsf{o} \tag{139}$$

$$[\![\Psi \Vdash s(M) : \mathbf{nat}]\!]u = \begin{cases} \bot & \text{if } [\![\Psi \Vdash M : \mathbf{nat}]\!]u = \bot \\ n+1 & \text{if } [\![\Psi \Vdash M : \mathbf{nat}]\!]u = n \end{cases} \tag{140}$$

[16] If we did not lift the function spaces in eqs. (132) and (133) and also left eqs. (135) and (137) unchanged, then these two terms would have the same denotation, even though one of them is a value and the other diverges.

We interpret quoted processes as elements of stably ordered dcpo of stable, junk-free, complete, and frugal continuous functions between the corresponding objects in $\mathbf{CYO}(\mathbf{Stab}_\perp)$. We lift this dcpo to account for non-termination when evaluating terms of this type: as typical in semantics for functional languages, the bottom element $\perp$ represents non-termination.[17]

$$\begin{aligned}&[\![\Xi \vdash \{a_o : A_o \leftarrow a_1 : A_1, \ldots, a_n : A_n\}\ \mathsf{type_f}]\!]\\ &= (-)_\perp \circ \mathrm{diag}_{[\![\Xi]\!]}\,(\mathbf{JFC}\,[\langle \Xi \vdash A_1\ \mathsf{type_s}\rangle_\perp, \ldots, \langle \Xi \vdash A_n\ \mathsf{type_s}\rangle_\perp \to \langle \Xi \vdash A_o\ \mathsf{type_s}\rangle_\perp])\end{aligned} \tag{141}$$

We take the component for the initial object $\perp$ as the representative of each family $\langle \Xi \vdash A_i\ \mathsf{type_s}\rangle$. This choice is arbitrary and not semantically meaningful thanks to the simplifying assumption that the types appearing in (T{}) are all closed. Indeed, in this case, $\langle \Xi \vdash A_i\ \mathsf{type_s}\rangle$ are constant families of morphisms.[18]

The (I-{}) and (E-{}) introduction and elimination rules respectively quote and unquote processes. Their denotations are:

$$[\![\Psi \Vdash a \leftarrow \{P\} \leftarrow \overline{a_i} : \{a : A \leftarrow \overline{a_i : A_i}\}]\!] = \mathsf{up} \circ [\![\Psi\ ;\ \overline{a_i : A_i} \vdash P :: a : A]\!] \tag{142}$$

$$[\![\Psi\ ;\ \overline{a_i : A_i} \vdash a \leftarrow \{M\} \leftarrow \overline{a}_i :: a : A]\!] = \mathsf{down} \circ [\![\Psi \Vdash M : \{a : A \leftarrow \overline{a_i : A_i}\}]\!] \tag{143}$$

Because these denotations involve quoted processes, we only know that they lie in **DCPO**. In eq. (142), we lift the image of $[\![\Psi\ ;\ \overline{a_i : A_i} \vdash P :: a : A]\!]$ to differentiate the bottom element of $\mathbf{JFC}\left[\overline{a_i : A_i} \to a : A\right]$ (the least junk-free, frugal, complete continuous function of that type) from the bottom element of $[\![\vdash \{a : A \leftarrow \overline{a_i : A_i}\}\ \mathsf{type_f}]\!]$ (the denotation of non-terminating computations).

A communication of type $\tau \wedge A$ is one of the following:

(1) a value $v \in [\![\tau]\!]$, followed by a communication a of satisfying A;
(2) a value $v \neq \perp$, followed by no further communication;
(3) the empty communication.

They respectively correspond to elements $(v, [a])$, $(v, [\perp])$, and $\perp$ of the smash product $[\![\tau]\!] \otimes [\![A]\!]_\perp$. We use the smash product instead of the cartesian product to rule out communications of the form $(\perp, a)$. These communications are problematic, because sequentially executed processes should not be able to communicate while evaluating a divergent term.[19] We lift the communications of type A to allow for the possibility that no communications follow the value of type τ. The value travels in the positive direction, so it only appears in the positive aspect.

$$[\![\Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+]\!] = [\![\Xi \vdash \tau\ \mathsf{type_f}]\!] \otimes [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+]\!]_\perp \tag{144}$$

$$[\![\Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+]\!]^+ = [\![\Xi \vdash \tau\ \mathsf{type_f}]\!] \otimes [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+]\!]_\perp^+ \tag{145}$$

$$[\![\Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+]\!]^- \tag{146}$$

$$\langle \Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+\rangle^+ = \mathsf{id}_{[\![\Xi \vdash \tau\ \mathsf{type_f}]\!]} \otimes (-)_\perp \langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^+\rangle^+ \tag{147}$$

$$\langle \Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+\rangle^- = \mathsf{down} * \pi_2 * \langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^+\rangle^- \tag{148}$$

$$\langle \Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+\rangle_\xi^p((v, [a^+]), a^-) = (v, [\langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^+\rangle_\xi^p(a^+, a^-)]) \tag{149}$$

We abuse notation to pattern match in eq. (149). This family is strict in the positive component.

The process $\_ \leftarrow \mathsf{output}\ a\ M;\ P$ sends a functional value on a and continues as P. To send the term M on a, we evaluate it under the current environment u to get an element $[\![\Psi \Vdash M : \tau]\!]u \in [\![\tau]\!]$. Divergence is represented by $\perp_{[\![\tau]\!]}$; the other elements represent values of type τ. If $[\![\Psi \Vdash M : \tau]\!]u$ represents a value, then we pair it with the communications of the continuation process P on a.

[17] This is in contrast to role played by bottom elements in domains of communications, where $\perp$ represents the absence of communication.

[18] That is, the components of each family are pairwise equal.

[19] Recall that $\perp$ is the denotation of divergent functional terms.

Otherwise, the process transmits nothing. The resulting complete communications are the least ones compatible with the input.

$$\begin{aligned}&[\![\Psi\ ;\ \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]u(\delta^+, a^-)\\ &= \begin{cases}(\delta, (v, [a])) & \text{if } [\![\Psi \Vdash M : \tau]\!]u = v \neq \bot\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \langle\tau \wedge A\rangle^p(\bot, a^-)) & \text{if } [\![\Psi \Vdash M : \tau]\!]u = \bot\end{cases}\\ &\quad \text{where } [\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u(\delta^+, a^-) = (\delta, a)\end{aligned} \tag{150}$$

The process $x \leftarrow \mathsf{input}\ a;\ P$ blocks until it receives a communication on the channel a. If a communication $(v, [\alpha^+])$ arrives on a^+, then the process binds v to x in the environment and continues as P with the remaining communication α^+ on a^+. If it receives no message, then we observe no communications on a, and the minimal consistent communications on the other channels.

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, (v, [a]), c) & \text{if } a^+ = (v, [a_o^+])\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise}\end{cases}\\ &\quad \text{where } [\![\Psi, x : \tau\ ;\ \Delta, a : A \vdash P :: c : C]\!][u \mid x \mapsto v](\delta^+, a_o^+, c^-) = (\delta, a, c)\end{aligned} \tag{151}$$

8.3.3. Shifts in Polarity. Recall that a communication of type $\downarrow A$ is a shift message followed by a communication of type A. By analogy with (C⊕), we could model complete communications of type $\downarrow A$ using a unary coalesced sum

$$[\![\downarrow A]\!] = \bigoplus_{l \in \{\mathsf{shift}\}} [\![A]\!]_\bot,$$

whose elements are $\bot$ and $(\mathsf{shift}, [a])$ for $a \in [\![A]\!]$. This domain is isomorphic to $[\![A]\!]_\bot$, so we omit the coalesced sum for clarity. "Downshifting" A to $\downarrow A$ introduces only *positive* communication (the "shift" message), so the negative aspect of $\downarrow A$ is the same as the negative aspect of A.

$$[\![\Xi \vdash \downarrow A\ \mathsf{type}_s^+]\!] = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]_\bot \tag{152}$$

$$[\![\Xi \vdash \downarrow A\ \mathsf{type}_s^+]\!]^+ = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]_\bot^+ \tag{153}$$

$$[\![\Xi \vdash \downarrow A\ \mathsf{type}_s^+]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]^- \tag{154}$$

$$\langle\Xi \vdash \downarrow A\ \mathsf{type}_s^+\rangle^+ = (-)_\bot\langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^+ \tag{155}$$

$$\langle\Xi \vdash \downarrow A\ \mathsf{type}_s^+\rangle^- = \mathsf{down} * \langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^- \tag{156}$$

$$\langle\Xi \vdash \downarrow A\ \mathsf{type}_s^+\rangle^p = (-)_\bot\langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^p \cdot \delta \tag{157}$$

In our asynchronous setting, the process $\mathsf{send}\ a\ \mathsf{shift};\ P$ always sends the shift message on a. This corresponds to lifting the output of P on the a component. We interpret (↓R) as:

$$[\![\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: a : \downarrow A]\!]u = (\mathsf{id} \times (a : \mathsf{up})) \circ [\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u \tag{158}$$

The client $\mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P$ blocks until it receives the shift message on a^+. We lower $[\![\downarrow A]\!] = [\![A]\!]_\bot^+$ to $[\![A]\!]^+$ to extract the positive communication expected by P, and then lift the output of P on a to capture that we did, indeed, receive the shift message:

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, a : \downarrow A \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, [a], c) & \text{if } a^+ = [a_o^+]\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise}\end{cases}\\ &\quad \text{where } (\delta, a, c) = [\![\Psi\ ;\ \Delta, a : A \vdash P :: c : C]\!]u(\delta^+, a_o^+, c^-)\end{aligned} \tag{159}$$

Example 8.3.2. Upshifts are the polar duals of downshifts. The following process waits for its client to synchronize with it before closing the channel. The protocol $\uparrow\mathbf{1}$ has denotations $[\![\uparrow\mathbf{1}]\!]^- = [\![\mathbf{1}]\!]_\bot^- =$

$\{\bot\}_\bot$ and $[\![\uparrow\mathbf{1}]\!]^+ = [\![\mathbf{1}]\!]^+ = \{\bot \fatsemi \mathsf{close}\}$. The element $[\bot] \in [\![\uparrow\mathbf{1}]\!]^-$ captures the synchronizing shift message. The process closes a if and only if it receives the shift message:

$$[\![\cdot\,;\cdot \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ \mathsf{close}\ a :: a : \uparrow\mathbf{1}]\!]\bot(a^- : \alpha) = \begin{cases} (a : \bot) & \text{if } \alpha = \bot \\ (a : [*]) & \text{if } \alpha = [\bot]. \end{cases}$$ ◂

8.3.4. Making Choices. A communication of type $\oplus\{l : A_l\}_{l\in L}$ is a label $k \in L$ sent in the positive direction followed by a communication satisfying A_k. Denotationally, this corresponds to tagging a communication $a_k \in [\![A_k]\!]$ with the label k. Tagged communications (k, a_k) are the elements of the disjoint union $\uplus_{l\in L}[\![A_l]\!]$. To account for the potential lack of communication, we lift this disjoint union. This lifted disjoint union is isomorphic to the coalesced sum $\bigoplus_{l\in L}[\![A_l]\!]_\bot$. Coalesced sums are coproducts in $\mathbf{Stab}_{\bot!}$, and we define the interpretation using a coalesced sum to make this structure evident. Explicitly, its elements are $\bot$ and $(k, [a_k])$ for $k \in L$ and $a_k \in [\![A_k]\!]$. The provider sends the label on the positive aspect of the channel, justifying eq. (161). The client does not know a priori which branch it will take: it must be ready to send negative information for each possible branch. This justifies eq. (162).

$$[\![\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+]\!] = \bigoplus_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_\mathsf{s}^+]\!]_\bot \tag{160}$$

$$[\![\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+]\!]^+ = \bigoplus_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_\mathsf{s}^+]\!]^+_\bot \tag{161}$$

$$[\![\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+]\!]^- = \prod_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_\mathsf{s}^+]\!]^- \tag{162}$$

$$\langle\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+\rangle^+ = \bigoplus_{l\in L} (-)_\bot \langle\Xi \vdash A_l\ \mathsf{type}_\mathsf{s}^+\rangle^+ \tag{163}$$

$$\langle\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+\rangle^- = \mathrm{diag}\,(\mathsf{down} * \langle\Xi \vdash A_l\ \mathsf{type}_\mathsf{s}^+\rangle^-)_{l\in L} \tag{164}$$

$$\langle\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+\rangle^p_\xi((k, [a_k^+]), (a_l^-)_{l\in L}) = (k, [\langle\Xi \vdash A_k\ \mathsf{type}_\mathsf{s}^+\rangle^p_\xi(a_k^+, a_k^-)]) \tag{165}$$

The category $\mathbf{Stab}_{\bot!}$ has zero morphisms and we use matrix notation[20] for morphisms from coproducts to products in eq. (164). Explicitly, each component of $\langle\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+\rangle^-$ is the strict morphism whose action on non-bottom elements is

$$\langle\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_\mathsf{s}^+\rangle^-_\xi\,(k, [a_k]) = \iota_k(\langle\Xi \vdash A_k\ \mathsf{type}_\mathsf{s}^+\rangle^-(a_k)).$$

We abuse notation to pattern match in eq. (165). This family is strict in the positive component.

To interpret ($\oplus$R), we extract from a^- the negative information a_k^- required by the continuation process P. Afterwards, we tag P's output on a with the label k.

$$[\![\Psi\,;\Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u\left(\delta^+, (a_l^-)_{l\in L}\right) = (\delta, (k, [a_k])) \tag{166}$$
$$\text{where } [\![\Psi\,;\Delta \vdash P :: a : A_k]\!]u\,(\delta^+, a_k^-) = (\delta, a_k)$$

The interpretation of ($\oplus$L) is analogous. If the client a label, then the case statement selects the corresponding branch, and we observe the received label.

$$\begin{aligned} &[\![\Psi\,;\Delta, a : \oplus\{l : A_l\}_{l\in L} \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: c : C]\!]u(\delta^+, a^+, c^-) \\ &= \begin{cases} (\delta, (l, [a_l]), c) & \text{if } a^+ = (l, [a_l^+]) \\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise} \end{cases} \\ &\quad \text{where } [\![\Psi\,;\Delta, a : A_l \vdash P_l :: c : C]\!]u\,(\delta^+, a_l^+, c^-) = (\delta, a_l, c) \end{aligned} \tag{167}$$

Example 8.3.3. We build on example 8.3.2. External choices $\&\{l : A_l\}_{l\in L}$ are the polar duals of internal choices. Let $A = \&\{\mathtt{j} : \uparrow\mathbf{1}, \mathtt{k} : \uparrow\mathbf{1}\}$. A provider of A receives a label and a synchronizing shift before closing the channel. The elements $(l, [[\bot]]) \in [\![A]\!]^-$ correspond to receiving the label l over a followed by a shift, while the elements $(l, [\bot])$ correspond to receiving l but no shift. In the

[20] It is defined in section 2.1.

first case, the denotation makes clear that the channel gets closed. In the second case, we see that no close message is sent:

$$\llbracket \cdot ; \cdot \vdash \mathsf{case}\ a\ \{l \Rightarrow \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ \mathsf{close}\ a\}_{l \in \{\mathtt{j},\mathtt{k}\}} :: a : A \rrbracket \bot (a^- : \alpha)$$
$$= \begin{cases} (a : (\mathtt{j}, [[\mathsf{close}]])) & \text{if } \alpha = (\mathtt{j}, [[\bot]]) \\ (a : (\mathtt{k}, [[\mathsf{close}]])) & \text{if } \alpha = (\mathtt{k}, [[\bot]]) \\ (a : (\mathtt{j}, [\bot])) & \text{if } \alpha = (\mathtt{j}, [\bot]) \\ (a : (\mathtt{k}, [\bot])) & \text{if } \alpha = (\mathtt{k}, [\bot]) \\ (a : \bot) & \text{if } \alpha = \bot \end{cases}$$

◀

8.3.5. Recursive Types. The substitution property (proposition 8.5.3) determines the denotation of the variable rule (CVar). Indeed, it forces eqs. (168) to (170) to be projection functors and eqs. (171) and (172) to be given by the identity natural transformation. These interpretations uniquely determine eq. (173). It is because of eq. (173), proposition 2.2.31, and remark 2.2.32 that dcpos of session-typed communications must be bounded-complete domains.

$$\llbracket \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \vdash \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \rrbracket = \pi_{\alpha}^{\Xi,\alpha} \tag{168}$$
$$\llbracket \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \vdash \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \rrbracket^{+} = \pi_{\alpha}^{\Xi,\alpha} \tag{169}$$
$$\llbracket \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \vdash \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \rrbracket^{-} = \pi_{\alpha}^{\Xi,\alpha} \tag{170}$$
$$\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \vdash \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \rangle^{+} = \mathsf{id} \tag{171}$$
$$\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \vdash \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \rangle^{-} = \mathsf{id} \tag{172}$$
$$\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \vdash \alpha\ \mathsf{type}_{\mathsf{s}}^{p} \rangle^{p} = \sqcap \tag{173}$$

As a step towards defining the denotations of general recursive types, we introduce bounded recursive types $\rho^n \alpha.A$ formed by:

$$\frac{\Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}}{\Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+}}\ (\mathrm{C}\rho_n^{+})$$

There are no communications of type $\rho^0 \alpha.A$, while a communication of type $\rho^{n+1}\alpha.A$ consists of an unfold message followed by a communication of type $[\rho^n \alpha.A/\alpha]A$. Their denotations are defined by induction on n. As in the denotations of (C↓), we use lifting to capture that an unfold message was sent, instead of an explicit unfold label. We use the following helper functor for convenience, which specializes the functor Ω from proposition 4.5.1:

$$\mathsf{iter}_n : \mathbf{CAT}\left[\mathbf{Stab}_{\bot!} \to \mathbf{Stab}_{\bot!}\right] \to \mathbf{Stab}_{\bot!}$$
$$\mathsf{iter}_n F = \left(\Omega(\bot, \bot, F)\right)(n) = F^n \bot$$
$$\mathsf{iter}_n(\eta : F \Rightarrow G) = \left(\Omega(\bot, \bot, F)\right)_n = \left(\eta^{(n)}\right)_{\bot} : F^n \bot \to G^n \bot.$$

We also use the abstraction functor Λ:[21]

$$\llbracket \Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+} \rrbracket = \mathsf{iter}_n * \left(\Lambda\left((-)_{\bot} \llbracket \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+} \rrbracket\right)\right) \tag{174}$$
$$\llbracket \Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+} \rrbracket^{+} = \mathsf{iter}_n * \left(\Lambda\left((-)_{\bot} \llbracket \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+} \rrbracket^{+}\right)\right) \tag{175}$$
$$\llbracket \Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+} \rrbracket^{-} = \mathsf{iter}_n * \left(\Lambda \llbracket \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+} \rrbracket^{-}\right) \tag{176}$$
$$\langle \Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+} \rangle^{+} = \mathsf{iter}_n * \left(\Lambda\left((-)_{\bot} \langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+} \rangle^{+}\right)\right) \tag{177}$$
$$\langle \Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^{+} \rangle^{-} = \mathsf{iter}_n * \left(\Lambda\left(\mathsf{down} * \langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+} \rangle^{-}\right)\right) \tag{178}$$

[21] Explicitly, if $\eta : F \Rightarrow G : \mathbf{A} \times \mathbf{B} \to \mathbf{C}$ is a natural transformation, then $\Lambda\eta : \Lambda F \Rightarrow \Lambda G : \mathbf{A} \to \mathbf{CAT}\left[\mathbf{B} \to \mathbf{C}\right]$ is given by $\left((\Lambda\eta)_A\right)_B = \eta_{(A,B)}$.

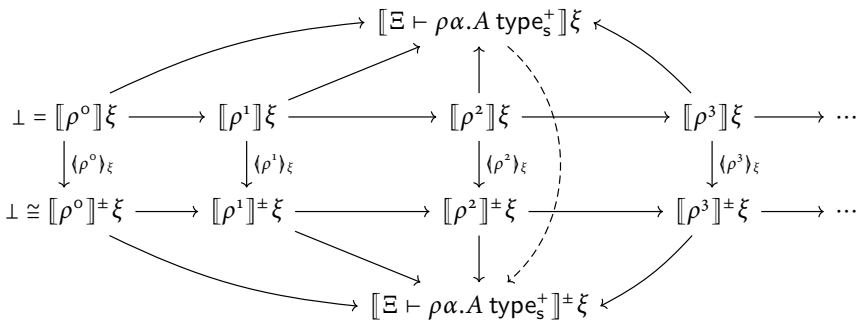


FIGURE 8.1. Colimit diagram defining the components of $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle$ as mediating morphisms of cocones

We interpret recursive types by parametrized solutions of recursive domain equations. We use the parametrized fixed point operator of section 4.5.2 to define the domains of communications.

$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!] = ((-)_\perp [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!])^\dagger \tag{179}$$

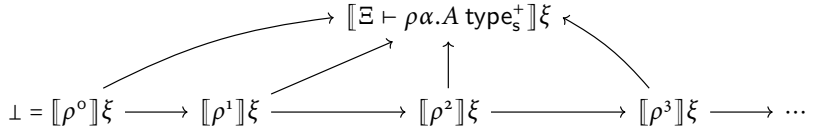
$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^+ = ((-)_\perp [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^+)^\dagger \tag{180}$$

$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^- = ([\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^-)^\dagger \tag{181}$$

$$\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^+ = ((-)_\perp \langle \Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+ \rangle^+)^\dagger \tag{182}$$

$$\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^- = (\mathsf{down} * \langle \Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+ \rangle^-)^\dagger \tag{183}$$

Informally, we can think of $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]\xi$ as "$\lim_{n\to\infty} [\![\Xi \vdash \rho^n\alpha.A\ \mathsf{type}_s^+]\!]$". Indeed, the techniques of chapter 4 defines it to be the colimiting object in the following diagram, where we abbreviate $\Xi \vdash \rho^n\alpha.A\ \mathsf{type}_s$ by ρ^n:



The elements of $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]\xi$ are elements $(x_n)_{n\in\mathbb{N}}$ of the infinite product $\prod_{n\in\mathbb{N}} [\![\rho^n]\!]\xi$ such that, where e_{mn} is the embedding $[\![\rho^n]\!] \to [\![\rho^m]\!]$ in the above ω-chain, $e^p_{mn}(x_n) = x_m$. The details are given by theorem 2.2.53. The interpretations $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^+$ and $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^-$ are similarly constructed.

Given a type forming judgment $\mathcal{J}$, abbreviate $[\![\mathcal{J}]\!]^+ \times [\![\mathcal{J}]\!]^-$ by $[\![\mathcal{J}]\!]^\pm$. We recognize the components of natural transformation $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle$ as the mediating morphism of the cocones of fig. 8.1. By construction, each component $\langle \rho^n \rangle_\xi$ is an embedding, so by proposition 2.2.63, the mediating morphism $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]\xi \to [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^\pm\xi$ is an embedding. These embeddings assemble into a natural transformation $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle : [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!] \Rightarrow [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^\pm$.

By proposition 4.3.4, there exists a canonical isomorphism

$$\mathsf{Unfold} : [\![\vdash \rho\alpha.A\ \mathsf{type}_s]\!] \to (-)_\perp \circ [\![\alpha \vdash A\ \mathsf{type}_s]\!] \circ [\![\vdash \rho\alpha.A\ \mathsf{type}_s]\!].$$

Using the substitution property (proposition 8.5.3), we recognize it as the isomorphism

$$\mathsf{Unfold} : [\![\vdash \rho\alpha.A\ \mathsf{type}_s]\!] \to (-)_\perp [\![\vdash [\rho\alpha.A/\alpha]A\ \mathsf{type}_s]\!]. \tag{184}$$

Its inverse is Fold. Similar canonical isomorphisms exist for $[\![\vdash \rho\alpha.A\ \mathsf{type}_s]\!]^+$ and $[\![\vdash \rho\alpha.A\ \mathsf{type}_s]\!]^-$.

We draw attention to the fact that, in contrast to usual presentations of isorecursive types (see, e.g., [Pie02, § 20.2]), isorecursive session types are not isomorphic to their unfolding! Indeed,

eq. (184) specifies that a recursive type is semantically equivalent to the *lifting* of its unfolding. This is because isorecursive session types contain one additional message compared to their unfolding: the "unfold" message captured by lifting. In contrast, had we used equirecursive session types, then processes would not have needed to exchange "unfold" messages, and the denotation of a recursive session type would have been equivalent to that of its unfolding.

We can express $\langle \rho\alpha.A \rangle$ in terms of $\langle [\rho\alpha.A/\alpha]A \rangle$, Fold, and Unfold. The following diagram[22] commutes by propositions 4.3.4 and 8.5.3,

$$\begin{array}{ccc} [\![\rho\alpha.A]\!] & \xrightarrow{\quad\mathsf{Unfold}\quad} & (-)_\perp[\![[\rho\alpha.A/\alpha]A]\!] \\ {\scriptstyle\langle\rho\alpha.A\rangle}\downarrow & & \downarrow{\scriptstyle\langle (-)_\perp\langle [\rho\alpha.A/\alpha]A\rangle^+, \mathsf{down} * \langle [\rho\alpha.A/\alpha]A\rangle^-\rangle} \\ [\![\rho\alpha.A]\!]^+ \times [\![\rho\alpha.A]\!]^- & \xrightarrow{\mathsf{Unfold}\times\mathsf{Unfold}} & (-)_\perp[\![[\rho\alpha.A/\alpha]A]\!]^+ \times [\![[\rho\alpha.A/\alpha]A]\!]^- \end{array} \tag{185}$$

We recognize the right morphism as the following composition, where δ^e is given by lemma 2.2.50:

$$\langle (-)_\perp\langle [\rho\alpha.A/\alpha]A\rangle^+, \mathsf{down} * \langle [\rho\alpha.A/\alpha]A\rangle^-\rangle = \delta^e \circ (-)_\perp\langle [\rho\alpha.A/\alpha]A\rangle.$$

Combining these facts, we derive eq. (186). It will be useful when reasoning about recursive types.

$$\langle \rho\alpha.A\rangle = (\mathsf{Fold} \times \mathsf{Fold}) \circ \delta^e \circ (-)_\perp\langle [\rho\alpha.A/\alpha]A\rangle \circ \mathsf{Unfold}. \tag{186}$$

Taking projections throughout, we deduce:

$$\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_\mathsf{s}^+\rangle^p = \mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A\rangle^p \circ \delta \circ (\mathsf{Unfold} \times \mathsf{Unfold}) \tag{187}$$

Processes unfold recursive types by transmitting unfold messages. Semantically, the unfold messages is captured by lifting subsequent communications. Unfolding and folding recursive types is given by pre- and post-composition with the corresponding canonical isomorphisms Fold and Unfold. We interpret $(\rho^+\mathrm{R})$ and $(\rho^+\mathrm{L})$ by:

$$\begin{aligned} &[\![\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A]\!]u \\ &= (\mathsf{id} \times (a : \mathsf{Fold} \circ \mathsf{up})) \circ [\![\Psi\ ;\ \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u \circ (\mathsf{id} \times (a^- : \mathsf{Unfold})) \end{aligned} \tag{188}$$

$$\begin{aligned} &[\![\Psi\ ;\ \Delta, a : \rho\alpha.A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-) \\ &= \begin{cases} (\delta, \mathsf{Fold}([a]), c) & \text{if } a^+ = \mathsf{Fold}([a_\mathsf{o}^+]) \\ ((\langle\Delta\rangle^p(\delta^+, \perp), \perp, \langle C\rangle^p(\perp, c^-)) & \text{otherwise} \end{cases} \\ &\quad \text{where } (\delta, a, c) = [\![\Psi\ ;\ \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C]\!]u(\delta^+, a_\mathsf{o}^+, c^-) \end{aligned} \tag{189}$$

8.4. Well-Definedness of Interpretations

> The details are all included, but as usual they are tedious and not too instructive.
>
> Larry C. Eggan [Egg21]

We show that the denotations of section 8.3 are well-defined. A general principle in the design of denotational semantics is given by the following slogan:

> The sound categorical interpretation of notion of term formation amounts to requiring that certain naturality conditions hold in the categorical model. [Cro93, p. 165]

These naturality conditions will let us easily deduce that our semantics satisfies the desired structural properties enjoyed by the language's judgments.

Recall that denotational semantics are defined compositionally, i.e., the denotation of a term is a function of the denotations of its subphrases. When working with open terms, the aforementioned naturality condition states that this function must be natural in the structural contexts appearing in the judgment.

[22] It is given for positive $\rho\alpha.A$. An analogous diagram commutes when $\rho\alpha.A$ in negative.

We illustrate this principle using interpretations of functional terms. Recall that judgments $\Psi \Vdash M : \tau$ involving processes denote stable functions $[\![\Psi]\!] \to [\![\tau]\!]$ in $\mathbf{DCPO}_\perp$. Consider a term-forming rule

$$\frac{\Psi, \Psi_1 \,;\, \Delta_1 \vdash P_1 :: c_1 : C_1 \quad \cdots \quad \Psi, \Psi_n \,;\, \Delta_n \vdash P_n :: c_n : C_n \quad \Psi, \Psi_{n+1} \Vdash M_1 : \tau_1 \quad \cdots \quad \Psi, \Psi_{n+m} \Vdash M_m : \tau_m}{\Psi \Vdash F(P_1, \ldots, P_n, M_1, \ldots, M_m) : \tau}$$

Assume that its interpretation is given by

$$\begin{aligned} &[\![\Psi \,;\, \Delta \vdash F(P_1, \ldots, P_n, M_1, \ldots, M_m) :: c : C]\!] \\ &= [\![F]\!]_{[\![\Psi]\!]} \left([\![\Psi, \Psi_1 \,;\, \Delta_1 \vdash P_1 :: c_1 : C_1]\!], \ldots, [\![\Psi, \Psi_n \,;\, \Delta_n \vdash P_n :: c_n : C_n]\!], \right. \\ &\qquad \left. [\![\Psi, \Psi_{n+1} \Vdash M_1 : \tau_1]\!], \ldots, [\![\Psi, \Psi_{n+m} \Vdash M_m : \tau_m]\!] \right), \end{aligned} \tag{190}$$

where $[\![F]\!]$ is a family of (set-theoretic) morphisms

$$\begin{aligned} [\![F]\!]_{[\![\Psi]\!]} : \left(\prod_{i=1}^{n} \mathbf{DCPO}_\perp([\![\Psi, \Psi_i]\!], \mathbf{JFC}\,[\langle \Delta_i \rangle \to \langle C_i \rangle]) \right) \times \\ \times \left(\prod_{i=1}^{m} \mathbf{DCPO}_\perp([\![\Psi, \Psi_{n+i}]\!], [\![\tau_i]\!]) \right) \to \mathbf{DCPO}([\![\Psi]\!], [\![\tau]\!]). \end{aligned} \tag{191}$$

We say that interpretation (190) is **natural in its environment** if the family (191) is natural in $[\![\Psi]\!]$. In this case, we call $[\![F]\!]$ a **natural interpretation** of the rule. The general principle requires that all interpretations be natural in their environments.

8.4.1. Semantic Results for Types. Recall that if Ξ is a context of type variables, then we write $[\![\Xi]\!]$ for the product $\prod_{\alpha \in \Xi} \mathbf{Stab}_{\perp!}$.

We start by showing that the interpretations of types in the functional layer are well defined.

Proposition 8.4.1. *If $\Xi \vdash \tau\ \mathsf{type_f}$ is purely functional, then the interpretation $[\![\Xi \vdash \tau\ \mathsf{type_f}]\!]$ is a constant and locally continuous functor from $[\![\Xi]\!]$ to $\mathbf{Stab}_{\perp!}$. If $\Xi \vdash \tau\ \mathsf{type_f}$ is impurely functional, then the interpretation $[\![\Xi \vdash \tau\ \mathsf{type_f}]\!]$ is a constant and locally continuous functor from $\prod_{\alpha \in \Xi} \mathbf{DCPO}_{\perp!}$ to $\mathbf{DCPO}_{\perp!}$.*

Proof. By induction on the derivation of $\Xi \vdash \tau\ \mathsf{type_f}$. We silently use the fact that $\mathbf{Stab}_{\perp!}$ is a subcategory of $\mathbf{DCPO}_{\perp!}$. Constant functors are locally continuous, so local continuity will follow automatically.

Case (T-$\mathbb{N}$)**:** Recall eq. (138). The flat domain of natural numbers is a dI-domain, and the functor is by definition constant.

Case (T{})**:** Recall eq. (141). This functor is by definition constant, and its image is a dcpo by corollary 8.2.23.

Case (T$\to$)**:** Recall eqs. (132) and (133). By the induction hypothesis, $[\![\Xi \vdash \tau\ \mathsf{type_f}]\!]$ and $[\![\Xi \vdash \sigma\ \mathsf{type_f}]\!]$ are both constant, so $[\![\Xi \vdash \tau \to \sigma\ \mathsf{type_f}]\!]$ is constant. In all cases, the image of $[\![\Xi \vdash \tau \to \sigma\ \mathsf{type_f}]\!]$ is a dcpo. If $\Xi \vdash \tau \to \sigma\ \mathsf{type_f}$ is purely functional, then by the induction hypothesis, $[\![\Xi \vdash \tau\ \mathsf{type_f}]\!]$ and $[\![\Xi \vdash \sigma\ \mathsf{type_f}]\!]$ are both functors into $\mathbf{Stab}_{\perp!}$. It follows that $[\![\Xi \vdash \tau \to \sigma\ \mathsf{type_f}]\!]$ is also a functor into $\mathbf{Stab}_{\perp!}$. □

Recall that we interpret judgments $\Xi \vdash A\ \mathsf{type_s}$ as 2-cells

$$\langle \Xi \vdash A\ \mathsf{type_s} \rangle : [\![\Xi \vdash A\ \mathsf{type_s}]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type_s}]\!]^- \times [\![\Xi \vdash A\ \mathsf{type_s}]\!]^+ : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}$$

in the 2-category **CFP** defined in section 4.5.2.

We begin by showing that the functors interpreting types are locally continuous.

Proposition 8.4.2 (Functorial Interpretations are Well-Defined). *If $\Xi \vdash A\ \mathsf{type_s}$, then the interpretations $[\![\Xi \vdash A\ \mathsf{type_s}]\!]$, $[\![\Xi \vdash A\ \mathsf{type_s}]\!]^-$, and $[\![\Xi \vdash A\ \mathsf{type_s}]\!]^+$ are functors from $[\![\Xi]\!]$ to $\mathbf{Stab}_{\perp!}$. They are locally continuous relative to the stable ordering.*

Proof. By induction on the derivations of $\Xi \vdash A\ \mathsf{type}_s$. By the simplifying assumptions of section 8.1.1, the interpretations $[\![\Xi \vdash \tau\ \mathsf{type}_f]\!]$ are constant functors, so they are automatically locally continuous.

Case (C**1**)**:** Recall eqs. (116) to (117). Constant functors are locally continuous. Their images are obviously dI-domains.

Case (CVar)**:** Recall eqs. (168) to (169). The projection functors are locally continuous, and their codomains are assumed to be dI-domains.

Case (Cρ_n^+)**:** Recall eqs. (174) to (176). By the induction hypothesis, local continuity of Λ, the obvious specialization of proposition 4.5.1, and the fact that locally continuous functors are closed under composition.

Case (Cρ^+)**:** Recall eqs. (179) to (181). The category $\mathbf{Stab}_{\perp!}$ is a **CFP** category: its initial object is $\{\perp\}$, its morphisms are strict, and it is **O**-cocomplete. Parametrized fixed points of locally continuous functors are then locally continuous by the results of section 4.5.2.

Case (C$\wedge$)**:** Recall eqs. (144) to (145). By proposition 8.4.1 and the simplifying assumptions of section 8.1.1, $\Xi \vdash \tau\ \mathsf{type}_f$ is a locally continuous functor from $[\![\Xi]\!]$ to $\mathbf{Stab}_{\perp!}$. The result follows from the fact that locally continuous functors are closed under composition.

Local continuity in the remaining cases follow either by analogy with one of the above cases, or from the observation that they are compositions of locally continuous functors and that local continuity is closed under composition. The fact that their codomain is $\mathbf{Stab}_{\perp!}$ follows from the fact that $\mathbf{Stab}_{\perp!}$ is closed under lifting, coalesced products, and coalesced sums. □

Next, we show that the 2-cells $\langle \Xi \vdash A\ \mathsf{type}_s \rangle$ are families of stable maps, and that each component is an embedding relative to the pointwise ordering.

Proposition 8.4.3 (Types Denote 2-Cells). *If* $\Xi \vdash A\ \mathsf{type}_s$*, then*

$$\langle \Xi \vdash A\ \mathsf{type}_s \rangle : [\![\Xi \vdash A\ \mathsf{type}_s]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type}_s]\!]^+ \times [\![\Xi \vdash A\ \mathsf{type}_s]\!]^- : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}$$

is a natural transformation.

Proof. By induction on the derivation of $\Xi \vdash A\ \mathsf{type}_s$. We omit cases that follow by analogy from others. We sometimes abuse notation and write $\eta \cdot \rho$ for the component-wise composition of families η and ρ, even when they are not natural transformations.

The majority of cases, naturality follows from the induction hypothesis and the following three facts:

- natural transformations are closed under composition,
- functors preserve natural transformations, and
- the pairing of two natural transformations is a natural transformation.

Stability follows from the induction hypothesis and the fact that $\mathbf{Stab}_{\perp!}$ is closed under lifting, products, pairing, coalesced sums, and smash products. We omit these cases.

Case (C**1**)**:** Recall eqs. (119) and (120). The constant family $\langle \Xi \vdash \mathbf{1}\ \mathsf{type}_s \rangle$ of morphisms between constant functors is clearly natural, and constant functions are stable.

Case (CVar)**:** Recall eqs. (171) and (172). The family $\langle \Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p \rangle$ is clearly natural, and each component is clearly stable.

Case (Cρ_n^+)**:** Recall eqs. (177) and (178). By the induction hypothesis and the fact that functors preserve commuting diagrams.

Case (Cρ^+)**:** Recall eqs. (182) and (183). To see that $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle$ is natural, observe that

$$\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle = \langle \langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^+, \langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^- \rangle,$$

and that $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^+$ and $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^-$ are natural by proposition 4.3.1. □

The proof that $\langle \Xi \vdash A\ \mathsf{type}_s \rangle$ is a family of embeddings is most easily shown using a substitution property. This substitution property relies on the fact that these 2-cells satisfy the appropriate naturality conditions. We adapt the overview given at the start of section 8.4 to the setting of type interpretations, on account of their complexity in this setting. Recall that for every 2-category **C**,

there exists a category $\mathbf{Cell}_{\mathbf{C}}$ whose objects are the objects of $\mathbf{C}$, whose morphisms are the 2-cells of $\mathbf{C}$, and whose composition is the horizontal composition of $\mathbf{C}$. Consider a type-forming rule

$$\frac{\Xi, \Xi_1 \vdash A_1\ \mathsf{type_s} \quad \cdots \quad \Xi, \Xi_n \vdash A_n\ \mathsf{type_s}}{\Xi \vdash F(A_1, \ldots, A_n)\ \mathsf{type_s}}$$

Assume that its interpretation is given by

$$\begin{aligned} &\langle \Xi \vdash F(A_1, \ldots, A_n)\ \mathsf{type_s} \rangle \\ &= \ulcorner F \urcorner_{[\![\Xi]\!]} \left(\langle \Xi, \Xi_1 \vdash A_1\ \mathsf{type_s} \rangle, \ldots, \langle \Xi, \Xi_n \vdash A_n\ \mathsf{type_s} \rangle \right). \end{aligned} \tag{192}$$

where $\ulcorner F \urcorner$ is a $[\![\Xi]\!]$-indexed family of morphisms

$$\ulcorner F \urcorner_{[\![\Xi]\!]} : \left(\prod_{i=1}^{n} \mathbf{Cell}_{\mathbf{CFP}}([\![\Xi, \Xi_i]\!], \mathbf{Stab}_{\perp!}) \right) \to \mathbf{Cell}_{\mathbf{CFP}}([\![\Xi]\!], \mathbf{Stab}_{\perp!}). \tag{193}$$

Equation (192) is **natural in its environment** if the family $\ulcorner F \urcorner_{[\![\Xi]\!]}$ is natural in $[\![\Xi]\!]$, i.e., if for all 2-cells $\sigma : \hat{\sigma} \Rightarrow \check{\sigma} : \mathbf{C} \to \mathbf{D}$, the following diagram commutes in **Set**:

$$\begin{array}{ccc} \prod_{i=1}^{n} \mathbf{Cell}_{\mathbf{CFP}}(\mathbf{D} \times [\![\Xi_i]\!], \mathbf{Stab}_{\perp!}) & \xrightarrow{\ulcorner F \urcorner_{\mathbf{D}}} & \mathbf{Cell}_{\mathbf{CFP}}(\mathbf{D}, \mathbf{Stab}_{\perp!}) \\ \downarrow{\scriptstyle \prod_{i=1}^{n} \mathbf{Cell}_{\mathbf{CFP}}(\sigma \times [\![\Xi_i]\!], \mathbf{Stab}_{\perp!})} & & \downarrow{\scriptstyle \mathbf{Cell}_{\mathbf{CFP}}(\sigma, \mathbf{Stab}_{\perp!})} \\ \prod_{i=1}^{n} \mathbf{Cell}_{\mathbf{CFP}}(\mathbf{C} \times [\![\Xi_i]\!], \mathbf{Stab}_{\perp!}) & \xrightarrow{\ulcorner F \urcorner_{\mathbf{C}}} & \mathbf{Cell}_{\mathbf{CFP}}(\mathbf{C}, \mathbf{Stab}_{\perp!}) \end{array}$$

Concretely, this means that for all n-tuples of 2-cells $(\alpha_i : \hat{\alpha}_i \Rightarrow \check{\alpha}_i : \mathbf{D} \to \mathbf{Stab}_{\perp!})_{1 \le i \le n}$,

$$\ulcorner F \urcorner_{\mathbf{C}} \left((\alpha_i * (\sigma \times [\![\Xi_i]\!]) : \hat{\alpha}_i \circ (\hat{\sigma} \times [\![\Xi_i]\!]) \Rightarrow \check{\alpha}_i \circ (\check{\sigma} \times [\![\Xi_i]\!]))_{1 \le i \le n} \right) = \ulcorner F \urcorner_{\mathbf{D}} \left((\alpha_i)_{1 \le i \le n} \right) * \sigma$$

Proposition 8.4.4. *If* $\Xi \vdash A\ \mathsf{type_s}$*, then the interpretations*

$$\langle \Xi \vdash A\ \mathsf{type_s} \rangle^{+} : [\![\Xi \vdash A\ \mathsf{type_s}]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type_s}]\!]^{+} : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!},$$
$$\langle \Xi \vdash A\ \mathsf{type_s} \rangle^{-} : [\![\Xi \vdash A\ \mathsf{type_s}]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type_s}]\!]^{-} : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}$$

are natural in their environment.

Proof. By case analysis on the last rule in the derivation of $\Xi \vdash A\ \mathsf{type_s}$. We omit cases that follow easily by duality. In most cases, the given natural transformations are clearly the desired interpretations.

Case (C**1**)**:** Recall eqs. (116) to (120). We show the positive case; the negative case is analogous. The rule has no hypotheses, so we must show that there exists a family of morphisms

$$\eta_{\mathbf{C}} : \mathrm{diag}_{\mathbf{Cell}_{\mathbf{CFP}}} \top_{\mathbf{Set}} \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(\mathbf{C}, \mathbf{Stab}_{\perp!})$$

natural in $\mathbf{C}$ such that

$$\langle \Xi \vdash \mathbf{1}\ \mathsf{type_s^+} \rangle^{+} = \eta_{[\![\Xi]\!]}(*)$$

The family we seek is the constant family

$$\eta_{\mathbf{C}}(*) = \mathsf{id},$$

and this family is obviously natural.

Case (CVar)**:** Recall eqs. (119), (120) and (168) to (170). We show the positive case; the negative case is identical. The rule has no hypotheses, so we must show that there exists a family of morphisms

$$\eta_{\mathbf{C}} : \mathrm{diag}_{\mathbf{Cell}_{\mathbf{CFP}}}(\top_{\mathbf{Set}}) \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(\mathbf{C} \times [\![\alpha]\!], \mathbf{Stab}_{\perp!})$$

natural in $\mathbf{C}$ such that

$$\langle \Xi, \alpha\ \mathsf{type_s^p} \vdash \alpha\ \mathsf{type_s^p} \rangle^{+} = \eta_{[\![\Xi]\!]}(*).$$

The obvious choice is

$$\eta_{\mathbf{C}}(*) = \mathsf{id} : \pi_\alpha \Rightarrow \pi_\alpha$$

and it is obviously natural.

Case ($C\rho_n^+$)**:** Recall eqs. (174) to (178). We show the positive case. We must show that there exists a natural interpretation

$$\eta^+ : \mathbf{Cell}_{\mathbf{CFP}}(- \times [\![\alpha]\!], \mathbf{Stab}_{\perp!}) \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!})$$

such that

$$\eta^+_{[\![\Xi]\!]}(\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^+ \vdash A\ \mathsf{type}_{\mathsf{s}}^+ \rangle^+) = \langle \Xi \vdash \rho^n \alpha.A\ \mathsf{type}_{\mathsf{s}}^+ \rangle^+.$$

Take

$$\eta^+_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \times [\![\alpha]\!] \to \mathbf{Stab}_{\perp!}) = \mathsf{iter}_n * (\Lambda((-)_\perp \sigma)).$$

To show the naturality of η^+, we must show that for all 2-cells $\nu : H \Rightarrow I : \mathbf{D} \to \mathbf{C}$,

$$\mathsf{iter}_n * (\Lambda((-)_\perp(\sigma * (\nu \times \mathsf{id})))) = (\mathsf{iter}_n * (\Lambda((-)_\perp \sigma))) * \nu.$$

But this is immediate by naturality of Λ. The natural interpretation in the negative case is analogous. Explicitly, it is the natural transformation

$$\eta^- : \mathbf{Cell}_{\mathbf{CFP}}(- \times [\![\alpha]\!], \mathbf{Stab}_{\perp!}) \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!})$$

given by

$$\eta^-_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \times [\![\alpha]\!] \to \mathbf{Stab}_{\perp!}) = \mathsf{iter}_n * (\Lambda\,(\mathsf{down} * \sigma))\,.$$

Case ($C\rho^+$)**:** Recall eqs. (179) to (183). We start with the positive case. We must show that there exists a natural interpretation

$$\eta^+ : \mathbf{Cell}_{\mathbf{CFP}}(- \times [\![\alpha]\!], \mathbf{Stab}_{\perp!}) \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!})$$

such that

$$\eta^+_{[\![\Xi]\!]}(\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^+ \vdash A\ \mathsf{type}_{\mathsf{s}}^+ \rangle^+) = \langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_{\mathsf{s}}^+ \rangle^+.$$

Take

$$\eta^+_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \times [\![\alpha]\!] \to \mathbf{Stab}_{\perp!}) = ((-)_\perp \sigma)^\dagger : ((-)_\perp F)^\dagger \Rightarrow ((-)_\perp G)^\dagger : \mathbf{C} \to \mathbf{Stab}_{\perp!}.$$

To show the naturality of η^+, we must show that for all 2-cells $\nu : H \Rightarrow I : \mathbf{D} \to \mathbf{C}$,

$$\begin{aligned}&(((-)_\perp \sigma) * (\nu \times \mathsf{id}))^\dagger : (((-)_\perp F) \circ (H \times \mathsf{id}))^\dagger \Rightarrow (((-)_\perp G) \circ (I \times \mathsf{id}))^\dagger : \mathbf{D} \to \mathbf{Stab}_{\perp!} \\ &= ((-)_\perp \sigma)^\dagger * \nu : ((-)_\perp F)^\dagger H \Rightarrow ((-)_\perp G)^\dagger I : \mathbf{D} \to \mathbf{Stab}_{\perp!}.\end{aligned}$$

This is exactly the parameter identity given by corollary 4.3.6. The negative case is analogous. The natural interpretation is

$$\eta^-_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \times [\![\alpha]\!] \to \mathbf{Stab}_{\perp!}) = (\mathsf{down} * \sigma)^\dagger : ((-)_\perp F)^\dagger \Rightarrow G^\dagger : \mathbf{C} \to \mathbf{Stab}_{\perp!}.$$

Naturality means that we must show that for all 2-cells $\nu : H \Rightarrow I : \mathbf{D} \to \mathbf{C}$,

$$\begin{aligned}&((\mathsf{down} * \sigma) * (\nu \times \mathsf{id}))^\dagger : (((-)_\perp F) \circ (H \times \mathsf{id}))^\dagger \Rightarrow (G \circ (I \times \mathsf{id}))^\dagger : \mathbf{D} \to \mathbf{Stab}_{\perp!} \\ &= (\mathsf{down} * \sigma)^\dagger * \nu : ((-)_\perp F)^\dagger H \Rightarrow G^\dagger I : \mathbf{D} \to \mathbf{Stab}_{\perp!}.\end{aligned}$$

It too follows from corollary 4.3.6.

Case (C↓)**:** Recall eqs. (152) to (156). We must show that there exist natural transformations

$$\eta^+, \eta- : \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!}) \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!})$$

that are the respective natural interpretations. Take

$$\begin{aligned}\eta^-_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!}) &= \mathsf{down} * \sigma : (-)_\perp F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!} \\ \eta^+_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!}) &= (-)_\perp \sigma : (-)_\perp F \Rightarrow (-)_\perp G : \mathbf{C} \to \mathbf{Stab}_{\perp!}\end{aligned}$$

To show the naturality of η^- and η^+, we must show for all 2-cells $\nu : H \Rightarrow I : \mathbf{D} \to \mathbf{C}$ that

$$\begin{aligned}\eta^-_{\mathbf{D}}(\sigma * \nu : FH \Rightarrow GI : \mathbf{D} \to \mathbf{Stab}_{\perp!}) &= \eta^-_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!}) * \nu \\ \eta^+_{\mathbf{D}}(\sigma * \nu : FH \Rightarrow GI : \mathbf{D} \to \mathbf{Stab}_{\perp!}) &= \eta^+_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!}) * \nu.\end{aligned}$$

In the negative case, we have by associativity of composition:

$$\begin{aligned}
&\eta^-_{\mathbf{D}}(\sigma * \nu : FH \Rightarrow GI : \mathbf{D} \to \mathbf{Stab}_{\perp!}) \\
&= \mathsf{down} * (\sigma * \nu) : (-)_\perp(FH) \Rightarrow GI : \mathbf{D} \to \mathbf{Stab}_{\perp!} \\
&= (\mathsf{down} * \sigma) * \nu : ((-)_\perp F)H \Rightarrow GI : \mathbf{D} \to \mathbf{Stab}_{\perp!} \\
&= \eta^-_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!}) * \nu.
\end{aligned}$$

In the positive case,

$$\begin{aligned}
&\eta^+_{\mathbf{D}}(\sigma * \nu : FH \Rightarrow GI : \mathbf{D} \to \mathbf{Stab}_{\perp!}) \\
&= (-)_\perp(\sigma * \nu) : (-)_\perp(FH) \Rightarrow (-)_\perp(GI) : \mathbf{D} \to \mathbf{Stab}_{\perp!} \\
&= ((-)_\perp\sigma) * \nu : ((-)_\perp F)H \Rightarrow ((-)_\perp G)I : \mathbf{D} \to \mathbf{Stab}_{\perp!} \\
&= \eta^+_{\mathbf{C}}(\sigma : F \Rightarrow G : \mathbf{C} \to \mathbf{Stab}_{\perp!}) * \nu.
\end{aligned}$$

Case (C⊕)**:** Recall eqs. (160) to (164). We must show that there exist natural transformations

$$\eta^+, \eta^- : \left(\prod_{l\in L} \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!})\right) \Rightarrow \mathbf{Cell}_{\mathbf{CFP}}(-, \mathbf{Stab}_{\perp!})$$

that are the respective natural interpretations. Take

$$\begin{aligned}
&\eta^-_{\mathbf{C}}\left((\sigma_l : F_l \Rightarrow G_l : \mathbf{C} \to \mathbf{Stab}_{\perp!})_{l\in L}\right) \\
&= \mathrm{diag}\,(\mathsf{down} * \sigma_l)_{l\in L} : \bigoplus_{l\in L}(-)_\perp F_l \Rightarrow \prod_{l\in L} G_l : \mathbf{C} \to \mathbf{Stab}_{\perp!} \\
&\eta^+_{\mathbf{C}}\left((\sigma_l : F_l \Rightarrow G_l : \mathbf{C} \to \mathbf{Stab}_{\perp!})_{l\in L}\right) \\
&= \bigoplus_{l\in L}(-)_\perp\sigma_l : \bigoplus_{l\in L}(-)_\perp F_l \Rightarrow \bigoplus_{l\in L}(-)_\perp G_l : \mathbf{C} \to \mathbf{Stab}_{\perp!}.
\end{aligned}$$

The family η^+ is natural by associativity of composition. To show that η^- is natural, we must show that for any 2-cell $\nu : H \Rightarrow I : \mathbf{D} \to \mathbf{C}$,

$$\begin{aligned}
&\mathrm{diag}\,(\mathsf{down} * \sigma_l * \nu)_{l\in L} : \bigoplus_{l\in L}(-)_\perp(F_l H) \Rightarrow \prod_{l\in L}(G_l I) : \mathbf{C} \to \mathbf{Stab}_{\perp!} \\
&= \mathrm{diag}\,(\mathsf{down} * \sigma_l)_{l\in L} * \nu : \left(\bigoplus_{l\in L}(-)_\perp F_l\right) H \Rightarrow \left(\prod_{l\in L} G_l\right) I : \mathbf{C} \to \mathbf{Stab}_{\perp!}.
\end{aligned}$$

The sources and targets of these vertical morphisms are equal by associativity of functorial composition. We show that the families are equal. Let D be arbitrary in $\mathbf{D}$, then

$$\left(\mathrm{diag}\,(\mathsf{down} * \sigma_l * \nu)_{l\in L}\right)_D : \bigoplus_{l\in L}(-)_\perp(F_l H)D \Rightarrow \prod_{l\in L}(G_l I)D$$

is the mediating morphism in $\mathbf{Stab}_{\perp!}$ given by the coproduct:

$$\begin{array}{ccc}
(-)_\perp F_l HD & \xrightarrow{\quad (\mathsf{down}*\sigma_l*\nu)_D \quad} & G_l ID \\
\big\downarrow & & \big\uparrow{\scriptstyle \pi_l} \\
\oplus_{l\in L}(-)_\perp F_l HD & \xrightarrow{\quad \left(\mathrm{diag}(\mathsf{down}*\sigma_l*\nu)_{l\in L}\right)_D \quad} & \prod_{l\in L} G_l ID \\
\big\uparrow & & \big\downarrow{\scriptstyle \pi_k} \\
(-)_\perp F_l HD & \xrightarrow{\quad \text{o when } l\neq k \quad} & G_k ID
\end{array} \tag{194}$$

Expanding the definition of horizontal composition in the top morphism of diagram 194, we get the top morphism of diagram 195, below. By definition of zero morphism, the bottom morphisms

of the two diagrams are also equal. So their perimeters are equal.

$$\begin{array}{ccccc}
(-)_\perp F_l HD & \xrightarrow{(\mathsf{down}*\sigma_l)_{HD}} & G_l HD & \xrightarrow{(G_l v)_D} & G_l ID \\
\downarrow & & \uparrow \pi_l & & \uparrow \pi_l \\
\bigoplus_{l\in L}(-)_\perp F_l HD & \xrightarrow{(\mathrm{diag}(\mathsf{down}*\sigma_l)_{l\in L})_{HD}} & \prod_{l\in L} G_l HD & \xrightarrow{\prod_{l\in L}(G_l v)_D} & \prod_{l\in L} G_l ID \\
\uparrow & & \downarrow \pi_k & & \downarrow \pi_k \\
(-)_\perp F_l HD & \xrightarrow{\text{o when } l\neq k} & G_k HD & \xrightarrow{(G_k v)_D} & G_k ID
\end{array} \tag{195}$$

We recognize the composition of the mediating morphisms in the centre of diagram 195 as

$$\begin{aligned}
&\left(\prod_{l\in L}(G_l v)_D\right)\circ\left(\mathrm{diag}\left(\mathsf{down}*\sigma_l\right)_{l\in L}\right)_{HD}\\
&\left(\prod_{l\in L}G_l(v_D)\right)\circ\left(\mathrm{diag}\left(\mathsf{down}*\sigma_l\right)_{l\in L}\right)_{HD}\\
&=\left(\prod_{l\in L}G_l\right)v_D\circ\left(\mathrm{diag}\left(\mathsf{down}*\sigma_l\right)_{l\in L}\right)_{HD}\\
&=\left(\mathrm{diag}\left(\mathsf{down}*\sigma_l\right)_{l\in L}*v\right)_D.
\end{aligned}$$

These are both mediating morphisms making the same coproduct diagram commute. By uniqueness of mediating morphisms, we conclude that they must be equal. Because D was an arbitrary component, we conclude that $\mathrm{diag}(\mathsf{down}*\sigma_l*v)_{l\in L}$ and $\mathrm{diag}(\mathsf{down}*\sigma_l)_{l\in L}*v$ are equal natural transformations, i.e., that η^- is a natural transformation.

Case (C⊗)**:** Recall eqs. (124) to (128). We must show that there exist natural transformations

$$\eta^+,\eta^- : \mathbf{Cell}_{\mathbf{CFP}}(-,\mathbf{Stab}_{\perp!})\times\mathbf{Cell}_{\mathbf{CFP}}(-,\mathbf{Stab}_{\perp!})\Rightarrow\mathbf{Cell}_{\mathbf{CFP}}(-,\mathbf{Stab}_{\perp!})$$

that are the respective natural interpretations. Take

$$\begin{aligned}
\eta^-_{\mathbf{C}}(\alpha:A\Rightarrow C,\beta:B\Rightarrow D) &= \mathsf{down}*(\alpha\times\beta):(A\times B)_\perp\Rightarrow C\times D:\mathbf{C}\to\mathbf{Stab}_{\perp!}\\
\eta^+_{\mathbf{C}}(\alpha:A\Rightarrow C,\beta:B\Rightarrow D) &= (\alpha\times\beta)_\perp:(A\times B)_\perp\Rightarrow(C\times D)_\perp:\mathbf{C}\to\mathbf{Stab}_{\perp!}.
\end{aligned}$$

We use the definition of products in $\mathbf{Cell}_{\mathbf{Stab}_{\perp!}}$ and associativity of composition to show naturality. Then:

$$\begin{aligned}
&\eta^-_{\mathbf{C}}(\alpha:A\Rightarrow C,\beta:B\Rightarrow D)*v\\
&=\mathsf{down}*(\alpha\times\beta)*v\\
&=\mathsf{down}*((\alpha*v)\times(\beta*v))\\
&=\eta^-_{\mathbf{D}}(\alpha*v:AH\Rightarrow CI,\beta*v:BH\Rightarrow DI).
\end{aligned}$$

We conclude that η^- is natural. A similar argument gives that η^+ is natural.

Case (C∧)**:** Recall eqs. (144) to (148). We must show that there exist natural transformations

$$\eta^+,\eta^- : \mathbf{Cell}_{\mathbf{Stab}_{\perp!}}(-,\mathbf{Stab}_{\perp!})\Rightarrow\mathbf{Cell}_{\mathbf{Stab}_{\perp!}}(-,\mathbf{Stab}_{\perp!})$$

that are the respective natural interpretations. Take

$$\begin{aligned}
\eta^-_{\mathbf{C}}(\sigma:F\Rightarrow G) &= \mathsf{down}*\pi_2*\sigma:(\llbracket\tau\rrbracket\times F)_\perp\Rightarrow F:\mathbf{C}\to\mathbf{Stab}_{\perp!}\\
\eta^+_{\mathbf{C}}(\sigma:F\Rightarrow G) &= (\llbracket\tau\rrbracket\times\sigma)_\perp:(\llbracket\tau\rrbracket\times F)_\perp\Rightarrow(\llbracket\tau\rrbracket\times G)_\perp:\mathbf{C}\to\mathbf{Stab}_{\perp!}
\end{aligned}$$

These are natural by the associativity of composition of functors. □

The proof of proposition 8.4.6 relies on the substitution property, which is given by proposition 8.5.3 below. Proposition 8.5.3 is an easy corollary of proposition 8.4.6, and no circularity is introduced between these three results. In the interest in a thematic presentation of our results,

we hope the reader will forgive our use of forward references. We encapsulate our use of the substitution property in the following lemma:

LEMMA 8.4.5. *If* $\Xi \vdash \rho^{n+1}\alpha.A\ \mathsf{type}_{\mathsf{s}}^{+}$, *then*

$$\langle \Xi \vdash \rho^{n+1}\alpha.A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+} = (-)_{\perp}\langle \Xi \vdash [\rho^{n}\alpha.A/\alpha]A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+},$$
$$\langle \Xi \vdash \rho^{n+1}\alpha.A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{-} = \mathsf{down} * \langle \Xi \vdash [\rho^{n}\alpha.A/\alpha]A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{-}.$$

The result is symmetric when $\Xi \vdash \rho^{n+1}\alpha.A\ \mathsf{type}_{\mathsf{s}}^{-}$.

Proof. By induction on n. Recall eqs. (177) and (178).

CASE $n = 0$**:** We show the positive case. Observe that $(\mathsf{iter}_0(\eta) = \mathsf{id}_{\perp}$ for all $\eta : F \Rightarrow G : \mathbf{Stab}_{\perp!} \to \mathbf{Stab}_{\perp!}$. Assume that $\Xi = \alpha_1\ \mathsf{type}_{\mathsf{s}}, \ldots, \alpha_m\ \mathsf{type}_{\mathsf{s}}$, and consider some arbitrary component $\xi = (\xi_1, \ldots, \xi_m) \in [\![\Xi]\!]$. We compute using eqs. (171) and (177):

$$\begin{aligned}
&\langle \Xi \vdash \rho^{1}\alpha.A\ \mathsf{type}_{\mathsf{s}}\rangle^{+}_{\xi} \\
&= (\mathsf{iter}_1 * (\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+})))_{\xi} \\
&= (-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}_{\xi,\perp} \\
&= ((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+} * \langle \pi_{\alpha_1}, \ldots, \pi_{\alpha_n}, [\![\Xi \vdash \rho^{0}\alpha.A\ \mathsf{type}_{\mathsf{s}}]\!]\rangle)_{\xi} \\
&= ((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+} * \\
&\qquad * \langle \langle \Xi \vdash \alpha_1\ \mathsf{type}_{\mathsf{s}}\rangle^{+}, \ldots, \langle \Xi \vdash \alpha_m\ \mathsf{type}_{\mathsf{s}}\rangle^{+}, \langle \Xi \vdash \rho^{0}\alpha.A\ \mathsf{type}_{\mathsf{s}}\rangle^{+}\rangle)_{\xi},
\end{aligned}$$

which by proposition 8.5.3:

$$\begin{aligned}
&= ((-)_{\perp}\langle \Xi \vdash [\vec{\alpha}_i, \rho^{0}\alpha.A/\vec{\alpha}_i, \alpha]A\ \mathsf{type}_{\mathsf{s}}\rangle)_{\xi} \\
&= ((-)_{\perp}\langle \Xi \vdash [\rho^{0}\alpha.A/\alpha]A\ \mathsf{type}_{\mathsf{s}}\rangle)_{\xi}.
\end{aligned}$$

The negative case is analogous.

CASE $n = k + 1$**:** Assume the result holds for k. We show the positive case. Consider some arbitrary component ξ:

$$\begin{aligned}
&\langle \Xi \vdash \rho^{k+1}\alpha.A\ \mathsf{type}_{\mathsf{s}}\rangle^{+}_{\xi} \\
&= (\mathsf{iter}_{k+1} * (\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+})))_{\xi} \\
&= \left((\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}))_{\xi}\right)^{(k+1)}_{\perp} \\
&= \left((\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}))_{\xi} * \left((\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}))_{\xi}\right)^{(k)}\right)_{\perp} \\
&= \left((\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}))_{\xi}\right)_{[\![\Xi \vdash \rho^{k}\alpha.A\ \mathsf{type}_{\mathsf{s}}]\!]^{+}\xi} \circ \\
&\qquad \circ \left((\Lambda((-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}))_{\xi}\right)^{(k)}_{\perp} \\
&= (-)_{\perp}\langle \Xi, \alpha\ \mathsf{type}_{\mathsf{s}}^{+} \vdash A\ \mathsf{type}_{\mathsf{s}}^{+}\rangle^{+}_{\xi,[\![\Xi \vdash \rho^{k}\alpha.A\ \mathsf{type}_{\mathsf{s}}]\!]^{+}\xi} \circ \langle \Xi \vdash \rho^{k}A.\ \mathsf{type}_{\mathsf{s}}\rangle^{+}_{\xi}
\end{aligned}$$

which by an argument using eq. (171) and proposition 8.5.3 similar to the one in the base case:

$$= (-)_{\perp}\langle \Xi \vdash [\Xi \vdash \rho^{k}A.\ \mathsf{type}_{\mathsf{s}}/\alpha]A\ \mathsf{type}_{\mathsf{s}}\rangle_{\xi}.$$

Again, the negative case is analogous. □

PROPOSITION 8.4.6 (Natural Family of Embeddings). *If* $\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}$, *then*

$$\langle \Xi \vdash A\ \mathsf{type}_{\mathsf{s}}\rangle : [\![\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}]\!] \Rightarrow [\![\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}]\!]^{+} \times [\![\Xi \vdash A\ \mathsf{type}_{\mathsf{s}}]\!]^{-} : [\![\Xi]\!] \to \mathbf{Stab}_{\perp!}$$

is a family of continuous embeddings relative to the pointwise order.

Proof. It is sufficient to show that $\langle \Xi \vdash A\ \mathsf{type}_{\mathsf{s}}\rangle$ is a family of embeddings: lower adjoints are continuous by [AJ95, Proposition 3.1.14].

We use a well-founded induction on the set of open session types, ordered by a relation that captures their recursive structure. In the general recursive case, this will let us use a lemma that

holds for the related bounded recursive types. We remark that an induction on the derivation would not permit us to use this lemma, because bounded recursive types are not structurally smaller than general recursive types. Additionally, we highlight the fact that care must be taken in constructing the binary relation that orders session types, so as to not introduce any infinite descending chains.

Concretely, consider the set T of well-formed open session types $\Xi \vdash A\ \mathsf{type_s}$, ordered by the least transitive relation generated by the following rules:

(1) whenever $\Xi_1 \vdash A_1\ \mathsf{type_s}$ is a premise to a rule with conclusion $\Xi_2 \vdash A_2\ \mathsf{type_s}$,

$$\frac{}{(\Xi_1 \vdash A_1\ \mathsf{type_s}) \prec (\Xi_2 \vdash A_2\ \mathsf{type_s})}$$

(2) for all n,

$$\frac{}{(\Xi \vdash \rho^n \alpha.A\ \mathsf{type_s}) \prec (\Xi \vdash \rho\alpha.A\ \mathsf{type_s})}$$

(3) for all n,

$$\frac{}{(\Xi \vdash [\rho^n \alpha.A/\alpha]A\ \mathsf{type_s}) \prec (\Xi \vdash \rho^{n+1}\alpha.A\ \mathsf{type_s})}$$

This ordering has no infinite descending chains, so it is well-founded by the axiom of dependent choice.

We proceed by well-founded induction on T. We omit cases that follow by analogy from others. We show that there exists a corresponding family[23] of projections $\langle \Xi \vdash A\ \mathsf{type_s}\rangle^p$ such that

$$\langle \Xi \vdash A\ \mathsf{type_s}\rangle^p_\xi \circ \langle \Xi \vdash A\ \mathsf{type_s}\rangle_\xi = \mathsf{id}_{[\![\Xi \vdash A\ \mathsf{type_s}]\!]\xi}$$

and

$$\langle \Xi \vdash A\ \mathsf{type_s}\rangle_\xi \circ \langle \Xi \vdash A\ \mathsf{type_s}\rangle^p_\xi \sqsubseteq \mathsf{id}_{([\![\Xi \vdash A\ \mathsf{type_s}]\!]^+ \times [\![\Xi \vdash A\ \mathsf{type_s}]\!]^-)\xi}$$

for all components ξ. We sometimes abuse notation and write $\eta \cdot \rho$ for the component-wise composition of families η and ρ, even when they are not natural transformations. Where it improves legibility, we may abbreviate $\Xi \vdash A\ \mathsf{type_s}$ by A and use the abbreviation:

$$[\![\Xi \vdash A\ \mathsf{type_s}]\!]^\pm = [\![\Xi \vdash A\ \mathsf{type_s}]\!]^+ \times [\![\Xi \vdash A\ \mathsf{type_s}]\!]^-.$$

Case (C$\mathbf{1}$): Recall eqs. (119) to (121). The constant family $\langle \Xi \vdash \mathbf{1}\ \mathsf{type_s}\rangle$ of morphisms between constant functors is clearly natural. Because $[\![\Xi \vdash \mathbf{1}\ \mathsf{type_s^+}]\!] = [\![\Xi \vdash \mathbf{1}\ \mathsf{type_s^+}]\!]^+$, we compute:

$$\langle \Xi \vdash \mathbf{1}\ \mathsf{type_s^+}\rangle^p \cdot \langle \Xi \vdash \mathbf{1}\ \mathsf{type_s^+}\rangle = \pi_1 \cdot \langle \mathsf{id}, \top\rangle = \mathsf{id}.$$

Next consider some arbitrary component ξ and element

$$(x^-, x^+) \in [\![\Xi \vdash \mathbf{1}\ \mathsf{type_s}]\!]^\pm \xi,$$

then

$$\left(\langle \Xi \vdash \mathbf{1}\ \mathsf{type_s^+}\rangle_\xi \circ \langle \Xi \vdash \mathbf{1}\ \mathsf{type_s^+}\rangle^p_\xi\right)(x^-, x^+) = \langle \mathbf{1}\rangle(x^+) = (\bot, x^+) \sqsubseteq (x^-, x^+).$$

The components are all clearly stable and strict.

Case (CVar): Recall eqs. (171) to (173). The family $\langle \Xi, \alpha\ \mathsf{type_s^p} \vdash \alpha\ \mathsf{type_s^p}\rangle$ is clearly natural, and each component is clearly stable. The components of eq. (173) are continuous by proposition 2.2.31. To show that they are stable, it is sufficient to show that they are projections.

It is obvious that

$$\langle \Xi, \alpha\ \mathsf{type_s^p} \vdash \alpha\ \mathsf{type_s^p}\rangle^p \cdot \langle \Xi, \alpha\ \mathsf{type_s^p} \vdash \alpha\ \mathsf{type_s^p}\rangle = \mathsf{id}$$

and

$$\langle \Xi, \alpha\ \mathsf{type_s^p} \vdash \alpha\ \mathsf{type_s^p}\rangle \cdot \langle \Xi, \alpha\ \mathsf{type_s^p} \vdash \alpha\ \mathsf{type_s^p}\rangle^p \sqsubseteq \mathsf{id}.$$

[23] This family need not be natural.

Case (C↓): Recall eqs. (155) to (157). By the induction hypothesis, $\langle \Xi \vdash A \ \mathsf{type}_s^- \rangle$ is a natural family of stable embeddings, with associated projections $\langle \Xi \vdash A \ \mathsf{type}_s^- \rangle^p$. It is sufficient to recognize $\langle \Xi \vdash \downarrow A \ \mathsf{type}_s^+ \rangle$ as $\delta^e \cdot \langle \Xi \vdash A \ \mathsf{type}_s^- \rangle$, recall that e-p-pairs and stable morphisms are closed under composition, and that the lifting functor is locally continuous. Indeed,

$$\begin{aligned}
&\langle \Xi \vdash \downarrow A \ \mathsf{type}_s^+ \rangle^p \cdot \langle \Xi \vdash \downarrow A \ \mathsf{type}_s^+ \rangle \\
&= (-)_\perp \langle \Xi \vdash A \ \mathsf{type}_s^- \rangle \cdot \delta \cdot \delta^e \cdot \langle \Xi \vdash A \ \mathsf{type}_s^- \rangle \\
&= (-)_\perp \left(\langle \Xi \vdash A \ \mathsf{type}_s^- \rangle \cdot \langle \Xi \vdash A \ \mathsf{type}_s^- \rangle \right) \\
&= \mathsf{id}.
\end{aligned}$$

The proof that $\langle \Xi \vdash \downarrow A \ \mathsf{type}_s^+ \rangle \cdot \langle \Xi \vdash \downarrow A \ \mathsf{type}_s^+ \rangle^p \sqsubseteq \mathsf{id}$ is analogous.

Case (Cρ_n^+): Recall eqs. (177) and (178). We proceed by case analysis on n. If $n = 0$, then $\langle \Xi \vdash \rho^{\circ} \alpha.A \ \mathsf{type}_s \rangle_\xi(\perp) = (\perp, \perp)$ is a constant family of constant functions. For each component, the domain and codomain each contain exactly one element, so each component is clearly an embedding. If $n = k + 1$, then by lemma 8.4.5,

$$\begin{aligned}
\langle \Xi \vdash \rho^{k+1} \alpha.A \ \mathsf{type}_s^+ \rangle^+ &= (-)_\perp \langle \Xi \vdash [\rho^k \alpha.A/\alpha] A \ \mathsf{type}_s^+ \rangle^+, \\
\langle \Xi \vdash \rho^{k+1} \alpha.A \ \mathsf{type}_s^+ \rangle^- &= \mathsf{down} * \langle \Xi \vdash [\rho^k \alpha.A/\alpha] A \ \mathsf{type}_s^+ \rangle^-.
\end{aligned}$$

Observe that

$$(\Xi \vdash [\rho^k \alpha.A/\alpha] A \ \mathsf{type}_s^+) \prec (\Xi \vdash \rho^{k+1} \alpha.A \ \mathsf{type}_s^+),$$

so by the well-founded induction hypothesis, $\langle \Xi \vdash [\rho^k \alpha.A/\alpha] A \ \mathsf{type}_s^+ \rangle$ is a natural family of embeddings. The remainder of the case is identical to the case for (C↓).

Case (Cρ^+): Recall eqs. (182) and (183). The morphism $\langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle_\xi$ is given by the mediating morphism of cocones in fig. 8.1 . The top ω-chain lies in $\mathbf{Stab}_{\perp !}{}^e$ (relative to the stable ordering) by proposition 4.5.1, and the top cocone is colimiting by definition. By corollary 2.2.64, it is also colimiting in **DCPO**. The bottom ω-chain is the product of two ω-chains and also lies in $\mathbf{Stab}_{\perp !}{}^e$. The cocone on the bottom ω-chain is colimiting because because left-adjoints preserve colimits [Rie16, Theorem 4.5.3; Mac98, p. 119], so it too lies in $\mathbf{Stab}_{\perp !}{}^e$. The natural transformation between the two ω-chains is a family of embeddings by the induction hypothesis. The lower cocone on the top ω-chain then lies in $\mathbf{DCPO}^e$. It follows that the mediating morphism is an embedding (relative to the pointwise ordering) by proposition 2.2.63. By proposition 2.2.63, it is the directed supremum of compositions of stable maps, so it is stable. To see that $\langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle$ is natural, observe that

$$\langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle = \langle \langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle^+, \langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle^- \rangle,$$

and that $\langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle^+$ and $\langle \Xi \vdash \rho \alpha.A \ \mathsf{type}_s^+ \rangle^-$ are natural by proposition 4.3.1.

Case (C⊕): Recall eqs. (163) to (165). We first show that $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle$ is a natural transformation in $\mathbf{DCPO}_{\perp !}$. To do so, it is sufficient to show that $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle^+$ and $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle^-$ are natural. The case $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle^+$ follows by the induction hypothesis and the fact that functors preserve commuting diagrams. The case $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle^-$ follows by the induction hypothesis and an easy computation.

Next, we show that the components of $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle$ are stable. Consider some arbitrary component ξ, and assume that $x \uparrow y$ in $[\![\Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+]\!] \xi$. We must show that $\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle_\xi (x \sqcap y) = \langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle_\xi (x) \sqcap \langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle_\xi (y)$. Three cases are possible. The first, that $x = y$, is immediate. The second is that, without loss of generality, $x = \perp$. In this case, the result follows easily from the fact that embeddings are strict (proposition 2.2.21). The final is that $x = (l, [x_l])$ and $y = (l, [y_l])$. Then:

$$\begin{aligned}
&\langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle_\xi (x \sqcap y) \\
&= \langle \Xi \vdash \oplus \{ l : A_l \}_{l \in L} \ \mathsf{type}_s^+ \rangle_\xi (l, [x_l \sqcap y_l]) \\
&= ((l, [\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^+ (x_l \sqcap y_l)]), \iota_l (\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^- (x_l \sqcap y_l))),
\end{aligned}$$

which by the induction hypothesis:

$$\begin{aligned}
&= ((l, [\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^+(x_l) \sqcap \langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^+(y_l)]),\\
&\qquad\qquad \iota_l(\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^-(x_l) \sqcap \langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^-(y_l))),\\
&= ((l, [\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^+(x_l)]), \iota_l(\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^-(x_l))) \sqcap\\
&\qquad\qquad \sqcap ((l, [\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^+(y_l)]), \iota_l(\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle^-(y_l)))\\
&= \langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi(x) \sqcap \langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi(y).
\end{aligned}$$

Finally, we show that the components of $\langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle$ are embeddings. Consider some component ξ, and let $(l, [a_l]) \in [\![\Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+]\!]\xi$ be arbitrary.[24] Then, by computation and the induction hypothesis:

$$\begin{aligned}
&\left(\langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi^p \circ \langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi\right)(l, [a_l])\\
&= \langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi^p \left((l, [\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle_\xi^+ a_l]), \iota_l(\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle_\xi^- a_l)\right)\\
&= \left(l, [\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle_\xi^p(\langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle_\xi^+ a_l, \langle \Xi \vdash A_l \ \mathsf{type}_s^+ \rangle_\xi^- a_l)]\right)\\
&= (l, [a_l]).
\end{aligned}$$

The proof that

$$\left(\langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi \circ \langle \Xi \vdash \oplus\{l : A_l\}_{l\in L} \ \mathsf{type}_s^+ \rangle_\xi^p\right) \sqsubseteq \mathsf{id}$$

is similar.

Case (C⊗)**:** Recall eqs. (127) to (129). It follows from the induction hypothesis and general categorical properties that $\langle \Xi \vdash A \otimes B \ \mathsf{type}_s^+ \rangle$ is natural. To show that its components are stable, it is sufficient to show that $\langle \Xi \vdash A \otimes B \ \mathsf{type}_s^+ \rangle^+$ and $\langle \Xi \vdash A \otimes B \ \mathsf{type}_s^+ \rangle^-$ are stable. Stability of $\langle \Xi \vdash A \otimes B \ \mathsf{type}_s^+ \rangle^+$ follows from the induction hypothesis and the fact that $\mathbf{Stab}_{\perp!}$ is closed under lifting and products. Stability of $\langle \Xi \vdash A \otimes B \ \mathsf{type}_s^+ \rangle^-$ follows from the induction hypothesis, the fact that $\mathbf{Stab}_{\perp!}$ is closed under products, and the fact down is stable. Finally, it follows easily from the induction hypothesis and lemma 2.2.50 that the components of $\langle \Xi \vdash A \otimes B \ \mathsf{type}_s^+ \rangle$ are embeddings.

Case (C∧)**:** Recall eqs. (147) to (149). It follows from the induction hypothesis that $\langle \Xi \vdash \tau \wedge A \ \mathsf{type}_s^+ \rangle$ is natural. Stability follows from the induction hypothesis and the fact that $\mathbf{Stab}_{\perp!}$ is closed under smash products. An uninteresting computation reveals that the components of $\langle \Xi \vdash \tau \wedge A \ \mathsf{type}_s^+ \rangle$ are embeddings. □

Lemma 8.4.7. *Let $a : A \to A^+ \times A^-$ be a well-woven embedding, and let $\delta^e : (A^+ \times A^-)_\perp \to A_\perp^+ \times A^-$ be given by lemma 2.2.50. Then $\delta^e \circ a_\perp : A_\perp \to A_\perp^+ \times A^-$ is a well-woven embedding.*

Proof. It is clearly an embedding, for embeddings are closed under composition. Set $w = \delta^e \circ a_\perp$, and let $(a^+, a^-) \in A_\perp^+ \times A^-$ be arbitrary in its codomain. If $a^+ = \perp$, then the minimum solution (α^+, α^-) to the weaving equations

$$\begin{aligned}
(w^+ \circ w^p)(a^+, \alpha^-) &\sqsubseteq \alpha^+\\
(w^- \circ w^p)(\alpha^+, a^-) &\sqsubseteq \alpha^-
\end{aligned}$$

is $(\alpha^+, \alpha^-) = (\perp, \perp)$, and in this case,

$$w^p(a^+, \alpha^-) = \perp = w^p(\alpha^+, a^-)$$

as desired. If $a^+ = [a_o^+]$, then let (β^+, β^-) be the minimum solution to the weaving equations

$$\begin{aligned}
(a^+ \circ a^p)(a_o^+, \beta^-) &\sqsubseteq \beta^+\\
(a^- \circ a^p)(\beta^+, a^-) &\sqsubseteq \beta^-.
\end{aligned}$$

Then $([\beta^+], \beta^-$ is the least solution to the weaving equations for w^+, and

$$w^p(a^+, \beta^-) = [a(a_o^+, \beta^-)] = [a(\beta^+, a^-)] = w^p([\beta^+], a^-).$$

[24] Because morphisms in $\mathbf{Stab}_{\perp!}$ are strict, it is sufficient to consider only non-bottom elements.

We conclude that w is well-woven. □

Proposition 8.4.8. *If* $\Xi \vdash A\ \mathsf{type}_s$, *then each component of* $\langle \Xi \vdash A\ \mathsf{type}_s \rangle$ *is well-woven.*

Proof. By well-founded induction on the set of open session types, using the order defined in the proof of proposition 8.4.6.

Case (C**1**)**:** Recall:

$$\langle \Xi \vdash \mathbf{1}\ \mathsf{type}_s^+ \rangle^p = \pi_1 \tag{121}$$

Consider some arbitrary component ξ, and let (a^+, a^-) be arbitrary in its domain. A case analysis $a^+ = \bot$ or $a^+ = \mathsf{close}$ gives the result. In both cases, (a^+, a^-) is the minimal solution.

Case (CVar)**:** Recall:

$$\langle \Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p \rangle^p = \sqcap \tag{173}$$

Consider some arbitrary component ξ, and let (a^+, a^-) be arbitrary in its domain. Then $(\bot, \bot)$ is the minimal solution to the weaving equations, and it is clear that

$$\langle \Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p \rangle^p_\xi(a^+, \bot) = \bot = \langle \Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p \rangle^p_\xi(\bot, a^-).$$

Case (Cρ_n^+)**:** We proceed by case analysis on n. The case $n = 0$ is obvious: the domain and codomain of $\langle \Xi \vdash \rho^0\alpha.A\ \mathsf{type}_s \rangle_\xi$ are one-element domains. Assume now that $n = k + 1$. By lemma 8.4.5,

$$\langle \Xi \vdash \rho^{k+1}\alpha.A\ \mathsf{type}_s^+ \rangle^+ = (-)_\bot \langle \Xi \vdash [\rho^k\alpha.A/\alpha]A\ \mathsf{type}_s^+ \rangle^+,$$
$$\langle \Xi \vdash \rho^{k+1}\alpha.A\ \mathsf{type}_s^+ \rangle^- = \mathsf{down} * \langle \Xi \vdash [\rho^k\alpha.A/\alpha]A\ \mathsf{type}_s^+ \rangle^-.$$

We recognize $\langle \Xi \vdash \rho^{k+1}\alpha.A\ \mathsf{type}_s^+ \rangle$ as

$$\langle \Xi \vdash \rho^{k+1}\alpha.A\ \mathsf{type}_s^+ \rangle = \delta^e \cdot \langle \Xi \vdash [\rho^k\alpha.A/\alpha]A\ \mathsf{type}_s^+ \rangle,$$

where δ^e is given by lemma 2.2.50. By the induction hypothesis, each component of

$$\langle \Xi \vdash [\rho^k\alpha.A/\alpha]A\ \mathsf{type}_s^+ \rangle$$

is well-woven. We are done by lemma 8.4.7.

Case (Cρ^+)**:** Fix some component ξ. We want to show that $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle_\xi$ is well-woven. Abbreviate $\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+$ by ρ. By corollary 8.2.8, it is sufficient to show that

$$\mathrm{Tr}\Big([\![\rho]\!]^+ \times [\![\rho]\!]^- \times [\![\rho]\!]^+ \times [\![\rho]\!]^- \xrightarrow{\langle\langle\rho\rangle^p, \langle\rho\rangle^+ \circ \langle\rho\rangle^p\rangle \times \langle\langle\rho\rangle^- \circ \langle\rho\rangle^p, \langle\rho\rangle^p\rangle} [\![\rho]\!] \times [\![\rho]\!]^+ \times [\![\rho]\!]^- \times [\![\rho]\!] \xrightarrow{\mathsf{id}\times\sigma\times\mathsf{id}} [\![\rho]\!] \times [\![\rho]\!]^- \times [\![\rho]\!]^+ \times [\![\rho]\!] \Big) = \langle \langle\rho\rangle^p, \langle\rho\rangle^p \rangle, \tag{196}$$

where σ is the obvious product-permuting isomorphism. We recognize the right hand side as the mediating morphism of cocones

$$\begin{array}{ccc} [\![\rho]\!]^+ \times [\![\rho]\!]^- & \dashrightarrow & [\![\rho]\!] \times [\![\rho]\!] \\ {\scriptstyle p_m \times n_m}\uparrow\uparrow & & \uparrow\uparrow{\scriptstyle c_m \times c_m} \\ ([\![\rho^m]\!]^+)_m \times ([\![\rho^m]\!]^-)_m & \xrightarrow{\langle\langle\rho^m\rangle^p, \langle\rho^m\rangle^p\rangle} & ([\![\rho^m]\!])_m \times ([\![\rho^m]\!])_m \end{array}$$

where the bottom left corner of the diagram is the bottom ω-chain of fig. 8.1, the bottom right corner is the product of the top ω-chain of fig. 8.1 with itself, the bottom morphism is the obvious pairing of morphisms from the same figure, and the two vertical families of morphisms are the corresponding colimits. In particular, p_n, n_m, c_m are the legs of the canonical colimiting cones of fig. 8.1:

$$c_m : [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]\xi \to [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]\xi,$$
$$p_m : [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^+\xi \to [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^+\xi,$$
$$n_m : [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^-\xi \to [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^-\xi.$$

We begin with a few simplifying computations. By proposition 2.2.63,

$$\begin{aligned}\langle\rho\rangle &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(p_m \times n_m)\circ\langle\rho^m\rangle\circ c_m^p,\\ \langle\rho\rangle^p &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow} c_m\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p),\end{aligned}\tag{197}$$

so using proposition 2.2.11 and continuity, we calculate that:

$$\begin{aligned}&\langle\rho\rangle^+\circ\langle\rho\rangle^p\\ &= \pi_+\circ\left(\bigsqcup_{m\in\mathbb{N}}^{\uparrow}(p_m\times n_m)\circ\langle\rho^m\rangle\circ c_m^p\right)\circ\left(\bigsqcup_{m\in\mathbb{N}}^{\uparrow} c_m\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p)\right)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}\pi_+\circ(p_m\times n_m)\circ\langle\rho^m\rangle\circ c_m^p\circ c_m\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow} p_m\circ\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p),\end{aligned}$$

and symmetrically, that

$$\langle\rho\rangle^-\circ\langle\rho\rangle^p = \bigsqcup_{m\in\mathbb{N}}^{\uparrow} n_m\circ\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p).$$

The left-hand side of eq. (196) is then equal to:

$$\begin{aligned}&\mathrm{Tr}\left((\mathsf{id}\times\sigma\times\mathsf{id})\circ(\langle\langle\rho\rangle^p,\langle\rho\rangle^+\circ\langle\rho\rangle^p\rangle\times\langle\rho\rangle^-\circ\langle\rho\rangle^p,\langle\rho\rangle^p\rangle)\right)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}\mathrm{Tr}\Big((\mathsf{id}\times\sigma\times\mathsf{id})\circ\big(\langle c_m\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p), p_m\circ\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p)\rangle\times\\ &\qquad\times\langle n_m\circ\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p), c_m\circ\langle\rho^m\rangle^p\circ(p_m^p\times n_m^p)\rangle\big)\Big)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}\mathrm{Tr}\left((\mathsf{id}\times\sigma\times\mathsf{id})\circ(\langle c_m\circ\langle\rho^m\rangle^p, p_m\circ\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\rangle\times\right.\\ &\qquad\left.\times\langle n_m\circ\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p, c_m\circ\langle\rho^m\rangle^p\rangle)\circ\left(p_m^p\times n_m^p\times p_m^p\times n_m^p\right)\right),\end{aligned}$$

which by naturality of trace operators:

$$\begin{aligned}&= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(c_m\times c_m)\circ\mathrm{Tr}\left((\mathsf{id}\times\sigma\times\mathsf{id})\circ(\langle\langle\rho^m\rangle^p, p_m\circ\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\rangle\times\right.\\ &\qquad\left.\times\langle n_m\circ\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p,\langle\rho^m\rangle^p\rangle)\circ\left(\mathsf{id}\times n_m^p\times p_m^p\times\mathsf{id}\right)\right)\circ\left(p_m^p\times n_m^p\right),\end{aligned}$$

which by dinaturality of trace operators:

$$\begin{aligned}&= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(c_m\times c_m)\circ\mathrm{Tr}\left(\left(\mathsf{id}\times n_m^p\times p_m^p\times\mathsf{id}\right)\circ(\mathsf{id}\times\sigma\times\mathsf{id})\circ(\langle\langle\rho^m\rangle^p, p_m\circ\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\rangle\times\right.\\ &\qquad\left.\times\langle n_m\circ\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p,\langle\rho^m\rangle^p\rangle)\right)\circ\left(p_m^p\times n_m^p\right)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(c_m\times c_m)\circ\mathrm{Tr}\left((\mathsf{id}\times\sigma\times\mathsf{id})\circ\left(\mathsf{id}\times p_m^p\times n_m^p\times\mathsf{id}\right)\circ(\langle\langle\rho^m\rangle^p, p_m\circ\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\rangle\times\right.\\ &\qquad\left.\times\langle n_m\circ\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p,\langle\rho^m\rangle^p\rangle)\right)\circ\left(p_m^p\times n_m^p\right)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(c_m\times c_m)\circ\mathrm{Tr}\left((\mathsf{id}\times\sigma\times\mathsf{id})\circ(\langle\langle\rho^m\rangle^p,\pi_1\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p\rangle\times\right.\\ &\qquad\left.\times\langle\pi_2\circ\langle\rho^m\rangle\circ\langle\rho^m\rangle^p,\langle\rho^m\rangle^p\rangle)\right)\circ\left(p_m^p\times n_m^p\right)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(c_m\times c_m)\circ\mathrm{Tr}\left((\mathsf{id}\times\sigma\times\mathsf{id})\circ(\langle\langle\rho^m\rangle^p,\langle\rho^m\rangle^+\circ\langle\rho^m\rangle^p\rangle\times\right.\\ &\qquad\left.\times\langle\langle\rho^m\rangle^-\circ\langle\rho^m\rangle^p,\langle\rho^m\rangle^p\rangle)\right)\circ\left(p_m^p\times n_m^p\right),\end{aligned}$$

which by the induction hypothesis:

$$\begin{aligned}&= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}(c_m\times c_m)\circ\langle\langle\rho^m\rangle^p,\langle\rho^m\rangle^p\rangle\circ\left(p_m^p\times n_m^p\right)\\ &= \bigsqcup_{m\in\mathbb{N}}^{\uparrow}\langle c_m\circ\langle\rho^m\rangle^p\circ\left(p_m^p\times n_m^p\right), c_m\circ\langle\rho^m\rangle^p\circ\left(p_m^p\times n_m^p\right)\rangle,\end{aligned}$$

which by continuity of pairing and eq. (197):

$= \langle \langle \rho \rangle^p, \langle \rho \rangle^p \rangle$.

Case (C⊕): Recall:

$$\langle \Xi \vdash \oplus\{l : A_l\}_{l \in L} \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((k, [a_k^+]), (a_l^-)_{l \in L}) = (k, [\langle \Xi \vdash A_k \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p(a_k^+, a_k^-)]) \tag{165}$$

Consider some arbitrary component ξ, and let $(a^+, (a_l^-)_{l \in L})$ be arbitrary in its domain. If $a^+ = \bot$, then $(\bot, \bot)$ is the minimal solution to the weaving equations. In this case,

$$\langle \Xi \vdash \oplus\{l : A_l\}_{l \in L} \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p(\bot, \bot) = \bot = \langle \Xi \vdash \oplus\{l : A_l\}_{l \in L} \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p(\bot, a^-).$$

Otherwise, $a^+ = (k, [a_k^+])$. Observe that the least solution to the weaving equations for $\langle \Xi \vdash \oplus\{l : A_l\}_{l \in L} \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((k, [a_k^+]), (a_l^-)_{l \in L})$ and (a^+, a^-) is $((k, [\alpha^+]), (k : \alpha^-, l \neq k : \bot)_l)$, where (α^+, α^-) is the least solution to the weaving equations for $\langle \Xi \vdash A_k \ \mathsf{type}_\mathsf{s} \rangle_\xi^p$ and (a_k^+, a_k^-). By the induction hypothesis, it follows that:

$$\begin{aligned}
&\langle \Xi \vdash \oplus\{l : A_l\}_{l \in L} \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((k, [a_k^+]), (a_l^-)_{l \in L})((k, [a_k^+]), (k : \alpha^-, l \neq k : \bot)_l) \\
&= (k, [\langle \Xi \vdash A_k \ \mathsf{type}_\mathsf{s} \rangle_\xi^p(a_k^+, \alpha^-)] \\
&= (k, [\langle \Xi \vdash A_k \ \mathsf{type}_\mathsf{s} \rangle_\xi^p(\alpha^+, a_k^-)]) \\
&= \langle \Xi \vdash \oplus\{l : A_l\}_{l \in L} \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((k, [a_k^+]), (a_l^-)_{l \in L})((k, [\alpha^+]), (a_l^-)_{l \in L}).
\end{aligned}$$

Case (C∧): Recall:

$$\langle \Xi \vdash \tau \wedge A \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((v, [a^+]), a^-) = (v, [\langle \Xi \vdash A \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p(a^+, a^-)]) \tag{149}$$

Consider some arbitrary component ξ, and let (a^+, a^-) be arbitrary in its domain. If $a^+ = \bot$, then $(\bot, \bot)$ is the minimal solution to the weaving equations. Otherwise, a^+ is of the form $(v, [a_\mathsf{o}^+])$. Observe that the least solution to the weaving equations for $\langle \Xi \vdash \tau \wedge A \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((v, [a^+]), a^-)$ and (a_o^+, a^-) is $((v, [\alpha^+]), \alpha^-)$, where (α^+, α^-) is the least solution to the weaving equations for $\langle \Xi \vdash \tau \ \mathsf{type}_\mathsf{f} \rangle_\xi^p$ and (a_o^+, a^-). By the induction hypothesis, it follows that:

$$\begin{aligned}
&\langle \Xi \vdash \tau \wedge A \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((v, [a^+]), a^-)((v, [a_\mathsf{o}^+]), \alpha^-) \\
&= (v, \langle \Xi \vdash \tau \ \mathsf{type}_\mathsf{f} \rangle_\xi^p(a_\mathsf{o}^+, \alpha^-)) \\
&= (v, \langle \Xi \vdash \tau \ \mathsf{type}_\mathsf{f} \rangle_\xi^p(\alpha^+, a^-)) \\
&= \langle \Xi \vdash \tau \wedge A \ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((v, [a^+]), a^-)((v, [\alpha^+]), a^-).
\end{aligned}$$

The remaining cases follow easily by symmetry or analogy with the above cases. □

8.4.2. Semantic Results for Terms and Processes. We show that the denotations of terms and processes are well-defined. Because the definitions of terms and processes are mutually recursive, the proofs of properties of terms and processes will be intertwined.

We start by showing that processes denote junk-free morphisms in $\mathbf{CYO}(\mathbf{Stab}_\bot)$. This entails showing that the processes and terms denote continuous functions, and that processes are junk-free, complete, and frugal. Afterwards, we show that the denotations of processes and terms satisfy the appropriate naturality conditions.

Proposition 8.4.9. *If $\Psi \ ; \Delta \vdash P :: a : A$, then $[\![\Psi \ ; \Delta \vdash P :: a : A]\!]u$ is junk-free for all $u \in [\![\Psi]\!]$.*

Proof. By induction on the derivation $\Psi \ ; \Delta \vdash P :: a : A$. Recall the definition of junk-freedom from definition 8.2.10. We omit cases that follow easily by symmetry or by analogy with other cases. Where $u \in [\![\Psi]\!]$ is arbitrary and $p = [\![\Psi \ ; \Delta \vdash P :: a : A]\!]u$, we must show that

$$((\langle \Delta \rangle^+ \times \langle a : A \rangle^-) \upharpoonright \mathrm{im}(p), p)$$

is an e-p-pair.

Case (Fwd$^+$): Recall eq. (113). We compute:

$$\begin{aligned}&(\langle A\rangle^+ \times \langle A\rangle^-) \circ [\![\Psi \,;\, a : A \vdash a \to b :: b : A]\!]u \\ &= (\langle A\rangle^+ \times \langle A\rangle^-) \circ \langle \langle A\rangle^p, \langle A\rangle^p\rangle \\ &\sqsubseteq \mathsf{id}\end{aligned}$$

by definition of e-p-pair. Let $(a, a) \in \mathrm{im}([\![\Psi \,;\, a : A \vdash a \to b :: b : A]\!]u)$ be arbitrary, then:

$$([\![\Psi \,;\, a : A \vdash a \to b :: b : A]\!]u \circ (\langle A\rangle^+ \times \langle A\rangle^-))(a, a) = (a, a)$$

by definition of e-p-pair.

Case (Cut): Recall eq. (115). We first show that the two functions form an adjunction. We use one of the alternate characterizations of proposition 2.2.19: $l \dashv u$ if and only if for all x, $u(x) = \max(l^{-1}(\downarrow x))$. Let $(\delta_1^+, \delta_2^+, c^-)$ be arbitrary in the domain of $[\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u$, and let (a^+, a^-) be minimum such that

$$\begin{aligned}[\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!]u(\delta_1^+, a^-) &= (\delta_1, a_1) & \langle A\rangle^+(a_1) &\sqsubseteq a^+ \\ [\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]u(\delta_2^+, a^+, c^-) &= (\delta_2, a_2, c) & \langle A\rangle^-(a_2) &\sqsubseteq a^-.\end{aligned}$$

Then

$$[\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u(\delta_1^+, \delta_2^+, c^-) = (\delta_1, \delta_2, c).$$

By the induction hypothesis,

$$\begin{aligned}(\delta_1, a_1) &= \max\left((\langle \Delta_1\rangle^+ \times \langle A\rangle^-)\,(\downarrow(\delta_1^+, a^-))\right) \\ (\delta_2, a_2, c) &= \max\left((\langle \Delta_2, A\rangle^+ \times \langle C\rangle^-)\,(\downarrow(\delta_1^+, a^+, c^-))\right).\end{aligned}$$

The ordering of products is determined point-wise, so it immediately follows that

$$(\delta_1, \delta_2, c) = \max\left((\langle \Delta_1, \Delta_2\rangle^+ \times \langle C\rangle^-)\downarrow(\delta_1^+, \delta_2^+, c^-)\right)$$

as desired.

Next, we show that

$$([\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u \circ (\langle \Delta_1, \Delta_2\rangle^+ \times \langle C\rangle^-))(\delta_1, \delta_2, c) = (\delta_1, \delta_2, c).$$

By the induction hypothesis,

$$\begin{aligned}([\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!]u \circ (\langle \Delta_1\rangle^+ \times \langle A\rangle^-))(\delta_1, a_1) &= (\delta_1, a_1), \\ ([\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]u \circ (\langle \Delta_2, a : A\rangle^+ \times \langle C\rangle^-))(\delta_2, a_2, c) &= (\delta_2, a_2, c).\end{aligned}$$

But by corollary 2.3.9,

$$\begin{aligned}&([\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u \circ (\langle \Delta_1, \Delta_2\rangle^+ \times \langle C\rangle^-))(\delta_1, \delta_2, c) \\ &= (\pi_{\Delta_1,\Delta_2,C} \circ ([\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!]u \times [\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]u) \circ \\ &\qquad \circ (\langle \Delta_1, a : A, \Delta_2\rangle^+ \times \langle a : A, c : C\rangle^-))(\delta_1, a_1, \delta_2, a_2, c) \\ &= \pi_{\Delta_1,\Delta_2,C}(\delta_1, a_1, \delta_2, a_2, c) \\ &= (\delta_1, \delta_2, c).\end{aligned}$$

Case (**1**R): Recall eq. (122). It only has one element in its domain, so it is immediate that

$$\langle \mathbf{1}\rangle^- \circ [\![\Psi \,;\, \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}]\!]u = \mathsf{id}.$$

It only has one element in its image, so it is immediate that

$$[\![\Psi \,;\, \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}]\!]u \circ \langle \mathbf{1}\rangle^- = \mathsf{id}.$$

Case (**1**L): Recall eqs. (119) and (123). We show that

$$(\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-) \circ [\![\Psi \,;\, \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u \sqsubseteq \mathsf{id}.$$

Let (δ^+, a^+, c^-) be arbitrary in its domain. Assume first that $a^+ = \mathsf{close}$. All components except for the channel a are immediate by the induction hypothesis. The a component is immediate from the fact that

$$((\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-) \circ [\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u)\,(\delta^+, \mathsf{close}, c^-) = (\_, \mathsf{close}, \_).$$

If $a^+ = \bot$, then we calculate

$$\begin{aligned}
&((\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-) \circ [\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u)\,(\delta^+, \bot, c^-)\\
&= (\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-)\,(\langle \Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-))\\
&\sqsubseteq (\delta^+, \bot, c^-)
\end{aligned}$$

by definition of e-p-pair.

Conversely, we show that

$$\mathsf{id} = [\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u \circ (\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-)$$

when restricted to the image of $[\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]$. Let (δ, a, c) be arbitrary in this image. We consider two cases. First, assume that it is the image of $(\delta^+, \mathsf{close}, c^-)$. All components except the one for the channel a are immediate by the induction hypothesis. The a component is immediate from the fact that

$$([\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u \circ (\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-))\,(\delta, \mathsf{close}, c) = (\_, \mathsf{close}, \_).$$

Next, assume that it is the image of $(\delta^+, \bot, c^-)$. We show the inequality for the δ component; the c component is analogous. By assumption, $\delta = \langle \Delta\rangle^p(\delta^+, \bot)$, so by definition of e-p-pair, $\langle \Delta\rangle(\delta) = (\delta^+_{\mathrm{o}}, \bot)$ for some δ^+_{o}. It follows, again by definition of e-p-pair, that $\langle \Delta\rangle^p(\delta^+_{\mathrm{o}}, \bot) = \delta$. We deduce that

$$([\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u \circ (\langle \Delta, a : \mathbf{1}\rangle^+ \times \langle C\rangle^-))\,(\delta, \bot, c) = (\delta, \bot, c).$$

Case ($\oplus$R)**:** Recall eqs. (164) and (166). We show that

$$(\langle \Delta\rangle^+ \times \langle \oplus\{l : A_l\}_{l\in L}\rangle^-) \circ [\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u \sqsubseteq \mathsf{id}.$$

Let $(\delta^+, (a_l^-)_{l\in L})$ be arbitrary in its domain. Observe that

$$\left((\langle \Delta\rangle^+ \times \langle \oplus\{l : A_l\}_{l\in L}\rangle^-) \circ [\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u\right)(\delta^+, (a_l^-)_{l\in L}) = (\hat{\delta}^+, \iota_k(\hat{a}_k^-))$$

where

$$((\langle \Delta\rangle^+ \times \langle A_k\rangle^-) \circ [\![\Psi\ ;\ \Delta \vdash P :: a : A_k]\!]u)(\delta^+, a_k^-) = (\hat{\delta}^+, \hat{a}_k^-).$$

By the induction hypothesis, $(\hat{\delta}^+, \hat{a}_k^-) \sqsubseteq (\delta^+, a_k^-)$. It follows that $(\hat{\delta}^+, \iota_k(\hat{a}_k^-)) \sqsubseteq (\delta^+, (a_l^-)_{l\in L})$ as desired.

Conversely, we show that

$$\mathsf{id} = [\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u \circ (\langle \Delta\rangle^+ \times \langle \oplus\{l : A_l\}_{l\in L}\rangle^-)$$

when restricted to the image of $[\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u$. Let $(\delta^+, (a_l^-)_{l\in L})$ be arbitrary in the domain of $[\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u$, and set

$$[\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u(\delta^+, (a_l^-)_{l\in L}) = (\delta, (k, [a_k])).$$

Set

$$(\hat{\delta}, \hat{a}_k) = ([\![\Psi\ ;\ \Delta \vdash P :: a : A_k]\!]u \circ (\langle \Delta\rangle^+ \times \langle A_k\rangle^-))(\delta, a_k).$$

By the induction hypothesis, $(\delta, a_k) = (\hat{\delta}, \hat{a}_k)$. But a computation reveals that

$$([\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u \circ (\langle \Delta\rangle^+ \times \langle \oplus\{l : A_l\}_{l\in L}\rangle^-))(\delta, (k, [a_k])) = (\hat{\delta}, (k, [\hat{a}_k])).$$

The result is now obvious.

Case (&R): Recall eqs. (206) and (230). We begin by showing that

$$(\langle\Delta\rangle^+ \times \langle \&\{l : A_l\}_{l\in L}\rangle^-) \circ [\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u \sqsubseteq \mathsf{id}.$$

Let (δ^+, a^-) be arbitrary in the domain of $[\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u$. If $a^- = \bot$, then the proof is identical to the case (**1**L). If $a^- = (l, [a_l^-])$, then set

$$(\hat\delta, \hat a_l^-) = ((\langle\Delta\rangle^+ \times \langle A_l\rangle^-) \circ [\![\Psi \,;\, \Delta, a : A_l \vdash P_l :: c : C]\!]u)(\delta^+, a_l^-).$$

By the induction hypothesis, $(\hat\delta^+, \hat a_l^-) \sqsubseteq (\delta^+, a_l^-)$. By computation,

$$((\langle\Delta\rangle^+ \times \langle \&\{l : A_l\}_{l\in L}\rangle^-) \circ [\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u)(\delta^+, (l, [a_l^-]))$$
$$= (\hat\delta^+, (l, [\hat a_l^-])).$$

It is immediate that $(\hat\delta^+, (l, [\hat a_l^-])) \sqsubseteq (\delta^+, (l, [a_l^-]))$.

Conversely, we show that

$$\mathsf{id} = [\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u \circ (\langle\Delta\rangle^+ \times \langle \&\{l : A_l\}_{l\in L}\rangle^-).$$

Let (δ^+, a^-) be arbitrary in the domain of $[\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u$, and set

$$(\delta, a) = [\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u(\delta^+, a^-).$$

We consider two cases. If $a^- = \bot$, then $a = \bot$, and the proof is identical to the case (**1**L). If $a^- = (l, [a_l^-])$, then $a = (l, [a_l])$ for some a_l. Set

$$(\hat\delta, \hat a_l) = ([\![\Psi \,;\, \Delta \vdash P_l :: a : A_l]\!]u \circ (\langle\Delta\rangle^+ \times \langle A_l\rangle^-)(\delta, a_l).$$

By the induction hypothesis, $(\delta, a_l) = (\hat\delta, \hat a_l)$. We compute that

$$([\![\Psi \,;\, \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u \circ (\langle\Delta\rangle^+ \times \langle \&\{l : A_l\}_{l\in L}\rangle^-))(\delta, (l, [a_l]))$$
$$= (\hat\delta, (l, [\hat a_l])).$$

The result is now obvious.

Case ($\wedge$R): Recall eqs. (148) and (150). The result follows by the induction hypothesis when $[\![\Psi \Vdash M : \tau]\!]u \neq \bot$. When $[\![\Psi \Vdash M : \tau]\!]u$ is $\bot$, then the proof is identical to the case (**1**L).

Case ($\supset$R): Recall eqs. (211) and (242). We start by showing that

$$((\langle\Delta\rangle^+ \times \langle \tau \supset A\rangle^-) \circ [\![\Psi \,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u) \sqsubseteq \mathsf{id}.$$

Let (δ^+, a^-) be arbitrary in its domain. If $a^- = \bot$, then the proof is identical to the case (**1**L). If $a^- = (v, [a_o^-])$, then set

$$(\hat\delta^+, \hat a_o^-) = ((\langle\Delta\rangle^+ \times \langle A\rangle^-) \circ [\![\Psi, x : \tau \,;\, \Delta \vdash P :: a : A]\!][u \mid x \mapsto v])(\delta^+, a_o^-).$$

By the induction hypothesis,[25] $(\hat\delta^+, \hat a_o^-) \sqsubseteq (\delta^+, a_o^-)$. By computation,

$$((\langle\Delta\rangle^+ \times \langle \tau \supset A\rangle^-) \circ [\![\Psi \,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u)(\delta^+, a^-) = (\hat\delta^+, (v, [\hat a_o^-])).$$

The result is now obvious.

Conversely, we show that

$$\mathsf{id} = [\![\Psi \,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u \circ (\langle\Delta\rangle^+ \times \langle \tau \supset A\rangle^-)$$

when restricted to the image of $[\![\Psi \,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u$. Let (δ^+, a^-) be arbitrary in the domain of $[\![\Psi \,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u$, and set

$$(\delta, a) = [\![\Psi \,;\, \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u(\delta^+, a^-).$$

If $a^- = \bot$, then the proof is identical to the case (**1**L). If $a^- = (v, [a_o^-]$, then set

$$(\hat\delta, \hat a_o) = ([\![\Psi, x : \tau \,;\, \Delta \vdash P :: a : A]\!]u \circ (\langle\Delta\rangle^+ \times \langle A\rangle^-))(\delta, a).$$

[25] The induction hypothesis quantifies over all environments u, including $[u \mid x \mapsto v]$.

By the induction hypothesis, $(\delta, a) = (\hat{\delta}, \hat{a}_\circ)$. By computation,

$$(\llbracket \Psi \;;\; \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle \tau \supset A \rangle^-))(\delta, a) = (\hat{\delta}, (v, [\hat{a}_\circ])).$$

The result is now obvious.

Case (↓R)**:** Recall eqs. (156) and (158). The result is immediate from the induction hypothesis, and the fact that adjoints are closed under composition.

Case (↑R)**:** Recall eqs. (204) and (224). We begin by showing that

$$(\langle \Delta \rangle^+ \times \langle \uparrow A \rangle^-) \circ \llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u \sqsubseteq \mathsf{id}.$$

Let (δ^+, a^-) be arbitrary in its domain. If $a^- = \bot$, then the proof is identical to the case (**1**L). If $a^- = [a_\circ^-]$, then set

$$(\hat{\delta}^+, \hat{a}_\circ^-) = ((\langle \Delta \rangle^+ \times \langle A \rangle^-) \circ \llbracket \Psi \;;\; \Delta \vdash P :: a :\rrbracket u)(\delta^+, a_\circ^-).$$

By the induction hypothesis, $(\hat{\delta}^+, \hat{a}_\circ^-) \sqsubseteq (\delta^+, a_\circ^-)$. By computation,

$$(\hat{\delta}^+, [\hat{a}_\circ^-]) = ((\langle \Delta \rangle^+ \times \langle \uparrow A \rangle^-) \circ \llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u)(\delta^+, a^-).$$

The result is now obvious.

Conversely, we show that

$$\mathsf{id} = \llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle \uparrow A \rangle^-)$$

when restricted to the image of $\llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u$. Let (δ^+, a^-) be arbitrary in the domain of $\llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u$. If $a^- = \bot$, then the proof is identical to the case (**1**L). If $a^- = [a_\circ^-]$, then set

$$(\delta, [a_\circ]) = \llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u(\delta^+, a^-).$$

Set

$$(\hat{\delta}, \hat{a}_\circ) = (\llbracket \Psi \;;\; \Delta \vdash P :: a :\rrbracket u \circ (\langle \Delta \rangle^+ \times \langle A \rangle^-))(\delta, a_\circ).$$

By the induction hypothesis, $(\delta, a_\circ) = (\hat{\delta}, \hat{a}_\circ)$. By computation,

$$(\hat{\delta}, [\hat{a}_\circ]) = (\llbracket \Psi \;;\; \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle \uparrow A \rangle^-))(\delta, [a_\circ]).$$

The result is now obvious.

Case (⊗R)**:** Recall eqs. (128) and (130). We begin by showing that

$$(\langle \Delta \rangle^+ \times \langle B \otimes A \rangle^-) \circ \llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u \sqsubseteq \mathsf{id}.$$

Let $(\delta^+, b^+, (a_B^-, a_A^-))$ be arbitrary in its domain. Set

$$(\hat{\delta}^+, \hat{a}_A^-) = ((\langle \Delta \rangle^+ \times \langle A \rangle^-) \circ \llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u)(\delta^+, a_A^-).$$

By the induction hypothesis, $(\hat{\delta}^+, \hat{a}_A^-) \sqsubseteq (\delta^+, a_A^-)$. By computation,

$$\begin{aligned}&((\langle \Delta \rangle^+ \times \langle B \otimes A \rangle^-) \circ \llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u)(\delta^+, b^+, (a_B^-, a_A^-))\\ &= (\hat{\delta}^+, \hat{b}^+, (\hat{a}_B^+, \hat{a}_A^-))\end{aligned}$$

where $(\hat{b}^+, \hat{a}_B^-) = (\langle B \rangle \circ \langle B \rangle^p)(b^+, a_B^-)$. By definition of e-p-pair, $(\hat{b}^+, \hat{a}_B^-) \sqsubseteq (b^+, a_B^-)$. The result is now obvious.

Conversely, we show that

$$\mathsf{id} = \llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle B \otimes A \rangle^-)$$

when restricted to the image of $\llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u$. Let $(\delta^+, b^+, (a_B^-, a_A^-))$ be arbitrary in the domain of $\llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u$, and set

$$(\delta, b, [(b, a)]) = \llbracket \Psi \;;\; \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A \rrbracket u(\delta^+, b^+, (a_B^-, a_A^-)).$$

Set

$$(\hat{\delta}, \hat{a}) = (\llbracket \Psi \;;\; \Delta \vdash P :: a : A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle A \rangle^-))(\delta, a).$$

By the induction hypothesis, $(\delta, a) = (\hat{\delta}, \hat{a})$. By computation,

$$(\llbracket \Psi \;;\; \Delta \vdash P :: a : A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle B \otimes A \rangle^-))(\delta, b, [(b, a)]) = (\hat{\delta}, \hat{b}, [(\hat{b}, \hat{a})])$$

where $\hat{b} = \langle B \rangle^p(\langle B \rangle^+(b), \langle B \rangle^-(b))$. But b was in the image of $\langle B \rangle^p$, so $\hat{b} = b$. The result is now obvious.

Case ($\multimap$R)**:** Recall eqs. (208) and (236). We show that

$$(\langle \Delta \rangle^+ \times \langle B \multimap A \rangle^-) \circ \llbracket \Psi \;;\; \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A \rrbracket u \sqsubseteq \mathsf{id}.$$

Let (δ^+, a^-) be arbitrary in the domain. If $a^- = \bot$, then the proof is identical to the case (**1**L). If $a^- = [(b_o^-, a_o^-)]$, then the result follows easily from the induction hypothesis.

Conversely, we show that

$$\mathsf{id} = \llbracket \Psi \;;\; \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle B \multimap A \rangle^-)$$

when restricted to the image of $\llbracket \Psi \;;\; \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A \rrbracket u$. Let (δ^+, a^-) be arbitrary in the domain of $\llbracket \Psi \;;\; \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A \rrbracket u$, and set

$$(\delta, a) = \llbracket \Psi \;;\; \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A \rrbracket u(\delta^+, a^-).$$

If $a^- = \bot$, then $a = \bot$ and the proof is identical to the case (**1**L). If $a^- = [(b_o^+, a_o^-)]$, then $a = [(b_o, a_o)]$. Set

$$(\hat{\delta}, \hat{b}, \hat{a}) = (\llbracket \Psi \;;\; \Delta, b : B \vdash P :: a : A \rrbracket u \circ (\langle \Delta, B \rangle^+ \times \langle A \rangle^-))(\delta, b_o, a_o).$$

By the induction hypothesis, $(\delta, b, a) = (\hat{\delta}, \hat{b}, \hat{a})$. By computation,

$$(\llbracket \Psi \;;\; \Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle B \multimap A \rangle^-))(\delta, a) = (\hat{\delta}, [(\hat{b}, \hat{a})]).$$

The result is now obvious.

Case (ρ^+R)**:** Recall eqs. (183) and (188). By corollary 4.3.5, we recognize $\langle \Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+ \rangle^-$ as

$$\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A \rangle^- \circ \mathsf{Unfold} \circ \mathsf{down}$$

We compute:

$$\begin{aligned}
&(\langle \Delta \rangle^+ \times \langle \rho\alpha.A \rangle^-) \circ \llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u \\
&= (\langle \Delta \rangle^+ \times (\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A \rangle^- \circ \mathsf{Unfold} \circ \mathsf{down})) \circ \\
&\qquad \circ (\mathsf{id} \times (a : \mathsf{up} \circ \mathsf{Fold})) \circ \llbracket \Psi \;;\; \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \rrbracket u \circ (\mathsf{id} \times (a^- : \mathsf{Unfold})) \\
&= (\langle \Delta \rangle^+ \times (\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A \rangle^-)) \circ \\
&\qquad \circ \llbracket \Psi \;;\; \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \rrbracket u \circ (\mathsf{id} \times (a^- : \mathsf{Unfold}))
\end{aligned}$$

which by the induction hypothesis,

$$\begin{aligned}
&\sqsubseteq (\mathsf{id} \times (\mathsf{Fold} \circ \mathsf{Unfold})) \\
&= \mathsf{id}.
\end{aligned}$$

We use a similar approach to show that

$$\mathsf{id} = \llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle \rho\alpha.A \rangle^-)$$

when restricted to the image of $\llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u$. Let (δ^+, a^-) be arbitrary in the domain of $\llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u$, and set

$$(\delta, [\mathsf{Fold}(a)]) = \llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u(\delta^+, a^-).$$

Observe that $(\delta, [\mathsf{Fold}(a)])$ is in the image of $\llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u$ if and only if (δ, a) is in the image of $\llbracket \Psi \;;\; \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A \rrbracket u$. We compute using the above identities

and the induction hypothesis:

$$\begin{aligned}&(\llbracket \Psi \;;\; \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A \rrbracket u \circ (\langle \Delta \rangle^+ \times \langle \rho\alpha.A \rangle^-))(\delta, [\mathsf{Fold}(a)])\\ &= (\mathsf{id} \times (a : \mathsf{up} \circ \mathsf{Fold}))\,(\delta, a)\\ &= (\delta, [\mathsf{Fold}(a)]).\end{aligned}$$

Case (E-{}): Recall eq. (143). The result is then immediate from the fact that $\mathsf{down} \circ \llbracket \Psi \Vdash M : \{a : A \leftarrow \overline{a_i : A_i}\} \rrbracket u$ is (by definition and construction) an element of a dcpo of junk-free functions. □

Proposition 8.4.10. *If* $\Psi \;;\; \Delta \vdash P :: a : A$*, then* $\llbracket \Psi \;;\; \Delta \vdash P :: a : A \rrbracket u$ *is complete for all* $u \in \llbracket \Psi \rrbracket$.

Proof. By induction on the derivation $\Psi \;;\; \Delta \vdash P :: a : A$. Recall the definition of completeness from definition 8.2.10. In each case, we must show that two functions between products are equal. To do so, we show that both functions agree in each component of their image. In the majority of cases, agreement in all but one of the components (typically the provided channel) will be given by the induction hypothesis. The remaining component will follow by a computation. We omit cases that follow easily by symmetry or by analogy with other cases.

Case (Fwd^+): Recall eq. (113). We must show that:

$$\langle A, \overline{A} \rangle^p \circ \langle \mathsf{id}, \langle A, \overline{A} \rangle^- \circ \langle \langle A \rangle^p, \langle A \rangle^p \rangle \rangle = \langle \langle A \rangle^p, \langle A \rangle^p \rangle.$$

For all (a^+, a^-) in their domain,

$$\begin{aligned}&(\langle A, \overline{A} \rangle \circ \langle A, \overline{A} \rangle^p)(a^+, a^-, a^-, a^+)\\ &\sqsubseteq \langle \mathsf{id}, \langle A, \overline{A} \rangle^- \circ \langle \langle A \rangle^p, \langle A \rangle^p \rangle \rangle (a^+, a^-)\\ &\sqsubseteq (a^+, a^-, a^-, a^+)\end{aligned}$$

by definition e-p-pair and monotonicity. By proposition 2.2.19,

$$\langle A \rangle^p \circ \langle A \rangle \circ \langle A \rangle^p = \langle A \rangle^p.$$

By monotonicity, it follows that if f is such that $\langle A \rangle \circ \langle A \rangle^p \sqsubseteq f \sqsubseteq \mathsf{id}$, then $\langle A \rangle^p \circ f = \langle A \rangle^p$. The result follows easily from this observation instantiated with the above inequalities, and a component-wise analysis.

Case (Cut): Recall:

$$\llbracket \Psi \;;\; \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C \rrbracket u = \llbracket \Psi \;;\; a : A, \Delta_2 \vdash Q :: c : C \rrbracket u \circ_a \llbracket \Psi \;;\; \Delta_1 \vdash P :: a : A \rrbracket u \quad (115)$$

Let $(\delta_1^+, \delta_2^+, c^-)$ be arbitrary in its domain, and let (a^+, a^-) be the witnesses for the above composition at $(\delta_1^+, \delta_2^+, c^-)$. Then

$$\llbracket \Psi \;;\; \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C \rrbracket u(\delta_1^+, \delta_2^+, c^-) = (\delta_1, \delta_2, c)$$

where

$$\begin{aligned}\llbracket \Psi \;;\; \Delta_1 \vdash P :: a : A \rrbracket u(\delta_1^+, a^-) &= (\delta_1, \_)\\ \llbracket \Psi \;;\; a : A, \Delta_2 \vdash Q :: c : C \rrbracket u(\delta_2^+, a^+, c^-) &= (\delta_2, \_, c).\end{aligned}$$

The result follows by the induction hypothesis.

Case (**1**R): Recall eq. (122). The only element in its domain is $\bot$. Completeness follows from the fact that $\langle \mathbf{1} \rangle^p(\mathsf{close}, \bot) = \mathsf{close}$.

Case (**1**L): Recall eq. (123). Let (δ^+, a^+, c^-) be arbitrary in its domain. We proceed by case analysis on a^+.

Subcase $a^+ = \mathsf{close}$: The Δ and c components are immediate by the induction hypothesis. The a component of $\llbracket \Psi \;;\; \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C \rrbracket u(\delta^+, a^+, c^-)$ is close. Completeness follows from the observation that $\langle \mathbf{1} \rangle(a) = (\mathsf{close}, \bot)$ and $\langle \mathbf{1} \rangle^p(\mathsf{close}, \bot) = \mathsf{close}$.

Subcase $a^+ = \bot$**:** The interpretation is complete in the a component: $\langle \mathbf{1} \rangle(\bot) = (\bot, \bot)$, and $\langle \mathbf{1} \rangle^p(\bot, \bot) = \bot$. The other two components are complete by monotonicity and the definition of e-p-pair. Taking the Δ component as a concrete example, let $\delta^- = (\langle \Delta \rangle^- \circ \langle \Delta \rangle^p)(\delta^+, \bot)$. Then $\delta^- = \bot$ by monotonicity and properties of e-p-pairs. The result then follows by reflexivity: $\langle \Delta \rangle^p(\delta^+, \delta^-) = \langle \Delta \rangle^p(\delta^+, \bot)$.

Case ($\oplus$R)**:** Recall eqs. (163), (165) and (166). Let $(\delta^+, (a_l^-)_{l \in L})$ be arbitrary in the domain of eq. (166). Set

$$(\delta, a_k) = [\![\Psi \,;\, \Delta \vdash P :: a : A_k]\!] u \, (\delta^+, a_k^-) \,,$$
$$(\delta^-, a_k^+) = (\langle \Delta \rangle^- \times \langle A_k \rangle^+)(\delta, a_k).$$

By the induction hypothesis,

$$\langle \Delta \rangle^p(\delta^+, \delta^-) = \delta,$$

so eq. (166) is complete in the Δ component. As for the a component,

$$\langle A_k \rangle^p(a_k^+, a_k^-) = a_k$$

by the induction hypothesis. By definition,

$$\langle \oplus\{l : A_l\}_{l \in L} \rangle(k, [a_k]) = (k, [a_k^+]).$$

It follows that:

$$\langle \oplus\{l : A_l\}_{l \in L} \rangle^p \left((k, [a_k^+]), (a_l^-)_{l \in L} \right) = (k, [\langle A_k \rangle^p(a_k^+, a_k^-)]) = (k, [a_k]).$$

Completeness is now immediate.

Case (&R)**:** Recall eqs. (206), (229) and (231). Let (δ^+, a^-) be arbitrary in the domain of eq. (206). As in case (**1**L), we proceed by case analysis on a^-. If $a^- = (l, [a_l^-])$, then set

$$(\delta, a_l) = [\![\Psi \,;\, \Delta \vdash P_l :: a : A_l]\!] u(\delta^+, a_l^-).$$

Completeness of eq. (206) in the Δ component follows by the induction hypothesis. By definition,

$$\langle \&\{l : A_l\}_{l \in L} \rangle^+(l, [a_l]) = \iota_l(a_l^+)$$

where $a_l^+ = \langle A_l \rangle^+(a_l)$. By the induction hypothesis, $\langle A_l \rangle^p(a_l^+, a_l^-) = a_l$. It follows that

$$\langle \&\{l : A_l\}_{l \in L} \rangle^p \left(\iota_l(a_l^+), (l, [a_l^+]) \right) = (l, [\langle A_l \rangle^p(a_l^+, a_l^-)]) = (l, [a_l]).$$

Completeness in the a component is now immediate. When $a^- = \bot$, the proof is analogous to case (**1**L).

Case ($\wedge$R)**:** Recall eqs. (148) to (150). The result follows easily by the induction hypothesis when $[\![\Psi \Vdash M : \tau]\!] u \neq \bot$. When $[\![\Psi \Vdash M : \tau]\!] u = \bot$, the proof is analogous to case (**1**L).

Case ($\supset$R)**:** Recall eqs. (211), (241) and (243). Let (δ^+, a^-) be arbitrary in the domain of eq. (211). If $a^- = \bot$, then the proof is analogous to case (**1**L). If $a^- = (v, [a_o^-])$, then the result follows by the induction hypothesis.

Case ($\downarrow$R)**:** Recall eqs. (156) to (158). The result follows easily from the induction hypothesis.

Case ($\uparrow$R)**:** Recall eqs. (204), (223) and (225). Let (δ^+, a^-) be arbitrary in the domain of eq. (204). If $a^- = \bot$, then the proof is analogous to case (**1**L). If $a^- = [a_o^-]$, then set

$$(\delta, [a_o]) = [\![\Psi \,;\, \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : \uparrow A]\!] u(\delta^+, a^-),$$
$$a^+ = \langle A \rangle^+(a_o).$$

Completeness in the Δ component is immediate by the induction hypothesis. Completeness in the a component also follows straightforwardly from the induction hypothesis. Indeed, $\langle A \rangle^p(a^+, a_o^-) = a_o$ by the induction hypothesis, so $\langle \uparrow A \rangle^p(a^+, [a_o^-]) = [a_o]$ as desired.

Case ($\otimes$R)**:** Recall eqs. (128) to (130). Let $(\delta^+, b^+, (a_B^-, a_A^-))$ be arbitrary in the domain of eq. (130). Completeness in the Δ component and the A portion of the a component follows by the induction hypothesis. We must show completeness in the $b : B$ component and the B portion of

the a component. In particular, where $b = \langle B \rangle^p(b^+, a_B)$ and $(a_B^+, b^-) = \langle B \rangle(b)$, it is sufficient to show that:

$$\langle B \rangle^p(b^+, b^-) = b,$$
$$\langle B \rangle^p(a_B^+, a_B^-) = b.$$

Both of these equations hold by the definition of e-p-pair, monotonicity, antisymmetry. In the first case:

$$b = \langle B \rangle^p(a_B^+, b^-) \sqsubseteq \langle B \rangle^p(b^+, b^-) \sqsubseteq \langle B \rangle^p(b^+, a_B^-) = b.$$

In the second case:

$$b = \langle B \rangle^p(a_B^+, b^-) \sqsubseteq \langle B \rangle^p(a_B^+, a_B^-) \sqsubseteq \langle B \rangle^p(b^+, a_B^-) = b.$$

Case ($\multimap$R)**:** Recall eqs. (208), (235) and (237). Let (δ^+, a^-) be arbitrary in the domain of eq. (208). If $a^- = \bot$, then the proof is analogous to case (**1**L). If $a^- = [(b_o^+, a_o^-)]$, then the result follows easily by the induction hypothesis.

Case (ρ^+R)**:** Recall eqs. (182) and (188). Let (δ^+, a^-) be arbitrary in the domain of eq. (188), and set

$$(\delta, a) = [\![\Psi \,;\, \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u(\delta^+, \mathsf{Unfold}(a^-)).$$

Then

$$[\![\Psi \,;\, \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A]\!]u(\delta^+, a^-) = (\delta, \mathsf{Fold}([a])).$$

Completeness in the Δ component follows by the induction hypothesis. To show that it is complete in the a component, we must show that

$$\langle \rho\alpha.A \rangle^p(\langle \rho\alpha.A \rangle^+(\mathsf{Fold}([a]), a^-) = \mathsf{Fold}([a]).$$

By the induction hypothesis,

$$\langle [\rho\alpha.A/\alpha]A \rangle^p(\langle [\rho\alpha.A/\alpha]A \rangle^+(a), \mathsf{Unfold}(a^-)) = a.$$

By proposition 8.5.3 and corollary 4.3.6,

$$\langle \rho\alpha.A \rangle^+(\mathsf{Fold}([a])) = \mathsf{Fold}([\langle [\rho\alpha.A/\alpha]A \rangle^+(a)]).$$

By eq. (186),

$$\langle \rho\alpha.A \rangle^p = \mathsf{Fold} \circ (-)_\bot \langle [\rho\alpha.A/\alpha]A \rangle^p \circ \delta \circ (\mathsf{Unfold} \times \mathsf{Unfold}).$$

We compute, using the above identities:

$$\begin{aligned}
&\langle \rho\alpha.A \rangle^p(\langle \rho\alpha.A \rangle^+(\mathsf{Fold}([a]), a^-) \\
&= \langle \rho\alpha.A \rangle^p(\mathsf{Fold}([\langle [\rho\alpha.A/\alpha]A \rangle^+(a)]), a^-) \\
&= (\mathsf{Fold} \circ (-)_\bot \langle [\rho\alpha.A/\alpha]A \rangle^p \circ \delta \circ (\mathsf{Unfold} \times \mathsf{Unfold}))\,(\mathsf{Fold}([\langle [\rho\alpha.A/\alpha]A \rangle^+(a)]), a^-) \\
&= (\mathsf{Fold} \circ (-)_\bot \langle [\rho\alpha.A/\alpha]A \rangle^p)\,([(\langle [\rho\alpha.A/\alpha]A \rangle^+(a), \mathsf{Unfold}(a^-))]) \\
&= \mathsf{Fold}([\langle [\rho\alpha.A/\alpha]A \rangle^p(\langle [\rho\alpha.A/\alpha]A \rangle^+(a), \mathsf{Unfold}(a^-))]) \\
&= \mathsf{Fold}([a]).
\end{aligned}$$

Case (ρ^-R)**:** Recall eqs. (212) and (217). We proceed by case analysis on a^-: it is either $\bot$ or $\mathsf{Fold}([a_o^-])$ for some $a_o^- \in [\![\rho\alpha.A/\alpha]A]\!]^-$. If $a^- = \bot$, then the proof is analogous to case (**1**L). If $a^- = \mathsf{Fold}([a_o^-])$, the proof is analogous to the case (ρ^+R). Indeed, completeness in the Δ components follows by the induction hypothesis. To see completeness in the a component, set

$$(\delta, a) = [\![\Psi \,;\, \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u(\delta^+, a_o^-)$$

and observe that by the induction hypothesis,

$$\langle [\rho\alpha.A/\alpha]A \rangle^p(\langle [\rho\alpha.A/\alpha]A \rangle^+(a), a_o^-) = a.$$

We must show that

$$\langle \rho\alpha.A \rangle^p(\langle \rho\alpha.A \rangle^+(\mathsf{Fold}([a])), \mathsf{Fold}([a_o^-])) = \mathsf{Fold}([a]).$$

By proposition 8.5.3 and corollary 4.3.6,

$$\langle\rho\alpha.A\rangle^+(\mathsf{Fold}([a])) = \mathsf{Fold}(\langle[\rho\alpha.A/\alpha]A\rangle^+(a)).$$

By the negative analog of eq. (186),

$$\langle\rho\alpha.A\rangle^p = \mathsf{Fold}\circ(-)_\perp\langle[\rho\alpha.A/\alpha]A\rangle\circ\delta\circ(\mathsf{Unfold}\times\mathsf{Unfold}).$$

We now compute using the above identities:

$$\begin{aligned}
&\langle\rho\alpha.A\rangle^p(\langle\rho\alpha.A\rangle^+(\mathsf{Fold}([a])),\mathsf{Fold}([a_\circ^-]))\\
&\langle\rho\alpha.A\rangle^p\,(\mathsf{Fold}(\langle[\rho\alpha.A/\alpha]A\rangle^+(a)),\mathsf{Fold}([a_\circ^-]))\\
&= (\mathsf{Fold}\circ(-)_\perp\langle[\rho\alpha.A/\alpha]A\rangle\circ\delta)\,(\langle[\rho\alpha.A/\alpha]A\rangle^+(a),[a_\circ^-])\\
&= \mathsf{Fold}\,([\langle[\rho\alpha.A/\alpha]A\rangle\,(\langle[\rho\alpha.A/\alpha]A\rangle^+(a),a_\circ^-)])\\
&= \mathsf{Fold}\,([a])\,.
\end{aligned}$$

Case (E-{}): Recall eq. (143). The result follows from the fact that for all $u\in[\![\Psi]\!]$, $[\![\Psi\,;\,\overline{a_i:A_i}\vdash a\leftarrow\{M\}\leftarrow\overline{a}_i::a:A]\!]u$ is defined to be an element of a dcpo of complete functions. □

Lemma 8.4.11. *Let $f:A^+\to A$ be frugal relative to $\alpha:A\to A^+\times A^-$. Then, where β, β^+, and β^- are isomorphisms, $\beta^{-1}\circ f\circ(\beta^+)^{-1}:B^+\to B$ is frugal relative to*

$$B\xrightarrow{\beta}A\xrightarrow{\alpha}A^+\times A^-\xrightarrow{\beta^+\times\beta^-}B^+\times B^-.$$

Proof. Let $b_\circ^+\in B^+$ be arbitrary. The frugality system for $\beta^{-1}\circ f\circ(\beta^+)^{-1}$ and $b_\circ^+$ is:

$$\begin{aligned}
\left((\beta^+\circ\alpha^+\circ\beta)\circ((\beta^+\times\beta^-)\circ\alpha\circ\beta)^p\right)(b_\circ^+,b^-)&\sqsubseteq b^+\\
\left((\beta^-\circ\alpha^-\circ\beta)\circ\left(\beta^{-1}\circ f\circ(\beta^+)^{-1}\right)^p\right)(b^+)&\sqsubseteq b^-.
\end{aligned}$$

We must show that its least solution (b^+,b^-) is given by

$$\left(((\beta^+\times\beta^-)\circ\alpha\circ\beta)\circ\left(\beta^{-1}\circ f\circ(\beta^+)^{-1}\right)\right)(b_\circ^+).$$

Cancelling out inverses, we observe that minimizing (b^+,b^-) in this system is equivalent to minimizing it in the system

$$\begin{aligned}
\left(\beta^+\circ\alpha^+\circ\alpha^p\circ(\beta^+\times\beta^-)^{-1}\right)(b_\circ^+,b^-)&\sqsubseteq b^+\\
\left(\beta^-\circ\alpha^-\circ f\circ\left((\beta^+)^{-1}\right)^p\right)(b^+)&\sqsubseteq b^-.
\end{aligned}$$

The functions β, β^+, and β^- are isomorphisms, so (b^+,b^-) is the least solution to the above system if and only if $(a^+,a^-)=(\beta^+\times\beta^-)^{-1}(b^+,b^-)$ is the least solution to the system

$$\begin{aligned}
(\alpha^+\circ\alpha^p)\left((\beta^+)^{-1}(b_\circ^+),a^-\right)&\sqsubseteq a^+\\
(\alpha^-\circ f)(a^+)&\sqsubseteq a^-.
\end{aligned}$$

But f is frugal, so the least solution to this system is $(a^+,a^-)=(\alpha\circ f)\left((\beta^+)^{-1}(b_\circ^+)\right)$. It follows that:

$$\begin{aligned}
(b^+,b^-)&=((\beta^+\times\beta^-)\circ\alpha\circ f)\left((\beta^+)^{-1}(b_\circ^+)\right)\\
&=\left((\beta^+\times\beta^-)\circ\alpha\circ\beta\circ\beta^{-1}\circ f\circ(\beta^+)^{-1}\right)(b_\circ^+).
\end{aligned}$$

This is what we wanted to show. □

Recall the notation given in remark 8.2.1 for combining embeddings $\alpha:A\to A^+\times A^-$ and $\beta:B\to B^+\times B^-$ to form an embedding $(\alpha,\beta):A\times B\to(A^+\times B^+)\times(A^-\times B^-)$.

LEMMA 8.4.12. *Consider embeddings $\alpha : A \to A^+ \times A^-$ and $\beta : B \to B^+ \times B^-$. If $f : A^+ \times B^+ \to A \times B$ is frugal relative to (α, β), then the function F given by*

$$(\mathsf{id} \times \mathsf{up}) \circ f : A^+ \times B^+ \to A \times B_\perp$$

is frugal relative to the embedding γ given by

$$A \times B_\perp \xrightarrow{\alpha \times \beta_\perp} (A^+ \times A^-) \times (B^+ \times B^-)_\perp \xrightarrow{\mathsf{id} \times \delta^e} (A^+ \times A^-) \times (B^+ \times B^-_\perp) \cong (A^+ \times B^+) \times (A^- \times B^-_\perp)$$

where δ^e is given by lemma 2.2.50.

Proof. Let $(a^+_\circ, b^+_\circ) \in A^+ \times B^+$ be arbitrary. We show that the least solution $((a^+, b^+), (a^-, b^-))$ to the frugality system

$$(\gamma^+ \circ \gamma^p)\,((a^+_\circ, b^+_\circ), (a^-, b^-)) \sqsubseteq (a^+, b^+)$$
$$(\gamma^- \circ F)\,(a^+, b^+) \sqsubseteq (a^-, b^-)$$

is given by $(\gamma \circ F)(a^+_\circ, b^+_\circ)$. For convenience, we start by expanding and simplifying the expressions in this system. Observe that

$$(\gamma^+ \circ \gamma^p)\,((a^+_\circ, b^+_\circ), (a^-, b^-)) = ((\alpha^+ \circ \alpha^p)(a^+_\circ, a^-), (\mathsf{down} \circ \beta^+_\perp \circ \beta^p_\perp \circ \delta)(b^+_\circ, b^-))$$

and that if $((\alpha, \beta)^- \circ f)(a^+, b^+) = (a^-_1, b^-_1)$, then

$$(\gamma^- \circ F)(a^+, b^+) = (a^-_1, [b^-_1]).$$

We claim that the least solution to the above frugality system is $((a^+_1, b^+_1), (a^-_1, [b^-_1]))$, where $((a^+_1, b^+_1), (a^-_1, b^-_1))$ is the least solution to the frugality system

$$((\alpha, \beta)^+ \circ (\alpha, \beta)^p)\,((a^+_\circ, b^+_\circ), (a^-_1, b^-_1)) \sqsubseteq (a^+_1, b^+_1)$$
$$((\alpha, \beta)^- \circ f)\,(a^+_1, b^+_1) \sqsubseteq (a^-_1, b^-_1).$$

Indeed, by the above, it is a solution:

$$\begin{aligned}
&(\gamma^+ \circ \gamma^p)\,((a^+_\circ, b^+_\circ), (a^-_1, [b^-_1])) \\
&= ((\alpha^+ \circ \alpha^p)(a^+_\circ, a^-_1), (\mathsf{down} \circ \beta^+_\perp \circ \beta^p_\perp \circ \delta)(b^+_\circ, [b^-_1])) \\
&= ((\alpha^+ \circ \alpha^p)(a^+_\circ, a^-_1), \mathsf{down}([(\beta^+ \circ \beta^p)(b^+_\circ, b^-_1)])) \\
&= ((\alpha^+ \circ \alpha^p)(a^+_\circ, a^-_1), (\beta^+ \circ \beta^p)(b^+_\circ, b^-_1)) \\
&\sqsubseteq (a^+_1, b^+_1)
\end{aligned}$$

and $(\gamma^- \circ F)(a^+_1, b^+_1) = (a^-_1, [b^-_1])$. It is also least: any smaller solution would induce a smaller solution to the frugality system for f, contradicting the minimality of $((a^+_1, b^+_1), (a^-_1, b^-_1))$.

We check that it is given by $(\gamma \circ F)(a^+_\circ, b^+_\circ)$. By assumption,

$$((a^+_1, b^+_1), (a^-_1, b^-_1)) = ((\alpha, \beta) \circ f)(a^+_\circ, b^+_\circ).$$

In particular, where $(a_\circ, b_\circ) = f(a^+_\circ, b^+_\circ)$,

$$\alpha(a_\circ) = (a^+_1, a^-_1), \qquad \beta(b_\circ) = (b^+_1, b^-_1).$$

Finally, observe that

$$(\gamma \circ F)(a^+_\circ, b^+_\circ) = \gamma(a_\circ, [b_\circ]) = ((a^+_1, b^+_1), (a^-_1, [b^-_1])).$$

This is what we wanted to show. □

LEMMA 8.4.13. *Consider embeddings $\alpha : A \to A^+ \times A^-$ and $\beta : B \to B^+ \times B^-$. If $f : A^+ \times B^+ \to A \times B$ is frugal relative to (α, β), then the function $F : A^+ \times B^+_\perp \to A \times B_\perp$ given by*

$$F(a^+, b^+) = \begin{cases} (a, [b]) & \text{if } b^+ = [b^+_\circ] \\ (\alpha^p(a^+, \perp), \perp) & \text{if } b^+ = \perp \end{cases}$$
$$\text{where } (a, b) = f(a^+, b^+_\circ)$$

is frugal relative to the embedding γ given by

$$A \times B_\perp \xrightarrow{\alpha\times\beta_\perp} (A^+ \times A^-) \times (B^+ \times B^-)_\perp \xrightarrow{\mathsf{id}\times\delta^e} (A^+ \times A^-) \times (B^+_\perp \times B^-) \cong (A^+ \times B^+_\perp) \times (A^- \times B^-)$$

where δ^e is given by lemma 2.2.50.

Proof. An arbitrary element of $A^+ \times B^+_\perp$ is either of the form $(a^+_{\mathrm{o}}, \perp)$ or of the form $(a^+_{\mathrm{o}}, [b^+_{\mathrm{o}}])$ for some $(a^+_{\mathrm{o}}, b^+_{\mathrm{o}}) \in A^+ \times B^+$. We proceed by case analysis on these two possibilities. In each case, the frugality system is

$$(\gamma^+ \circ \gamma^p)\,((a^+_{\mathrm{o}}, \perp), (a^-, b^-)) \sqsubseteq (a^+, b^+)$$
$$(\gamma^- \circ F)\,(a^+, b^+) \sqsubseteq (a^-, b^-).$$

Case $(a^+_{\mathrm{o}}, \perp)$**:** We must show the least solution $((a^+, b^+), (a^-, b^-))$ to the frugality system is given by $(\gamma \circ F)(a^+_{\mathrm{o}}, \perp)$. Observe that $(\gamma \circ F)(a^+_{\mathrm{o}}, \perp) = ((a^+_1, \perp), (\perp, \perp)$ where $a^+_1 = (\alpha^+ \circ \alpha^p)(a^+_{\mathrm{o}}, \perp)$. We check that it is a solution:

$$\begin{aligned}
&(\gamma^+ \circ \gamma^p)\,((a^+_{\mathrm{o}}, \perp), (\perp, \perp)) \\
&= \gamma^+(\alpha^p(a^+_{\mathrm{o}}, \perp), \perp) \\
&= (a^+_1, \perp)
\end{aligned}$$

and

$$\begin{aligned}
&(\gamma^- \circ F)\,(a^+_1, \perp) \\
&= \gamma^-(\alpha^p(a^+_1, \perp), \perp) \\
&= ((\alpha^- \circ \alpha^p)(a^+_1, \perp), \perp) \\
&= (\perp, \perp).
\end{aligned}$$

It is also clearly minimum. This gives the result.

Case $(a^+_{\mathrm{o}}, [b^+_{\mathrm{o}}])$**:** We must show the least solution $((a^+, b^+), (a^-, b^-))$ to the frugality system is given by $(\gamma \circ F)(a^+_{\mathrm{o}}, [b^+_{\mathrm{o}}])$. By the first inequality in the frugality system, if $((a^+, b^+), (a^-, b^-))$ is the least solution, then $b^+ = [b^+_1]$ for some b^+_1. It follows that minimizing $((a^+, b^+), (a^-, b^-))$ in the frugality system is equivalent to minimizing $((a^+_1, b^+_1), (a^-_1, b^-_1))$ in the system

$$((\alpha, \beta)^+ \circ (\alpha, \beta)^p)\,((a^+_{\mathrm{o}}, b^+_{\mathrm{o}}), (a^-_1, b^-_1)) \sqsubseteq (a^+_1, b^+_1)$$
$$((\alpha, \beta)^- \circ f)\,(a^+_1, b^+_1) \sqsubseteq (a^-_1, b^-_1)$$

and taking $((a^+, b^+), (a^-, b^-)) = ((a^+_1, [b^+_1]), (a^-_1, b^-_1))$. But f was assumed to be frugal, so the least solution to this second system is

$$((a^+_1, b^+_1), (a^-_1, b^-_1)) = ((\alpha, \beta) \circ f)(a^+_{\mathrm{o}}, b^+_{\mathrm{o}}).$$

We check that

$$((a^+_1, [b^+_1]), (a^-_1, b^-_1)) = (\gamma \circ F)(a^+_{\mathrm{o}}, [b^+_{\mathrm{o}}]).$$

Set $(a_{\mathrm{o}}, b_{\mathrm{o}}) = f(a^+_{\mathrm{o}}, b^+_{\mathrm{o}})$. We compute:

$$\begin{aligned}
&(\gamma \circ F)(a^+_{\mathrm{o}}, [b^+_{\mathrm{o}}]) \\
&= \gamma(a_{\mathrm{o}}, [b_{\mathrm{o}}]) \\
&= ((a^+_1, [b^+_1]), (a^-_1, b^-_1)).
\end{aligned}$$

This is what we wanted to show. □

Proposition 8.4.14. *If $\Psi \,;\, \Delta \vdash P :: a : A$, then $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u$ is frugal for all $u \in [\![\Psi]\!]$.*

Proof. By induction on the derivation $\Psi \,;\, \Delta \vdash P :: a : A$. Recall the definition of frugality from definition 8.2.10. In each case, we must characterize the least solution to a system of inequalities on elements of products. Elements of products are ordered component-wise, so it is sufficient to characterize the least solution on a component-by-component basis. Most components (typically

the used channels) will follow immediately from the induction hypothesis, while some (typically the provided channel) will follow by a straightforward computation.

For convenience, we will often name intermediate values in the frugality system. In particular, to show that $p : \Delta^+ \to \Delta$ is frugal relative to $\Delta \to \Delta^+ \times \Delta^-$ given some $\delta_0^+ \in \Delta^+$, we will minimize (δ^+, δ^-) in the system

$$\Delta^p(\delta_0^+, \delta^-) = \delta_1 \qquad \Delta^+(\delta_1) \sqsubseteq \delta^+$$
$$p(\delta^+) = \delta_2 \qquad \Delta^-(\delta_2) \sqsubseteq \delta^-.$$

It is obvious that doing so is equivalent to minimizing (δ^+, δ^-) in the system given by definition 8.2.10.

We omit cases that follow easily by symmetry or by analogy with other cases.

Case (Fwd$^+$)**:** Recall eq. (113). It is immediate from the definitions that every well-woven embedding is frugal. By proposition 8.4.8, $\langle A \rangle$ is well-woven. The result is now clear.

Case (Cut)**:** Recall eq. (115):

$$[\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u = [\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]u \circ_a [\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!]u \quad (115)$$

Let $(\delta_1^+, \delta_2^+, c^-)$ be arbitrary in its domain, and let (a^+, a^-) be the witnesses for the above composition at $(\delta_1^+, \delta_2^+, c^-)$. Then

$$[\![\Psi \,;\, \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u(\delta_1^+, \delta_2^+, c^-) = (\delta_1, \delta_2, c)$$

where

$$[\![\Psi \,;\, \Delta_1 \vdash P :: a : A]\!]u(\delta_1^+, a^-) = (\delta_1, \_)$$
$$[\![\Psi \,;\, a : A, \Delta_2 \vdash Q :: c : C]\!]u(\delta_2^+, a^+, c^-) = (\delta_2, \_, c).$$

The result follows by the induction hypothesis.

Case (**1**R)**:** Recall eq. (122). The only element in its domain is $\bot$. Frugality is given by the fact that $(\mathsf{close}, \bot)$ is the minimum solution to (x^+, x^-) to the system[26]

$$\langle \mathbf{1} \rangle^p(x^+, \bot) = x_1, \qquad \langle \mathbf{1} \rangle^-(x_1) \sqsubseteq x^-,$$
$$[\![\Psi \,;\, \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}]\!]u x^- = x_2, \qquad \langle \mathbf{1} \rangle^+(x_2) \sqsubseteq x^+.$$

Indeed, $x_2 = \mathsf{close}$, so $x^+ = \mathsf{close}$, so $x_1 = \mathsf{close}$ and $x^- = \bot$.

Case (**1**L)**:** Recall eq. (123). Let (δ^+, a^+, c^-) be arbitrary in its domain. We proceed by case analysis on a^+. If $a^+ = \mathsf{close}$, then the least solution $((\delta_1^+, a_1^+, c_1^-), (\delta_2^-, a_2^-, c_2^+)$ to the system

$$\langle \Delta, a : \mathbf{1} \rangle^p((\delta^+, a^+), (\delta_2^-, a_2^-)) = (\delta_1, a_1) \qquad \langle \Delta, a : \mathbf{1} \rangle^+(\delta_1, a_1) \sqsubseteq (\delta_1^+, a_1^+)$$
$$\langle C \rangle^p(c_2^+, c^-) = c_1 \qquad \langle C \rangle^-(c_1) \sqsubseteq c_1^-$$
$$[\![\Psi \,;\, \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u(\delta_1^+, a_1^+, c_1^-) = (\delta_2, a_2, c_2) \qquad \langle \Delta, a : \mathbf{1} \rangle^-(\delta_2, a_2) \sqsubseteq (\delta_2^-, a_2^-)$$
$$\langle C \rangle^+(c_2) \sqsubseteq c_2^+$$

is $((\delta_1^+, \mathsf{close}, c_1^-), (\delta_2^-, \bot, c_2^+))$, where $((\delta_1^+, c_1^-), (\delta_2^-, c_2^+))$ is the least solution to the system of equations for (δ^+, c^-) and $[\![\Psi \,;\, \Delta \vdash P :: c : C]\!]u$. To see this, it is sufficient to note that $a^+ = \mathsf{close}$ implies that $a_1 = \mathsf{close}$ and $a_1^+ = \mathsf{close}$, and then expanding the definition of $[\![\Psi \,;\, \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u$. Frugality in the components Δ and $c : C$ follows by the induction hypothesis. Frugality in the component $a : \mathbf{1}$ follows from the fact that $[\![\Psi \,;\, \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-) = (\_, \mathsf{close}, \_)$ and that $\langle \mathbf{1} \rangle(\mathsf{close}) = (\mathsf{close}, \bot)$ is the minimal (a_1^+, a_2^-) satisfying the above system.

If $a^+ = \bot$, then $a_1 = \bot$, $a_1^+ = \bot$ and $a_2 = \bot$, so $\delta_2 = \langle \Delta \rangle^p(\delta_1^+, \bot)$ and $c_2 = \langle C \rangle^p(\bot, c_1^-)$ by eq. (122). By properties of e-p-pair, the minimal δ_2^- and c_2^+ satisfying the equations are then both $\bot$, and so $\delta_1 = \langle \Delta \rangle^p(\delta^+, \bot)$ and $c_1 = \langle C \rangle^p(\bot, c^-)$. The elements $\delta_1^+ = \langle \Delta \rangle^+(\delta_1)$ and $c_1^- = \langle C \rangle^-(c_1)$ are clearly the minimal elements satisfying the system. The result is now immediate from inspection of eq. (122).

[26] Note that polarities have appropriately been swapped relative to definition 8.2.10.

As an aside, we remark that the case $a^+ = \bot$ alternatively follows straightforwardly from propositions 8.2.16 and 8.4.10.

Case (⊕R): Recall eqs. (163) to (166). Fix some arbitrary (δ_o^+, a_o^-) in the domain of eq. (166). The least solution $((\delta^+, a^-), (\delta^-, a^+))$ to frugality system

$$\begin{aligned} \langle\Delta\rangle^p(\delta_o^+, \delta^-) &= \delta_1 & \langle\Delta\rangle^+(\delta_1) &\sqsubseteq \delta^+ \\ \langle\oplus\{l : A_l\}_{l\in L}\rangle^p(a^+, a_o^-) &= a_1 & \langle\oplus\{l : A_l\}_{l\in L}\rangle^-(a_1) &\sqsubseteq a^- \\ [\![\Psi\,;\,\Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u(\delta^+, a^-) &= (\delta_2, a_2) & \langle\Delta\rangle^-(\delta_2) &\sqsubseteq \delta^- \\ & & \langle\oplus\{l : A_l\}_{l\in L}\rangle^+(a_2) &\sqsubseteq a^+ \end{aligned}$$

is $((\delta^+, \iota_k(a_k^-)), (\delta^-, (k, [a_k^+])))$, where we take $((\delta^+, a_k^-), (\delta^-, a_k^+))$ to be the least solution $((\delta^+, a^-), (\delta^-, a^+))$ to the frugality system

$$\begin{aligned} \langle\Delta\rangle^p(\delta_o^+, \delta^-) &= \delta_1 & \langle\Delta\rangle^+(\delta_1) &\sqsubseteq \delta^+ \\ \langle A_k\rangle^p(a^+, \pi_k(a_o^-)) &= a_1 & \langle A_k\rangle^-(a_1) &\sqsubseteq a^- \\ [\![\Psi\,;\,\Delta \vdash P :: a : A_k]\!]u(\delta^+, a^-) &= (\delta_2, a_2) & \langle\Delta\rangle^-(\delta_2) &\sqsubseteq \delta^- \\ & & \langle A_k\rangle^+(a_2) &\sqsubseteq a^+. \end{aligned}$$

By the induction hypothesis, where $[\![\Psi\,;\,\Delta \vdash P :: a : A_k]\!]u(\delta_o^+, \pi_k(a_o^-)) = (\delta_o, a_o)$,

$$\begin{aligned} \langle\Delta\rangle(\delta_o) &= (\delta^+, \delta^-), \\ \langle A_k\rangle(a_o) &= (a_k^+, a_k^-). \end{aligned}$$

It is then easy to check that, where $[\![\Psi\,;\,\Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u(\delta_o^+, a_o^-) = (\delta_o, (k, [a_o]))$,

$$\begin{aligned} \langle\Delta\rangle(\delta_o) &= (\delta^+, \delta^-), \\ \langle\oplus\{l : A_l\}_{l\in L}\rangle((k, [a_o])) &= ((k, [a_k^+]), \iota_k(a_k^-)). \end{aligned}$$

This is what we wanted to show.

Case (&R): Recall eqs. (206) and (229) to (231). Let (δ^+, a^-) be arbitrary in the domain of eq. (206). When $a^- = \bot$, the proof is analogous to case (**1**L). If $a^- = (l, [a_l^-])$, then the least solution to the frugality system is $((\hat{\delta}^+, (l, [\hat{a}_l^-])), (\delta^-, \iota_l(a_l^+)))$ where $((\hat{\delta}^+, \hat{a}_l^-), (\delta^-, a_l^+))$ is the least solution to the frugality system for $[\![\Psi\,;\,\Delta \vdash P_l :: a : A_l]\!]u$ and (δ^+, a_l^-). By the induction hypothesis, where $[\![\Psi\,;\,\Delta \vdash P_l :: a : A_l]\!]u(\delta^+, a_l^-) = (\delta, a_l)$,

$$\begin{aligned} \langle\Delta\rangle(\delta) &= (\hat{\delta}^+, \delta^-), \\ \langle A_l\rangle(a_l) &= (a_l^+, \hat{a}_l^-). \end{aligned}$$

It is then easy to check, where $[\![\Psi\ ;\ \Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u(\delta^+, a^-) = (\delta, (l, [a_l]))$, that

$$\begin{aligned} \langle\Delta\rangle(\delta) &= (\hat{\delta}^+, \delta^-), \\ \langle\&\{l : A_l\}_{l\in L}\rangle((l, [a_l])) &= (\iota_l(a_l^+), (l, [\hat{a}_l^-])). \end{aligned}$$

This is what we wanted to show.

Case (⊗R): Recall eqs. (127) to (130). Let $(\delta^+, b^+, (a_B^-, a_A^-))$ be arbitrary in the domain of eq. (130). The frugality system is:

$$\begin{aligned} \langle\Delta, b : B, a : B \otimes A\rangle^p((\delta^+, b^+)), (\delta_1^-, b_1^-)) &= (\delta_1, b_1) \\ \langle a : B \otimes A\rangle^p(a_1^+, (a_B^-, a_A^-)) &= a_1 \\ [\![\Psi\,;\,\Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A]\!]u(\delta_2^+, b_2^+, (a_{2B}^-, a_{2A}^-)) &= (\delta_2, b_2, [(b_2, a_2)]) \\ \langle\Delta, b : B\rangle^+(\delta_1, b_1) \sqsubseteq (\delta_2^+, b_2^+) \qquad \langle B \otimes A\rangle^-(a_1) &\sqsubseteq (a_{2B}^-, a_{2A}^-) \\ \langle\Delta, b : B\rangle^-(\delta_2, b_2) \sqsubseteq (\delta_2^-, b_2^-) \quad \langle B \otimes A\rangle^+([(b_2, a_2)]) &\sqsubseteq a_1^+. \end{aligned}$$

We deduce from the definition of $\langle B \otimes A\rangle^+$ that $a_1^+ = [(a_B^+, a_A^+)]$ for some a_B^+ and a_A^+. It follows that

$$a_1 = [(\langle B\rangle^p(a_B^+, a_B^-), \langle A\rangle^p(a_A^+, a_A^-))].$$

We recognize the equations involving $\langle B\rangle$ as an instance of the system given in lemma 8.2.9. Accordingly, the least solution $((\delta_2^+, b_2^+, (a_{2B}^-, a_{2A}^-)), (\delta_1^-, b_1^-, [(a_B^+, a_A^+)]))$ to the system is

$$((\delta_2^+, \beta^+, (\beta^-, \alpha^-)), (\delta_1^-, \beta^-, [(\beta^+, \alpha^+)]))$$

where $(\langle B\rangle \circ \langle B\rangle^p)(b^+, a_B^-) = (\beta^+, \beta^-)$, and where $((\delta_2^+, \alpha^-), (\delta_1^-, \alpha^+))$ is the least solution to the frugality system for $[\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u$ and (δ^+, a_A^-). We compute that

$$\begin{aligned}&(\langle \Delta, b : B, a : B \otimes A\rangle \circ [\![\Psi\ ;\ \Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A]\!]u)(\delta^+, b^+, (a_B^-, a_A^-))\\ &= ((\hat{\delta}^+, \beta^+, [(\beta^+, a^+)]), (\delta^-, \beta^-, (\beta^-, \hat{a}_A^-)))\end{aligned}$$

where

$$(\langle \Delta, a : A\rangle \circ [\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u)(\delta^+, a_A^-) = ((\hat{\delta}^+, a^+), (\delta^-, \hat{a}_A^-)).$$

By the induction hypothesis,

$$(\langle \Delta, a : A\rangle \circ [\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u)(\delta^+, a_A^-) = ((\delta_2^+, \alpha^+), (\delta_1^-, \alpha^-)).$$

Taking the collection of these equations, we conclude the result.

Case ($\multimap$R)**:** Recall eqs. (208) and (235) to (237). Let (δ^+, a^-) be arbitrary in the domain. If $a^- = \bot$, then the proof is analogous to case (**1**L). If $a^- = [(b_0^+, a_0^-)]$, then the result follows easily by the induction hypothesis.

Case ($\supset$R)**:** Recall eqs. (211) and (241) to (243). The result follows straightforwardly from the induction hypothesis. We illustrate the case nevertheless. Let (δ_0^+, a_0^-) be arbitrary in the domain. If $a_0^- = \bot$, then the proof is analogous to case (**1**L). If $(a_0^- = (v, [a^-])$, then the completeness system is:

$$\begin{aligned}\langle \Delta, a : \tau \supset A\rangle^p((\delta_0^+, a_0^-), (\delta^-, a^+)) &= (\delta_1, a_1)\\ [\![\Psi\ ;\ \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u(\delta_1^+, a_1^-) &= (\delta_2, a_2)\\ \langle \Delta\rangle^+(\delta_1) \sqsubseteq \delta^+ \quad \langle \tau \supset A\rangle^-(a_1) &\sqsubseteq a_1^-\\ \langle \Delta\rangle^-(\delta_2) \sqsubseteq \delta^- \quad \langle \tau \supset A\rangle^+(a_2) &\sqsubseteq a^+.\end{aligned}$$

We seek to minimize $(\delta^-, a^+, \delta_1^+, a_1^-)$, and show that this solution is appropriately related to $([\![\Psi\ ;\ \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u)_*(\delta_0^+, a_0^-)$. Expanding the definitions of $\langle \Delta, a : \tau \supset A\rangle^p$ and $[\![\Psi\ ;\ \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u$, we see that minimizing $(\delta^-, a^+, \delta_1^+, a_1^-)$ is equivalent to minimizing $(\delta^-, a^+, \delta_1^+, a_3^-)$ in

$$\begin{aligned}\langle \Delta, a : A\rangle^p((\delta_0^+, a^-), (\delta^-, a^+)) &= (\delta_1, a_3)\\ [\![\Psi, x : \tau\ ;\ \Delta \vdash P :: a : A]\!][u \mid x \mapsto v](\delta_1^+, [a_3^-]) &= (\delta_2, a_2)\\ \langle \Delta\rangle^+(\delta_1) \sqsubseteq \delta^+ \quad \langle A\rangle^-(a_3) &\sqsubseteq a_3^-\\ \langle \Delta\rangle^-(\delta_2) \sqsubseteq \delta^- \quad \langle A\rangle^+(a_2) &\sqsubseteq a^+.\end{aligned}$$

Indeed, given a minimum solution $(\delta^-, a^+, \delta_1^+, a_3^-)$ to the second system, the minimum solution to the first system is $(\delta^+, a^+, \delta_1^+, (v, [a_3^-]))$. By the induction hypothesis, this minimum solution satisfies

$$([\![\Psi, x : \tau\ ;\ \Delta \vdash P :: a : A]\!][u \mid x \mapsto v])_*(\delta_0^+, a^-) = (((\delta_1^+, a_3^-), (\delta^-, a^+)), \_).$$

But

$$([\![\Psi\ ;\ \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u)_*(\delta_0^+, a_0^-) = (((\delta_1^+, (v, [a_3^-])), (\delta^-, a^+)), \_).$$

This is exactly what we wanted to show.

Case (ρ^+R)**:** Recall eqs. (182), (183) and (188). By applying lemma 8.4.12 to the induction hypothesis, we deduce that

$$(\mathsf{id} \times \mathsf{up}) \circ [\![\Psi \,;\, \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u$$

is frugal relative to the embedding

$$\langle\Delta\rangle, (\delta^e \circ (-)_\perp \langle [\rho\alpha.A/\alpha]A \rangle) : [\![\Delta]\!] \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!] \to \\ \to ([\![\Delta]\!]^+ \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!]^+) \times ([\![\Delta]\!]^- \times [\![[\rho\alpha.A/\alpha]A]\!]^-).$$

By lemma 8.4.11,

$$(\mathsf{id} \times (a : \mathsf{Fold} \circ \mathsf{up})) \circ [\![\Psi \,;\, \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u \circ (\mathsf{id} \times (a^- : \mathsf{Unfold}))$$

is frugal relative to the embedding

$$[\![\Delta]\!] \times [\![\rho\alpha.A]\!] \xrightarrow{\mathsf{id}\times\mathsf{Unfold}} [\![\Delta]\!] \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!] \\ \xrightarrow{\langle\Delta\rangle, (\delta^e \circ (-)_\perp \langle [\rho\alpha.A/\alpha]A \rangle)} ([\![\Delta]\!]^+ \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!]^+) \times ([\![\Delta]\!]^- \times [\![[\rho\alpha.A/\alpha]A]\!]^-) \\ \xrightarrow{(\mathsf{id}\times\mathsf{Fold})\times(\mathsf{id}\times\mathsf{Fold})} ([\![\Delta]\!]^+ \times [\![\rho\alpha.A]\!]^+) \times ([\![\Delta]\!]^- \times [\![\rho\alpha.A]\!]^-).$$

We recognize this embedding as $(\langle\Delta\rangle, \langle\rho\alpha.A\rangle)$ by eq. (186).

Case (ρ^-R)**:** Recall eqs. (212), (217) and (218). By applying lemma 8.4.13 to the induction hypothesis, we deduce that

$$F(\delta^+, a^-) = \begin{cases} (\delta, [a]) & \text{if } a^- = [a_o^-] \\ (\langle\Delta\rangle^p(\delta^+, \perp), \perp) & \text{otherwise} \end{cases} \\ \text{where } (\delta, a) = [\![\Psi \,;\, \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u(\delta^+, a_o^-)$$

is sound relative to the embedding

$$\langle\Delta\rangle, (\delta^e \circ (-)_\perp \langle [\rho\alpha.A/\alpha]A \rangle) : [\![\Delta]\!] \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!] \to \\ \to ([\![\Delta]\!]^+ \times [\![[\rho\alpha.A/\alpha]A]\!]^+) \times ([\![\Delta]\!]^- \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!]^-).$$

By lemma 8.4.11,

$$\begin{cases} (\delta, \mathsf{Fold}([a])) & \text{if } a^- = \mathsf{Fold}([a_o^-]) \\ (\langle\Delta\rangle^p(\delta^+, \perp), \perp) & \text{otherwise} \end{cases} \\ \text{where } (\delta, a) = [\![\Psi \,;\, \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u(\delta^+, a_o^-)$$

is frugal relative to the embedding

$$[\![\Delta]\!] \times [\![\rho\alpha.A]\!] \xrightarrow{\mathsf{id}\times\mathsf{Unfold}} [\![\Delta]\!] \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!] \\ \xrightarrow{\langle\Delta\rangle, (\delta^e \circ (-)_\perp \langle [\rho\alpha.A/\alpha]A \rangle)} ([\![\Delta]\!]^+ \times [\![[\rho\alpha.A/\alpha]A]\!]^+) \times ([\![\Delta]\!]^- \times (-)_\perp [\![[\rho\alpha.A/\alpha]A]\!]^-) \\ \xrightarrow{(\mathsf{id}\times\mathsf{Fold})\times(\mathsf{id}\times\mathsf{Fold})} ([\![\Delta]\!]^+ \times [\![\rho\alpha.A]\!]^+) \times ([\![\Delta]\!]^- \times [\![\rho\alpha.A]\!]^-).$$

We recognize this embedding as $(\langle\Delta\rangle, \langle\rho\alpha.A\rangle)$ by the negative analog of eq. (186).

Case (E-{})**:** Recall eq. (143). The result follows from the fact that for all $u \in [\![\Psi]\!]$, $[\![\Psi \,;\, \overline{a_i : A_i} \vdash a \leftarrow \{M\} \leftarrow \overline{a}_i :: a : A]\!]u$ is defined to be an element of a dcpo of frugal functions. □

Corollary 8.4.15. *If* $\Psi \,;\, \Delta \vdash P :: a : A$*, then* $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u$ *is stable for all* $u \in [\![\Psi]\!]$.

Proof. Stability is immediate by the definition of junk-freedom, proposition 8.4.9, and the fact that upper-adjoints preserve existing infima (proposition 2.2.19). □

Recall from corollary 8.2.23 the DCPO $\mathbf{JFC}[\Delta \to \Psi]$ of junk-free, complete, frugal, stable functions $\Delta \to \Psi$ in $\mathbf{CYO}(\mathbf{Stab}_\perp)$.

Proposition 8.4.16. *If $\Psi \Vdash M : \tau$, then $[\![\Psi \Vdash M : \tau]\!]$ is continuous. It is stable if $\Psi \Vdash M : \tau$ does not use (*I-{}*) or any variables whose type involves (*T{}*). If $\Psi \,;\, \Delta \vdash P :: a : A$, then*

$$[\![\Psi \,;\, \Delta \vdash P :: a : A]\!] : [\![\Psi]\!] \to \mathbf{JFC}\,[\Delta \to a : A]$$

is continuous.

Proof. By induction on the derivation. The proofs for the cases in the functional layer are routine (see, e.g., [Gun92]), apart for:

Case (I-{}): Recall eq. (142). Continuity is immediate by the induction hypothesis. Stability is vacuous.

In the majority of cases for the process layer:

- Continuity of $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u$ follows from the induction hypothesis and the fact that continuous functions are closed under composition.
- Stability of $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u$ is a corollary of propositions 2.2.19 and 8.4.9: junk-free functions are projections, and projections preserve existing infima.
- Junk-freedom, completeness, and frugality of $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u$ are given by propositions 8.4.9, 8.4.10 and 8.4.14.
- The function $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u$ is then a morphism in $\mathbf{CYO}(\mathbf{Stab}_\perp)$ by corollary 8.2.15.
- Continuity of $[\![\Psi \,;\, \Delta \vdash P :: a : A]\!]$ follows from the induction hypothesis.

The interesting cases in the process layer are:

Case (Fwd$^+$): Recall eq. (113). Constant functions are continuous, so $[\![\Psi \;;\; a : A \vdash a \to b :: b : A]\!]$ is continuous. Continuity of $[\![\Psi \;;\; a : A \vdash a \to b :: b : A]\!]u$ is a consequence of proposition 8.4.6.

Case ($\wedge$R): Recall eq. (150). We show that $[\![\Psi \;;\; \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]$ is continuous. Let $U \subseteq [\![\Psi]\!]$ be directed. We must show that

$$\begin{aligned}&[\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]\left(\textstyle\bigsqcup^\uparrow U\right)\\ &= \textstyle\bigsqcup^\uparrow [\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]U.\end{aligned}$$

We consider two cases. Assume first that $[\![\Psi \Vdash M : \tau]\!]\left(\bigsqcup^\uparrow U\right) = \perp$. Then $[\![\Psi \Vdash M : \tau]\!]u = \perp$ for all $u \in U$ by monotonicity, and so $[\![\Psi \;;\; \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]u = (\langle\Delta\rangle^p(\delta^+, \perp), \langle\tau \wedge A\rangle^p(\perp, a^-))$ for all $u \in U$. Continuity is now clear:

$$\begin{aligned}&[\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]\left(\textstyle\bigsqcup^\uparrow U\right)\\ &= (\langle\Delta\rangle^p(\delta^+, \perp), \langle\tau \wedge A\rangle^p(\perp, a^-))\\ &= \textstyle\bigsqcup^\uparrow_{u\in U}(\langle\Delta\rangle^p(\delta^+, \perp), \langle\tau \wedge A\rangle^p(\perp, a^-))\\ &= \textstyle\bigsqcup^\uparrow_{u\in U} [\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]u.\end{aligned}$$

Otherwise, assume that $[\![\Psi \Vdash M : \tau]\!]\left(\bigsqcup^\uparrow U\right) \neq \perp$. Then the set $U' = \{u \in U \mid [\![\Psi \Vdash M : \tau]\!]u \neq \perp\}$ is non-empty. Recall that in general, if M is directed, then $\bigsqcup^\uparrow M = \bigsqcup^\uparrow(M \cup \{\perp\})$. It follows that

$$\begin{aligned}&[\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]\left(\textstyle\bigsqcup^\uparrow U\right)\\ &= [\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]\left(\textstyle\bigsqcup^\uparrow U'\right)\end{aligned}$$

and by proposition 8.2.24 that

$$\begin{aligned}&\textstyle\bigsqcup^\uparrow [\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]U\\ &= \textstyle\bigsqcup^\uparrow [\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]U'.\end{aligned}$$

It is thus sufficient to show that:

$$\begin{aligned}&[\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]\left(\textstyle\bigsqcup^\uparrow U'\right)\\ &= \textstyle\bigsqcup^\uparrow [\![\Psi \,;\, \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]U'.\end{aligned}$$

But by restricting our attention to U', we have eliminated the case analysis in the definition of $[\![\Psi \;;\; \Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]$. We recognize it as the composition of continuous functions, and we conclude the result.

Case ($\supset$R)**:** Recall eq. (211). Continuity of $[\![\Psi \;;\; \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]$ follows from the induction hypothesis and closure of continuous functions under composition: it is the composition of $[\![\Psi, x : \tau \;;\; \Delta \vdash P :: a : A]\!]$ with pairing and application. Continuity of $[\![\Psi \;;\; \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u$ follows similarly.

Case (ρ^+R)**:** Recall eq. (188). The functions Fold and Unfold are isomorphisms by proposition 4.3.4, so they are continuous. The result then follows from the induction hypothesis, and the fact that continuous functions are closed under composition.

Case (E-{})**:** Recall eq. (143). By the induction hypothesis and eq. (141). □

Proposition 8.4.17. *If $\Psi \Vdash M : \tau$, then the interpretation $[\![\Psi \Vdash M : \tau]\!]$ is natural in its environment. If $\Psi \;;\; \Delta \vdash P :: a : A$, then the interpretation $[\![\Psi \;;\; \Delta \vdash P :: a : A]\!]$ is natural in its environment.*

Proof. By case analysis on the last rule in the derivation of $\Psi \Vdash M : \tau$ and $\Psi \;;\; \Delta \vdash P :: a : A$. In the functional layer, we use $\mathbf{C}$ to range over $\mathbf{DCPO}_\perp$ or $\mathbf{Stab}_\perp$, depending on whether or not $[\![\Psi \Vdash M : \tau]\!]$ is stable.[27] We repeatedly use the following consequence of the Yoneda lemma [Rie16, chap. 2]: if $f : A \to B$ is a morphism of $\mathbf{D}$, then $\mathbf{D}(-, f)$ is a natural transformation $\mathbf{D}(-, A) \Rightarrow \mathbf{D}(-, B)$. We omit cases that follow by analogy from others.

Case (I-{})**:** Recall eq. (142). The corresponding natural interpretation is:

$$\mathbf{DCPO}_\perp(-, \mathsf{up}) : \mathbf{DCPO}_\perp(-, \mathbf{JFC}(\overline{a_i : A_i}, a : A)) \Rightarrow \mathbf{DCPO}_\perp(-, [\![\{a : A \leftarrow \overline{a_i : A_i}\}]\!]).$$

Case (F-Var)**:** Recall eq. (134). The corresponding natural interpretation is the family:

$$\left(\lambda \_ . \lambda u \in [\![\Psi, x : \tau]\!] . \pi_x^{\Psi, x} u\right)_{[\![\Psi]\!]} : \{*\} \to \mathbf{C}([\![\Psi, x : \tau]\!], [\![\tau]\!])$$

Case (F-Fix)**:** Recall eq. (137). Observe that

$$\begin{aligned} &[\![\Psi \Vdash \mathsf{fix}\, x . M : \tau]\!] u \\ &= [\![\Psi, x : \tau \Vdash M : \tau]\!]^\dagger u \\ &= \mathrm{lfp}\,(\lambda v \in [\![\tau]\!] . [\![\Psi, x : \tau \Vdash M : \tau]\!][u \mid x \mapsto v]) \\ &= (\mathrm{lfp} \circ \Lambda\,([\![\Psi, x : \tau \Vdash M : \tau]\!]))\,(u), \end{aligned}$$

where Λ is the currying natural isomorphism given by the adjunction for the exponential. The corresponding natural interpretation is then:

$$\mathbf{C}(-, \mathrm{lfp}) \circ \Lambda : \mathbf{C}(- \times [\![x : \tau]\!], \tau) \Rightarrow \mathbf{C}(-, \tau).$$

Case (F-Fun)**:** Recall eq. (135). Observe that

$$[\![\Psi \Vdash \lambda x : \tau . M : \tau \to \sigma]\!] = \mathsf{up} \circ \mathrm{strict} \circ \Lambda([\![\Psi, x : \tau \Vdash M : \sigma]\!]),$$

where Λ is the currying natural isomorphism.[28] The corresponding natural interpretation is:

$$\mathbf{C}(-, \mathsf{up} \circ \mathrm{strict}) \circ \Lambda : \mathbf{C}(- \times [\![x : \tau]\!], [\![\sigma]\!]) \Rightarrow \mathbf{C}(-, [\![\tau \to \sigma]\!]).$$

Case (F-App)**:** Recall eq. (136). There is a canonical natural isomorphism (see [Rie16, § 3.4])

$$\alpha : \mathbf{C}(-, [\![\tau \to \sigma]\!]) \times \mathbf{C}(-, \tau) \Rightarrow \mathbf{C}(-, [\![\tau \to \sigma]\!] \times [\![\tau]\!])$$

whose D-component is $\alpha_D(m, n)(u) = (mu, nu)$. The counit ev of the exponential adjunction is a natural transformation whose $[\![\tau]\!], [\![\sigma]\!]$ component is

$$\mathsf{ev}_{[\![\tau]\!], [\![\sigma]\!]} : \mathbf{C}\,[[\![\tau]\!] \to [\![\sigma]\!]] \times [\![\tau]\!] \to [\![\sigma]\!].$$

---

[27] Recall that it is assumed to be stable if it does not use (I-{}) or any variables whose types involve (T{}).

[28] In contrast to the previous case, we are here taking the Λ that is the right closure [Rie16, p. 129] of the cartesian product. Concretely, in this case $\Lambda : \mathbf{C}(- \times [\![x : \tau]\!], [\![\sigma]\!]) \Rightarrow \mathbf{C}(-, \mathbf{C}\,[[\![\tau]\!] \to [\![\sigma]\!]])$.

sending (f, v) to $f(v)$. The corresponding natural interpretation for (F-App) is then

$$\mathbf{C}(-, \mathsf{ev}_{[\![\tau]\!],[\![\sigma]\!]} \circ (\mathsf{down} \times \mathsf{id})) \circ \alpha : \mathbf{C}(-, [\![\tau \to \sigma]\!]) \times \mathbf{C}(-, [\![\tau]\!]) \Rightarrow \mathbf{C}(-, [\![\sigma]\!]).$$

Case (F-S): Recall eq. (140). Let $f : [\![\mathbf{nat}]\!] \to [\![\mathbf{nat}]\!]$ be given by

$$f(x) = \begin{cases} \bot & \text{if } x = \bot \\ x + 1 & \text{otherwise} \end{cases}$$

The natural interpretation for (F-S) is $\mathbf{C}(-, f) : \mathbf{C}(-, [\![\mathbf{nat}]\!]) \Rightarrow \mathbf{C}(-, [\![\mathbf{nat}]\!])$.

Case (Fwd$^+$): Recall eq. (113). The natural interpretation is

$$(\lambda\_.\lambda\_ \in [\![\Psi]\!].\langle a : \langle A \rangle^p, b : \langle A \rangle^p \rangle)_{[\![\Psi]\!]} : \{*\} \Rightarrow \mathbf{DCPO}_\bot([\![\Psi]\!], \mathbf{JFC}[\Delta \to a : A]).$$

This family is natural is because it is a constant family.

Case (Cut): Recall eq. (115). Composition of morphisms $p : \Delta_1 \to A$ and $q : \Delta_2, A \to C$ in $\mathbf{CYO}(\mathbf{Stab}_\bot)$ determines a continuous operation $\circ_A : \mathbf{JFC}[\Delta_1 \to A] \times \mathbf{JFC}[\Delta_2, A \to C] \to \mathbf{JFC}[\Delta_1, \Delta_2 \to C]$. There exists a canonical natural isomorphism

$$\begin{aligned} \alpha : \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta_1]\!] \to [\![a : A]\!]]) \times \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta_2, a : A]\!] \to [\![C]\!]]) \Rightarrow \\ \Rightarrow \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta_1]\!] \to [\![a : A]\!]] \times \mathbf{JFC}[[\![\Delta_2, a : A]\!] \to [\![C]\!]]). \end{aligned}$$

The natural interpretation is

$$\begin{aligned} \mathbf{DCPO}_\bot(-, \circ_a) \circ \alpha : \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta_1]\!] \to [\![a : A]\!]]) \times \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta_2, a : A]\!] \to [\![C]\!]]) \Rightarrow \\ \Rightarrow \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta_1, \Delta_2]\!] \to [\![C]\!]]). \end{aligned}$$

Case ($\mathbf{1}$R): Recall eq. (122). The natural interpretation is

$$(\lambda\_.\lambda\_ \in [\![\Psi]\!].\mathsf{close})_{[\![\Psi]\!]} : \{*\} \Rightarrow \mathbf{DCPO}_\bot(-, \mathbf{JFC}[\cdot \to [\![a : \mathbf{1}]\!]]).$$

This family is natural is because it is a constant family.

Case ($\mathbf{1}$L): Recall eq. (123). Let $f : \mathbf{JFC}[[\![\Delta]\!] \to [\![c : C]\!]] \to \mathbf{JFC}[[\![\Delta, a : \mathbf{1}]\!] \to [\![c : C]\!]]$ be the continuous function given by

$$f(p)(\delta^+, a^+, c^-) = \begin{cases} (\delta, \mathsf{close}, c) & \text{if } a^+ = \mathsf{close} \\ (\langle \Delta \rangle^p(\delta^+, \bot), \bot, \langle C \rangle^p(\bot, c^-)) & \text{otherwise} \end{cases}$$
$$\text{where } (\delta, c) = p(\delta^+, c^-).$$

The natural interpretation is then

$$\mathbf{DCPO}_\bot(-, f) : \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta]\!] \to [\![c : C]\!]]) \Rightarrow \mathbf{DCPO}_\bot(-, \mathbf{JFC}[[\![\Delta, a : \mathbf{1}]\!] \to [\![c : C]\!]]).$$

Case ($\wedge$R): Recall eq. (150). Let $f : [\![\tau]\!] \times \mathbf{JFC}[[\![\Delta]\!] \to [\![a : A]\!]] \to \mathbf{JFC}[[\![\Delta]\!] \to [\![a : \tau \wedge A]\!]]$ be given by

$$f(v, p)(\delta^+, a^-) = \begin{cases} (\delta, (v, [a])) & \text{if } v \neq \bot \\ (\langle \Delta \rangle^p(\delta^+, \bot), \langle \tau \wedge A \rangle^p(\bot, a^-)) & \text{if } v = \bot \end{cases}$$
$$\text{where } p(\delta^+, a^-) = (\delta, a).$$

The proof that is continuous closely follows the proof given in proposition 8.4.16. Let $V \subseteq [\![\tau]\!]$ and $P \subseteq \mathbf{JFC}[[\![\Delta]\!] \to [\![a : A]\!]]$ be directed. We must show that

$$f\left(\bigsqcup^\uparrow V, \bigsqcup^\uparrow P\right) = \bigsqcup^\uparrow f(V, P).$$

We consider two cases. Assume first that $\bigsqcup^\uparrow V = \bot$. Then $v = \bot$ for all $v \in V$, and

$$f(v, p)(\delta^+, a^-) = (\langle \Delta \rangle^p(\delta^+, \bot), \langle \tau \wedge A \rangle^p(\bot, a^-))$$

for all $v \in V$ and all p. It follows that

$$f\left(\bigsqcup^\uparrow V, \bigsqcup^\uparrow P\right) = (\langle \Delta \rangle^p(\delta^+, \bot), \langle \tau \wedge A \rangle^p(\bot, a^-)) = \bigsqcup^\uparrow f(\bot, P) = \bigsqcup^\uparrow f(V, P).$$

Otherwise, assume that $\bigsqcup^{\uparrow} V \neq \bot$. Then the set $V' = V \setminus \{\bot\}$ is non-empty. Recall that in general, if M is directed, then $\bigsqcup^{\uparrow} M = \bigsqcup^{\uparrow}(M \cup \{\bot\})$. It follows that

$$f(\bigsqcup^{\uparrow} V, \bigsqcup^{\uparrow} P) = f(\bigsqcup^{\uparrow} V', \bigsqcup^{\uparrow} P)$$

and by proposition 8.2.24 that $\bigsqcup^{\uparrow} f(V, P) = \bigsqcup^{\uparrow} f(V', P)$. It is thus sufficient to show that:

$$f(\bigsqcup^{\uparrow} V', \bigsqcup^{\uparrow} P) = \bigsqcup^{\uparrow} f(V', P).$$

But by restricting our attention to U', we have eliminated the case analysis in the definition of f. We now recognize f as the composition of continuous functions, so it is continuous.

The natural interpretation is then

$$\mathbf{DCPO}_{\bot}(-, f) \circ \alpha : \mathbf{DCPO}_{\bot}(-, [\![\tau]\!]) \times \mathbf{DCPO}_{\bot}(-, \mathbf{JFC}\,[\Delta \to [\![a : A]\!]]) \Rightarrow \\ \Rightarrow \mathbf{DCPO}_{\bot}(-, \mathbf{JFC}\,[\Delta \to [\![a : \tau \wedge A]\!]])$$

where α is the canonical natural isomorphism

$$\alpha : \mathbf{DCPO}_{\bot}(-, [\![\tau]\!]) \times \mathbf{DCPO}_{\bot}(-, \mathbf{JFC}\,[\Delta \to [\![a : A]\!]]) \Rightarrow \mathbf{DCPO}_{\bot}(-, [\![\tau]\!] \times \mathbf{JFC}\,[\Delta \to [\![a : A]\!]]).$$

The remaining cases follow by analogy with one of the previous cases. □

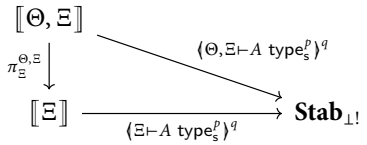
8.5. Semantic Properties

We show that the denotations of types, terms, and processes satisfy various structural properties.

8.5.1. Semantic Properties of Types. We show that the denotations of types respect the structural properties. It is immediate that they respect the exchange rule: contexts of type variables denote indexed products, so $[\![\Xi]\!] = [\![\Xi']\!]$ and $[\![\Xi \vdash A\ \mathsf{type_s}]\!] = [\![\Xi' \vdash A\ \mathsf{type_s}]\!]$ whenever Ξ' is a permutation of Ξ.

Weakening is semantically well-behaved, i.e., the semantic clauses are coherent [Ten95, p. 218]:

Proposition 8.5.1 (Coherence). *Let Θ, Ξ be a context of type variables. If $\Xi \vdash A\ \mathsf{type}^p_{\mathsf{s}}$, then the following diagram commutes in $\mathbf{Cell}_{\mathbf{CFP}}$ for $q \in \{-, +\}$:*



Proof. By induction on the derivation of $\Xi \vdash A\ \mathsf{type}^q_{\mathsf{s}}$.

Case (CVar)**:** Recall eqs. (168) to (172). We compute, using the definitions of products and horizontal composition:

$$\begin{aligned}
&\langle \Theta, \Xi, \alpha\ \mathsf{type_s} \vdash \alpha\ \mathsf{type_s} \rangle^q \\
&= \mathsf{id}_{\pi^{\Theta,\Xi,\alpha\alpha}_{\alpha}} : \pi^{\Theta,\Xi,\alpha}_{\alpha} \Rightarrow \pi^{\Theta,\Xi,\alpha}_{\alpha} \\
&= \mathsf{id}_{\pi^{\Xi,\alpha}_{\alpha}\pi^{\Theta,\Xi,\alpha}_{\Xi,\alpha}} : \pi^{\Xi,\alpha}_{\alpha}\pi^{\Theta,\Xi,\alpha}_{\Xi,\alpha} \Rightarrow \pi^{\Xi,\alpha}_{\alpha}\pi^{\Theta,\Xi,\alpha}_{\Xi,\alpha} \\
&= \langle \Xi, \alpha\ \mathsf{type_s} \vdash \alpha\ \mathsf{type_s} \rangle^q \pi^{\Theta,\Xi,\alpha}_{\Xi,\alpha} : [\![\Xi, \alpha\ \mathsf{type_s} \vdash \alpha\ \mathsf{type_s}]\!] \pi^{\Theta,\Xi,\alpha}_{\Xi,\alpha} \Rightarrow \\
&\qquad \Rightarrow [\![\Xi, \alpha\ \mathsf{type_s} \vdash \alpha\ \mathsf{type_s}]\!]^q \pi^{\Theta,\Xi,\alpha}_{\Xi,\alpha}.
\end{aligned}$$

This is what we wanted to show.

Case (C**1**)**:** Recall eqs. (116) to (120). This case follows from the fact that the interpretations constant functors onto the same domain.

The other cases follow from the induction hypothesis and proposition 8.4.4. Explicitly, consider a type-forming rule

$$\frac{\Xi, \Xi_1 \vdash A_1 \ \mathsf{type_s} \quad \cdots \quad \Xi, \Xi_n \vdash A_n \ \mathsf{type_s}}{\Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type_s}}$$

Assume that its interpretation is given by

$$\begin{aligned}&\langle \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type_s} \rangle^q \\ &= [\![F]\!]_{[\![\Xi]\!]} \left(\langle \Xi, \Xi_1 \vdash A_1 \ \mathsf{type_s} \rangle^p, \ldots, \langle \Xi_n \vdash A_n \ \mathsf{type_s} \rangle^q \right).\end{aligned}$$

where $[\![F]\!]$ is a natural interpretation

$$[\![F]\!]_{[\![\Xi]\!]} : \left(\prod_{i=1}^{n} \mathbf{Cell}_{\mathbf{CFP}}([\![\Xi, \Xi_i]\!], \mathbf{Stab}_{\perp!}) \right) \to \mathbf{Cell}_{\mathbf{CFP}}([\![\Xi]\!], \mathbf{Stab}_{\perp!}).$$

Given any other context of type variables Θ disjoint from Ξ, we would like to show that

$$\langle \Theta, \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type_s} \rangle^q = \langle \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type_s} \rangle^q * \pi_{\Xi}^{\Theta, \Xi}.$$

By the induction hypothesis, we have for all $1 \le i \le n$,

$$\langle \Theta, \Xi, \Xi_i \vdash A_i \ \mathsf{type_s} \rangle^q = \langle \Xi, \Xi_i \vdash A_i \ \mathsf{type_s} \rangle^q * \pi_{\Xi, \Xi_i}^{\Theta, \Xi, \Xi_i}.$$

Using these facts we compute:

$$\begin{aligned}&\langle \Theta, \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type_s} \rangle^q \\ &= [\![F]\!]_{[\![\Theta, \Xi]\!]} \left(\langle \Theta, \Xi, \Xi_1 \vdash A_1 \ \mathsf{type_s} \rangle^q, \ldots, \langle \Theta, \Xi, \Xi_n \vdash A_n \ \mathsf{type_s} \rangle^q \right)\end{aligned}$$

which by the induction hypothesis,

$$\begin{aligned}&= [\![F]\!]_{[\![\Theta, \Xi]\!]} \left(\langle \Xi, \Xi_1 \vdash A_1 \ \mathsf{type_s} \rangle^q * \pi_{\Xi, \Xi_1}^{\Theta, \Xi, \Xi_1}, \ldots, \langle \Xi, \Xi_n \vdash A_n \ \mathsf{type_s} \rangle^q * \pi_{\Xi, \Xi_n}^{\Theta, \Xi, \Xi_n} \right) \\ &= [\![F]\!]_{[\![\Theta, \Xi]\!]} \left(\langle \Xi, \Xi_1 \vdash A_1 \ \mathsf{type_s} \rangle^q * (\pi_{\Xi}^{\Theta, \Xi} \times [\![\Xi_1]\!]), \ldots, \langle \Xi, \Xi_n \vdash A_n \ \mathsf{type_s} \rangle^q * (\pi_{\Xi}^{\Theta, \Xi} \times [\![\Xi_n]\!]) \right)\end{aligned}$$

which by naturality of $[\![F]\!]$,

$$\begin{aligned}&= [\![F]\!]_{[\![\Xi]\!]} \left(\langle \Xi, \Xi_1 \vdash A_1 \ \mathsf{type_s} \rangle^q, \ldots, \langle \Xi, \Xi_n \vdash A_n \ \mathsf{type_s} \rangle^q \right) * \pi_{\Xi}^{\Theta, \Xi} \\ &= \langle \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type_s} \rangle^q * \pi_{\Xi}^{\Theta, \Xi}.\end{aligned}$$

This is what we wanted to show. □

Next, we show that substitution is given by composition. Recall context morphisms from definition 2.5.7. We write $\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi$ for context morphisms of session types to differentiate them from context morphisms at the term level, below. Context morphisms $\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi$ denote 2-cells

$$\langle A_1, \ldots, A_n :_{\mathsf{s}} \Theta \rightsquigarrow \alpha_1 \ \mathsf{type}_{\mathsf{s}}^{q_1}, \ldots, \alpha_n \ \mathsf{type}_{\mathsf{s}}^{q_n} \rangle^q = \langle \alpha_i : \langle \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i} \rangle^q \rangle_{1 \le i \le n}$$

where $q \in \{-, +\}$. In particular,

$$\langle \cdot :_{\mathsf{s}} \Theta \rightsquigarrow \cdot \rangle^q = \mathsf{id} : \top \Rightarrow \top : [\![\Theta]\!] \to \top_{\mathbf{CFP}},$$

where $\top_{\mathbf{CFP}}$ is the nullary product in **CFP**, and $\top : [\![\Theta]\!] \to \top_{\mathbf{CFP}}$ is the unique functor from $[\![\Theta]\!]$ to it.

Lemma 8.5.2 (Weakening of Context Morphisms). *Let $\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi$ be arbitrary and Θ, Ω a context. Then $\sigma :_{\mathsf{s}} \Omega, \Theta \rightsquigarrow \Xi$ and where q ranges over $\{-, +\}$,*

$$\langle \sigma :_{\mathsf{s}} \Omega, \Theta \rightsquigarrow \Xi \rangle^q = \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle^q \pi_{\Theta}^{\Omega, \Theta}.$$

Proof. We consider two cases. The first case is when σ is empty, i.e., $\sigma :_{\mathsf{s}} \Omega, \Theta \rightsquigarrow \cdot$. That

$$\langle \sigma :_{\mathsf{s}} \Omega, \Theta \rightsquigarrow \cdot \rangle^p = \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \cdot \rangle^p \pi_{\Theta}^{\Omega, \Theta}$$

follows immediately from the fact that $\top_{\mathbf{CFP}}$ is terminal.

Assume now that σ is $A_1, \ldots, A_n :_{\mathsf{s}} \Theta \rightsquigarrow \alpha_1 \ \mathsf{type}_{\mathsf{s}}^{q_1}, \ldots, \alpha_n \ \mathsf{type}_{\mathsf{s}}^{q_n}$. By weakening, $\Omega, \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i}$ for all $1 \le i \le n$, and by proposition 8.5.1

$$\langle \Omega, \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i} \rangle^q = \langle \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i} \rangle^q \pi_{\Theta}^{\Omega, \Theta}.$$

We then compute:

$$\begin{aligned}
&\langle \sigma :_{\mathsf{s}} \Omega, \Theta \rightsquigarrow \Xi \rangle^q \\
&= \langle \alpha_i : \langle \Omega, \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i} \rangle^q \rangle_{1 \le i \le n} \\
&= \langle \alpha_i : \langle \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i} \rangle^q \pi_{\Theta}^{\Omega,\Theta} \rangle_{1 \le i \le n} \\
&= \langle \alpha_i : \langle \Theta \vdash A_i \ \mathsf{type}_{\mathsf{s}}^{q_i} \rangle^q \rangle_{1 \le i \le n} \pi_{\Theta}^{\Omega,\Theta} \\
&= \langle \sigma, A :_{\mathsf{s}} \Theta \rightsquigarrow \Xi, \alpha \ \mathsf{type}_{\mathsf{s}}^{q} \rangle^p \pi_{\Theta}^{\Omega,\Theta}.
\end{aligned}$$

□

Proposition 8.5.3 (Semantic Substitution of Session Types). *Let $\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi$ be arbitrary and let q range over $\{-,+\}$. If $\Xi \vdash A \ \mathsf{type}_{\mathsf{s}}^p$, then*

$$\begin{aligned}
[\![\Theta \vdash [\sigma]A \ \mathsf{type}_{\mathsf{s}}^p]\!] &= [\![\Xi \vdash A \ \mathsf{type}_{\mathsf{s}}^p]\!] \circ [\![\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi]\!], \\
[\![\Theta \vdash [\sigma]A \ \mathsf{type}_{\mathsf{s}}^p]\!]^q &= [\![\Xi \vdash A \ \mathsf{type}_{\mathsf{s}}^p]\!]^q \circ [\![\sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi]\!]^q, \\
\langle \Theta \vdash [\sigma]A \ \mathsf{type}_{\mathsf{s}}^p \rangle^q &= \langle \Xi \vdash A \ \mathsf{type}_{\mathsf{s}}^p \rangle^q * \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle^q.
\end{aligned}$$

Proof. By induction on the derivation of $\Xi \vdash A \ \mathsf{type}_{\mathsf{s}}$. Each case follows the same pattern. Consider a type-forming rule

$$\frac{\Xi, \Xi_1 \vdash A_1 \ \mathsf{type}_{\mathsf{s}} \quad \cdots \quad \Xi, \Xi_n \vdash A_n \ \mathsf{type}_{\mathsf{s}}}{\Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type}_{\mathsf{s}}}$$

By proposition 8.4.4, $\langle \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type}_{\mathsf{s}} \rangle^q$ is given by a natural interpretation

$$\langle F \rangle^q_{[\![\Xi]\!]} : \left(\prod_{i=1}^{n} \mathbf{Cell}_{\mathbf{CFP}}([\![\Xi, \Xi_i]\!], \mathbf{Stab}_{\perp!}) \right) \to \mathbf{Cell}_{\mathbf{CFP}}([\![\Xi]\!], \mathbf{Stab}_{\perp!}).$$

We need to show that

$$\begin{aligned}
&\langle \Theta \vdash [\sigma](F(A_1, \ldots, A_n)) \ \mathsf{type}_{\mathsf{s}} \rangle^q \\
&= \langle \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type}_{\mathsf{s}} \rangle^q * \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle^q.
\end{aligned}$$

From this fact, it will immediately follow that the source and target horizontal morphisms will respect substitution. By the definition of syntactic substitution, we know that

$$[\sigma](F(A_1, \ldots, A_n)) = F([\sigma]A_1, \ldots, [\sigma]A_n).$$

For each $1 \le i \le n$, let σ_i be given by $\sigma, \Xi_i :_{\mathsf{s}} \Theta, \Xi_i \rightsquigarrow \Xi, \Xi_i$. Observe that $[\sigma]A_i = [\sigma_i]A_i$ for all $1 \le i \le n$. By lemma 8.5.2, properties of products, and the interpretations of (CVar),

$$\langle \sigma, \Xi_i :_{\mathsf{s}} \Theta, \Xi_i \rightsquigarrow \Xi, \Xi_i \rangle = \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle \times \mathsf{id}_{[\![\Xi_i]\!]}. \tag{198}$$

By the induction hypothesis, we know for $1 \le i \le n$ that

$$\langle \Theta \vdash [\sigma_i]A_i \ \mathsf{type}_{\mathsf{s}} \rangle^q = \langle \Xi, \Xi_i \vdash A_i \ \mathsf{type}_{\mathsf{s}} \rangle^p * \langle \sigma, \Xi_i :_{\mathsf{s}} \Theta \rightsquigarrow \Xi, \Xi_i \rangle^q. \tag{199}$$

Using these facts, we get:

$$\begin{aligned}
&\langle \Theta \vdash [\sigma](F(A_1, \ldots, A_n)) \ \mathsf{type}_{\mathsf{s}} \rangle^q \\
&= \langle \Theta \vdash F([\sigma]A_1, \ldots, [\sigma]A_n) \ \mathsf{type}_{\mathsf{s}} \rangle^q \\
&= \langle \Theta \vdash F([\sigma_1]A_1, \ldots, [\sigma_n]A_n) \ \mathsf{type}_{\mathsf{s}} \rangle^q \\
&= \langle F \rangle^q_{[\![\Theta]\!]} \left(\left(\langle \Theta, \Xi_i \vdash [\sigma_i]A_i \ \mathsf{type}_{\mathsf{s}} \rangle^q \right)_{1 \le i \le n} \right)
\end{aligned}$$

which by eqs. (198) and (199),

$$= \langle F \rangle^q_{[\![\Theta]\!]} \left(\left(\langle \Xi, \Xi_i \vdash A_i \ \mathsf{type}_{\mathsf{s}} \rangle^q * \left(\langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle^q \times \mathsf{id}_{[\![\Xi_i]\!]} \right) \right)_{1 \le i \le n} \right)$$

which by naturality of $\langle F \rangle^q$,

$$\begin{aligned}
&= \langle F \rangle^p_{[\![\Xi]\!]} \left(\langle \Xi \vdash A_i \ \mathsf{type}_{\mathsf{s}} \rangle^q \right)_{1 \le i \le n} * \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle^q \\
&= \langle \Xi \vdash F(A_1, \ldots, A_n) \ \mathsf{type}_{\mathsf{s}} \rangle^q * \langle \sigma :_{\mathsf{s}} \Theta \rightsquigarrow \Xi \rangle^q.
\end{aligned}$$

This is what we wanted to show. □

8.5.2. Semantic Properties of Terms and Processes. Our semantics respects the exchange rule because we interpret structural contexts as indexed products. It also respects weakening and substitution.

Proposition 8.5.4 (Coherence of Terms and Processes). *If* $\Psi \Vdash M : \tau$, *then* $[\![\Phi, \Psi \Vdash M : \tau]\!] = [\![\Psi \Vdash M : \tau]\!] \circ \pi_{\Psi}^{\Phi,\Psi}$. *If* $\Psi \,;\, \Delta \vdash P :: a : A$, *then* $[\![\Phi, \Psi \,;\, \Delta \vdash P :: a : A]\!] = [\![\Psi \,;\, \Delta \vdash P :: a : A]\!] \circ \pi_{\Psi}^{\Phi,\Psi}$.

Proof. By induction on the derivation of $\Psi \Vdash M : \tau$ and $\Psi \,;\, \Delta \vdash P :: a : A$.

Case (F-Var)**:** Recall eq. (134). We use properties of products to compute:

$$[\![\Phi, \Psi, x : \tau \Vdash x : \tau]\!] = \pi_x^{\Phi,\Psi,x} = \pi_x^{\Psi,x} \circ \pi_{\Psi,x}^{\Phi,\Psi,x} = [\![\Psi, x : \tau \Vdash x : \tau]\!] \circ \pi_{\Psi,x}^{\Phi,\Psi,x}.$$

Case (**1**R)**:** Recall eq. (122). The result is obvious.

The remaining cases all follow an identical proof outline. This outline the direct analog of the one given in the proof proposition 8.5.1, and it is not reproduced here. □

We write $\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi$ for context morphisms in the functional layer. Context morphisms $\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi$ denote continuous morphisms $[\![\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi]\!] : [\![\Phi]\!] \to [\![\Psi]\!]$. In particular,

$$[\![M_1, \ldots, M_n :_{\mathsf{f}} \Psi \rightsquigarrow x_1 : \tau_1, \ldots, x_n : \tau_n]\!] = \langle x_i : [\![\Psi \Vdash M_i : \tau_i]\!] \rangle_{1 \le i \le n}.$$

Lemma 8.5.5 (Weakening of Context Morphisms). *Let* $\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi$ *be arbitrary and* Γ, Φ *a context. Then* $\sigma :_{\mathsf{f}} \Gamma, \Phi \rightsquigarrow \Psi$ *and* $[\![\sigma :_{\mathsf{f}} \Gamma, \Phi \rightsquigarrow \Psi]\!] = [\![\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi]\!] \circ \pi_{\Phi}^{\Gamma,\Phi}$.

Proof. Analogous to the proof of lemma 8.5.2. □

Proposition 8.5.6 (Semantic Substitution of Terms). *Let* $\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi$ *be arbitrary.*

(1) If $\Psi \Vdash N : \tau$, *then* $[\![\Phi \Vdash [\sigma]N : \tau]\!] = [\![\Psi \Vdash N : \tau]\!] \circ [\![\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi]\!]$.

(2) If $\Psi \,;\, \Delta \vdash P :: c : C$, *then* $[\![\Phi \,;\, \Delta \vdash [\sigma]P :: c : C]\!] = [\![\Psi \,;\, \Delta \vdash P :: c : C]\!] \circ [\![\sigma :_{\mathsf{f}} \Phi \rightsquigarrow \Psi]\!]$.

Proof. Analogous to the proof of proposition 8.5.3. □

Proposition 8.5.7 (Renaming of channels). *If* $\Psi \,;\, \Delta \vdash P :: a : A$ *and* $\sigma \colon \Delta, a \leftrightarrow \Gamma, b$, *then for all* $u \in [\![\Psi]\!]$,

$$[\![\Psi \,;\, \Gamma \vdash [\sigma]P :: b : A]\!]u = [\![\sigma]\!]^{-1} \circ [\![\Psi \,;\, \Delta \vdash P :: a : A]\!]u \circ [\![\sigma]\!]^{\pm},$$

where $[\![\sigma]\!] : [\![\Gamma, b : A]\!] \to [\![\Delta, a : A]\!]$ *and* $[\![\sigma]\!]^{\pm} : [\![\Gamma]\!]^{+} \times [\![b : A]\!]^{-} \to [\![\Delta]\!]^{+} \times [\![a : A]\!]^{-}$ *are the obvious relabelling isomorphisms of indexed products.*

Proof. By induction on $\Psi \,;\, \Delta \vdash P :: a : A$. □

The following proposition states that forwarding acts to rename channels:

Proposition 8.5.8. *For all processes* $\Psi \,;\, \Delta \vdash P :: c : C$ *with* C *positive or negative, respectively,*

$$[\![\Psi \,;\, \Delta \vdash c \leftarrow P;\ c \to d :: d : C]\!] = [\![\Psi \,;\, \Delta \vdash [d/c]P :: d : C]\!],$$
$$[\![\Psi \,;\, \Delta \vdash c \leftarrow P;\ c \leftarrow d :: d : C]\!] = [\![\Psi \,;\, \Delta \vdash [d/c]P :: d : C]\!].$$

For all processes $\Psi \,;\, \Delta, a : A \vdash P :: c : C$ *with* A *positive or negative, respectively,*

$$[\![\Psi \,;\, \Delta, b : B \vdash a \leftarrow (b \to a);\ P :: c : C]\!] = [\![\Psi \,;\, \Delta, b : B \vdash [b/a]P :: d : C]\!],$$
$$[\![\Psi \,;\, \Delta, b : B \vdash a \leftarrow (b \leftarrow a);\ P :: c : C]\!] = [\![\Psi \,;\, \Delta, b : B \vdash [b/a]P :: d : C]\!].$$

Proof. We show the first equality; the other three will follow analogously. Let $u \in [\![\Psi]\!]$ be arbitrary. We compute, using eqs. (113) and (115), proposition 8.5.7, and the fact that $\mathsf{id}_{[\![C]\!]}$ is $\langle \langle C \rangle^{p}, \langle C \rangle^{p} \rangle$:

$$\begin{aligned}
&[\![\Psi \,;\, \Delta \vdash c \leftarrow P;\ c \to d :: d : C]\!]u \\
&= [\![\Psi \,;\, c : C \vdash c \to d :: d : C]\!]u \circ [\![\Psi \,;\, \Delta \vdash P :: c : C]\!]u \\
&= \mathsf{id}_{[\![C]\!]} \circ [\![\Psi \,;\, \Delta \vdash [d/c]P :: d : C]\!]u \\
&= [\![\Psi \,;\, \Delta \vdash [d/c]P :: d : C]\!]u.
\end{aligned}$$

The environment u was arbitrary, so we conclude the result. □

8.6. Soundness

In this section, we show that our denotational semantics is sound. In the case of the functional layer, this means that our denotational semantics agrees with evaluation. In the case of the process layer, this means that denotational equivalence implies barbed congruence. As summarized by fig. 7.1, barbed congruence implies external equivalence and external congruence.

Soundness of the functional interpretation is analogous to soundness of the (stable) fixed-point semantics of PCF [Gun92, Theorems 4.23 and 5.23].

Proposition 8.6.1 (Soundness of Functional Interpretation). *Let M and v be closed terms of type τ, i.e., such that $\cdot \Vdash M : \tau$ and $\cdot \Vdash v : \tau$.*

(1) If v val, then $[\![\cdot \Vdash v : \tau]\!]\bot \neq \bot$.
(2) If $M \Downarrow v$, then $[\![\cdot \Vdash M : \tau]\!] = [\![\cdot \Vdash v : \tau]\!]$.

Proof. Assume first that τ is not a quoted process type and that v val. We show that $[\![\cdot \Vdash v : \tau]\!]\bot \neq \bot$ by case analysis on τ:

Case $\{a : A \leftarrow \overline{a_i : A_i}\}$**:** By the canonical forms lemma (proposition 5.8.2), $v = a \leftarrow \{P\} \leftarrow \overline{a_i}$ for some process P. It is immediate from eq. (142) that $[\![\cdot \Vdash a \leftarrow \{P\} \leftarrow \overline{a_i} : \{a : A \leftarrow \overline{a_i : A_i}\}]\!]\bot \neq \bot$.

Case **nat:** An inductive argument extends the canonical forms lemma to state that values of type **nat** are either o or $s(v)$ for some value of type **nat**. In the first case, it is clear that $[\![\cdot \Vdash \mathsf{o} : \mathbf{nat}]\!]\bot = \mathsf{o} \neq \bot$. In the second case, $[\![\cdot \Vdash s(v) : \mathbf{nat}]\!]\bot = [\![\cdot \Vdash v : \mathbf{nat}]\!]\bot + 1 \neq \bot$.

Case $\sigma \to \sigma'$**:** By the canonical forms lemma (proposition 5.8.2), $v = \lambda x : \sigma.M'$ for some term $x : \sigma \Vdash M : \sigma'$. It is immediate from eq. (135) that $[\![\cdot \Vdash \lambda x : \sigma.M' : \sigma \to \sigma']\!]u \neq \bot$.

Assume next that $M \Downarrow v$. The fact that $[\![\cdot \Vdash M : \tau]\!] = [\![\cdot \Vdash v : \tau]\!]$ follows mutatis mutandis from the soundness proof for the fixed-point semantics of PCF [Gun92, Theorem 4.23]. Two changes are required. First, we add an axiom case for quoted processes: it is immediate. Second, we drop several cases from the proof (the cases involving the predecessor, zero test, and conditional operators). □

Though the introduction of quoted values to the functional layer poses no problems for soundness, it breaks particularly generous forms of adequacy (cf. [Gun92, Theorem 4.24]):

Falsehood ((Generous) Adequacy of Functional Interpretation). *If τ is a base type and M is a term and v is a value such that $\cdot \Vdash M : \tau$, $\cdot \Vdash v : \tau$, and $[\![\cdot \Vdash M : \tau]\!] = [\![\cdot \Vdash v : \tau]\!]$, then $M \Downarrow v$.*

Proof. We provide a counter-example. Consider processes P and Q given by:

$$P = b \leftarrow (a \to b);\ (c \leftarrow (b \to c);\ (c \to d)),$$
$$Q = c \leftarrow (b \leftarrow (a \to b);\ (b \to c));\ c \to d.$$

They are denotationally equivalent processes, so quoting them gives denotationally equivalent values:

$$[\![\cdot \Vdash d \leftarrow \{P\} \leftarrow a : \{d : A \leftarrow a : A\}]\!] = [\![\cdot \Vdash d \leftarrow \{Q\} \leftarrow a : \{d : A \leftarrow a : A\}]\!].$$

However, it is not the case that $d \leftarrow \{P\} \leftarrow a \Downarrow d \leftarrow \{Q\} \leftarrow a$. □

We can nevertheless show an adequacy result analogous to the one for PCF [Gun92, Theorem 6.12]:

Proposition 8.6.2 (Adequacy of Functional Interpretation). *If $\cdot \Vdash M : \tau$ is a purely functional closed term and $[\![\cdot \Vdash M : \tau]\!]\bot \neq \bot$, then there exists a value v such that $M \Downarrow v$.*

Proof. The proof carries over unchanged from [Gun92, Theorem 6.12]. It uses a logical relation between closed terms of type τ and elements of $[\![\tau]\!]$. It is not reproduced here. □

We turn our attention to soundness of the process layer. Recall that our observational notions of equivalence are defined on configurations, but on configurations. However, we have so far only defined our denotational semantics for processes and functional terms. We remedy this by lifting

denotations from processes to configurations. The interpretation of (Conf-C) generalizes the interpretation of (Cut) in the obvious way:

$$[\![\Sigma \parallel \Delta \vdash \cdot \vdash \mathsf{proc}(c, P) :: (c : A)]\!] = [\![\cdot\,; \Delta \vdash P :: c : A]\!]\bot, \tag{200}$$

$$[\![\Sigma \parallel \Delta \vdash \cdot \vdash \mathsf{msg}(c, m) :: (c : A)]\!] = [\![\cdot\,; \Delta \vdash m :: c : A]\!]\bot, \tag{201}$$

$$\begin{aligned}&[\![\Sigma, \check{\Pi}, \Sigma' \parallel \Gamma\Lambda \vdash \mathrm{I}_1 \Pi \mathrm{I}_2 \vdash \mathcal{C}, \mathcal{D} :: \Phi\Xi]\!] \\ &= [\![\check{\Pi}, \Sigma' \parallel \Pi\Lambda \vdash \mathrm{I}_2 \vdash \mathcal{D} :: \Xi]\!] \circ [\![\Sigma, \check{\Pi} \parallel \Gamma \vdash \mathrm{I}_1 \vdash \mathcal{C} :: \Phi\Pi]\!].\end{aligned} \tag{202}$$

Definition 8.6.3. **Denotational equivalence** on configurations is given by $\Gamma \vdash \mathcal{C} \equiv \mathcal{D} :: \Delta$ if and only if $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] = [\![\Gamma \vdash \mathcal{D} :: \Delta]\!]$. Write $\Gamma \vdash \mathcal{C} \sqsubseteq \mathcal{D} :: \Delta$ if and only if $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \sqsubseteq [\![\Gamma \vdash \mathcal{D} :: \Delta]\!]$. ◂

Three new concepts play a pivotal role in our proof of soundness: stability, denotational barbs, and bounded recursion. *Stable* configurations[29] are the process analogs of functional values:

Definition 8.6.4. A configuration $\mathcal{C}$ is **stable** if no rules are applicable to it. It **stabilizes** if there exists a $\mathcal{C}'$ such that $\mathcal{C} \to^* \mathcal{C}'$ and $\mathcal{C}'$ is stable. ◂

Denotational barbs are a denotational characterization of definition 7.3.5:

Definition 8.6.5. If $\Gamma \vdash \mathcal{C} :: \Delta$, then write $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \Downarrow_a$ if $(\pi_a \circ [\![\Gamma \vdash \mathcal{C} :: \Delta]\!])(\bot) \neq \bot$. ◂

Bounded fixed point operators will let us express unbounded fixed point operators in terms of their finite unfoldings. This will give us an inductive handle on general recursion. We introduce the following auxiliary typing and evaluation rules and denotation.

$$\frac{\Psi, x : \tau \Vdash M : \tau}{\Psi \Vdash \mathsf{fix}^n\, x.M : \tau}\ (\text{F-Fix}^n) \qquad \frac{[\mathsf{fix}^n\, x.M/x]M \Downarrow v}{\mathsf{fix}^{n+1}\, x.M \Downarrow v}\ (\text{EV-Fix}^{n+1})$$

$$[\![\Psi \Vdash \mathsf{fix}^n\, x.M : \tau]\!]u = (\lambda x \in [\![\tau]\!].[\![\Psi, x : \tau \Vdash M : \tau]\!](u, x))^n\, \bot.$$

Intuitively, the bounded fixed point operator $\mathsf{fix}^n\, x.M$ behaves like the fixed point operator $\mathsf{fix}\, x.M$, except that it can only be unfolded up to n times. Its denotation is natural in its environment and so enjoys the same substitution properties as the rest of Polarized SILL.

At a high-level, our proof of soundness has the following structure:

(1) We show that configurations without unbounded recursion are stabilizing.
(2) We show that definitions 7.3.5 and 8.6.5 coincide on stabilizing configurations.
(3) We show that the denotations of arbitrary configurations are the directed suprema of the denotations of stabilizing configurations below it.
(4) We show that if a stabilizing configuration below $\mathcal{C}$ has a denotational barb, then so does $\mathcal{C}$.
(5) We deduce that denotational equivalence of configurations is a weak barbed congruence.

Our soundness proof relies on the following simplifying assumption:

Assumption 8.6.6. The rule (E-{}) does not appear in the right premise of (F-Fun), i.e., (E-{}) never appears in the argument of a function abstraction.

We start by showing that configurations without (F-Fix) are stabilizing. The proof is syntax-driven and uninspired, but sufficient thanks to assumption 8.6.6. Roughly, the approach is to establish a simulation between a configuration $\mathcal{C}$ and the configuration $\ulcorner \mathcal{C} \urcorner$ in which every bounded fixed-point operator has been completely unrolled. We show that each configuration with no fixed-point operators whatsoever is stabilizing; this establishes a bound on the number of steps $\mathcal{C}$ can take.

Proposition 8.6.7. *If $\Gamma \vdash \mathcal{C} :: \Delta$ contains no instances of (*F-Fix*) or (*F-Fix^n*), then it is stabilizing.*

[29] This use of the adjective "stable" is unrelated to its use in "stable morphisms".

Proof. By induction on the number of process operators in $\mathsf{proc}(c, P)$ facts in $\mathcal{C}$. It is obvious that each multiset rewriting rule decreases the number of process operators, except potentially the rules rule (73) (unquoting), rules (74) and (77) (sending values), and rules (75) and (76) (receiving values). Those for sending and receiving values also decrease the number of process operators by assumption 8.1.8. Rule (73) also decreases the number of process operators. Indeed, assume $M \Downarrow v$ and that M contains no instances of (F-Fix) or (F-Fix^n). If we also assume assumption 8.6.6, then v contains at most as many process operators as M. Then the left hand side of the rule,

$$\mathbf{eval}(M, a \leftarrow \{P\} \leftarrow \overline{a_i}), \mathsf{proc}(a, a \leftarrow \{M\} \leftarrow \overline{a_i})$$

has at least one process operator more than the right hand side of the rule, $\mathsf{proc}(a, P)$. This gives the result. □

Next, we give a translation on configurations that unrolls bounded fixed-point operators.[30] It is given by induction on the configuration, where all cases are structure-preserving except for:

$$\ulcorner \mathsf{fix}^{n+1}\ x.M \urcorner = [\ulcorner \mathsf{fix}^n\ x.M \urcorner / x] \ulcorner M \urcorner.$$

This unfolding operation respects substitution (cf. [Gun92, Lemma 4.28]):

Lemma 8.6.8. *For all M and N, $\ulcorner [M/x]N \urcorner = [\ulcorner M \urcorner / x] \ulcorner N \urcorner$.*

Proof. By well-founded induction on the set of well-formed terms, ordered by the transitive closure of the least relation $<$ generated by:

(1) if M is a subphrase of N, then $M < N$; and
(2) $\mathsf{fix}^n\ x.N < \mathsf{fix}^{n+1}\ x.N$ for all n.

We induct on N. The variable and zero cases are immediate, and the abstraction, application, successor, and nullary bounded fixed-point operator cases follow immediately by the induction hypothesis. The only mildly interesting case involves a bounded fixed-point operator with a non-zero bound, which follows by a computation and the induction hypothesis:

$$\begin{aligned}
&\ulcorner [M/x]\mathsf{fix}^{n+1}\ y.N \urcorner \\
&= \ulcorner \mathsf{fix}^{n+1}\ y.[M/x]N \urcorner \\
&= [\ulcorner \mathsf{fix}^n\ y.[M/X]N \urcorner / y] \ulcorner [M/x]N \urcorner
\end{aligned}$$

which by the induction hypothesis:

$$\begin{aligned}
&= [[\ulcorner M \urcorner / x] \ulcorner \mathsf{fix}^n\ y.N \urcorner / y]([\ulcorner M \urcorner / x] \ulcorner N \urcorner) \\
&= [\ulcorner M \urcorner / x]([\ulcorner \mathsf{fix}^n\ y.N \urcorner / y]N) \\
&= [\ulcorner M \urcorner / x] \ulcorner \mathsf{fix}^{n+1}\ y.N \urcorner.
\end{aligned}$$

□

We use lemma 8.6.8 to show that unfolded bounded fixed points simulate bounded fixed points:

Lemma 8.6.9. *If $\Gamma \Vdash M : \tau$ contains no instances of (*F-Fix*) and $M \Downarrow v$, then $\ulcorner M \urcorner \Downarrow \ulcorner v \urcorner$.*

Proof. By induction on $M \Downarrow v$. The value cases are immediate, while the case (EV-Succ) follows easily from the induction hypothesis.

Case (EV-App)**:** Assume $MN \Downarrow v$ because $M \Downarrow \lambda x : \tau.M'$, $N \Downarrow w$, and $[w/x]M' \Downarrow v$. By the induction hypothesis, $\ulcorner M \urcorner \Downarrow \lambda x : \tau.\ulcorner M' \urcorner$, $\ulcorner N \urcorner \Downarrow \ulcorner w \urcorner$, and $\ulcorner [w/x]M' \urcorner \Downarrow \ulcorner v \urcorner$. By lemma 8.6.8, $[\ulcorner w \urcorner / x] \ulcorner M' \urcorner \Downarrow \ulcorner v \urcorner$. We conclude that $\ulcorner MN \urcorner \Downarrow \ulcorner v \urcorner$ as desired.

Case (EV-Fix^{n+1})**:** Assume $\mathsf{fix}^{n+1}\ x.M \Downarrow v$ because $[\mathsf{fix}^n\ x.M/x]M \Downarrow v$. We must show that $\ulcorner \mathsf{fix}^{n+1}\ x.M \urcorner \Downarrow \ulcorner v \urcorner$. Observe that $\ulcorner \mathsf{fix}^{n+1}\ x.M \urcorner = [\ulcorner \mathsf{fix}^n\ x.M \urcorner / x] \ulcorner M \urcorner$. By the induction hypothesis, $\ulcorner [\mathsf{fix}^n\ x.M/x]M \urcorner \Downarrow \ulcorner v \urcorner$. By lemma 8.6.8, $\ulcorner [\mathsf{fix}^n\ x.M/x]M \urcorner = [\ulcorner \mathsf{fix}^n\ x.M \urcorner / x] \ulcorner M \urcorner$. It then follows that $\ulcorner \mathsf{fix}^{n+1}\ x.M \urcorner \Downarrow \ulcorner v \urcorner$, as desired. □

[30] It is similar to the unrolling operator given by Gunter [Gun92, p. 139], except that we do define $\ulcorner \mathsf{fix}^0\ x.M \urcorner = \mathsf{fix}\ x.x$. This is to simplify the statement of results below, where we need to talk about configurations that do not use (F-Fix).

Proposition 8.6.10. *If* $\Gamma \vdash \mathcal{C} :: \Delta$ *contains no instances of (*F-Fix*) and* $\mathcal{C} \to \mathcal{C}'$*, then* $\ulcorner \mathcal{C} \urcorner \to \ulcorner \mathcal{C}' \urcorner$.

Proof. By case analysis on the rule used for the step $\mathcal{C} \to \mathcal{C}'$. All cases are obvious except those involving the functional layer. The cases involving sending values or unquoting processes follow by lemma 8.6.9. The cases involving receiving values follow by lemma 8.6.8. □

Corollary 8.6.11. *If* $\Gamma \vdash \mathcal{C} :: \Delta$ *contains no instances of (*F-Fix*), then it is stabilizing.*

Proof. Each step $\mathcal{C}$ makes is matched by a step $\ulcorner \mathcal{C} \urcorner$ can make. But $\ulcorner \mathcal{C} \urcorner$ is stabilizing, so it can only make finitely many steps. It follows that $\mathcal{C}$ can only take finitely many steps, i.e., that it is stabilizing. □

Next, we show that our two notions of barbs, definitions 7.3.5 and 8.6.5, coincide on stabilizing configurations. To do so, we will need the fact that denotational equivalence is closed under multiset rewriting. Proposition 8.6.12 is the denotational analogue of proposition 7.1.3. We remark that it can be used to generate a long list of semantic equivalences.

Proposition 8.6.12. *If* $\Gamma \vdash \mathcal{C} :: \Delta$ *and* $\mathcal{C} \to \mathcal{C}'$*, then* $\Gamma \vdash \mathcal{C} \equiv \mathcal{C}' :: \Delta$.

Proof. By proposition 5.9.1 and compositionality, it is sufficient to show that if $\mathcal{E} \to \mathcal{E}'$ is an instance of a rule in $\mathcal{P}$ and $\Lambda \vdash \mathcal{E} :: \Xi$, then $\Lambda \vdash \mathcal{E} \equiv \mathcal{E}' :: \Xi$. A case analysis on this rule using propositions 8.5.8, 9.1.1 and 9.1.2 gives the result. The only subtle cases involves value transmission. We treat these cases explicitly:

Case (74)**:** Then $\Gamma \vdash \mathcal{C} :: \Delta$ is $\Gamma \vdash \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) :: a : \tau \wedge A$. By proposition 8.6.1, $\mathbf{eval}(M, v)$ implies that $[\![\Vdash M : \tau]\!]\bot = [\![\Vdash v : \tau]\!]\bot \neq \bot$. By eq. (200):

$$[\![\Gamma \vdash \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) :: a : \tau \wedge A]\!]$$
$$= [\![\cdot\,;\, \Gamma \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]\bot,$$

which by compositionality and proposition 8.5.8:

$$= [\![\cdot\,;\, \Gamma \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ d \leftarrow [d/a]P;\ d \to a :: a : \tau \wedge A]\!]\bot,$$

which by proposition 9.1.2:

$$= [\![\cdot\,;\, \Gamma \vdash d \leftarrow [d/a]P;\ \_ \leftarrow \mathsf{output}\ a\ M;\ d \to a :: a : \tau \wedge A]\!]\bot,$$

which by eq. (115):

$$= [\![\cdot\,;\, d : A \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ d \to a :: a : \tau \wedge A]\!]\bot \circ [\![\cdot\,;\, \Gamma \vdash [d/a]P :: d : A]\!]\bot$$

which by compositionality:

$$= [\![\cdot\,;\, d : A \vdash \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a :: a : \tau \wedge A]\!]\bot \circ [\![\cdot\,;\, \Gamma \vdash [d/a]P :: d : A]\!]\bot$$

which by eqs. (200) and (201):

$$= [\![b : A \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a) :: a : \tau \wedge A]\!] \circ [\![\Gamma \vdash \mathsf{proc}(d, [d/a]P) :: bA]\!]$$

which by eq. (202):

$$= [\![\Gamma \vdash \mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ d \to a) :: a : \tau \wedge A]\!].$$

This is what we wanted to show.

Case (75)**:** Then $\Gamma \vdash \mathcal{C} :: \Delta$ is $\Gamma, d : A \mid a : \tau \wedge A \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a), \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) :: c : C$. By assumption, $\mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a)$ is well-formed, so v val. By proposition 8.6.1, this implies that $[\![\cdot \Vdash v : \tau]\!] \neq \bot$. By eq. (202):

$$[\![\Gamma, d : A \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a), \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) :: c : C]\!]$$
$$= [\![\Gamma, a : \tau \wedge A \vdash \mathsf{proc}(c, x \leftarrow \mathsf{input}\ a;\ P) :: c : C]\!] \circ [\![d : A \vdash \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a) :: a : \tau \wedge A]\!],$$

which by eqs. (200) and (201):

$$= [\![\cdot\,;\, \Gamma, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ P :: c : C]\!]\bot \circ [\![\cdot\,;\, d : A \vdash \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a :: a : \tau \wedge A]\!]\bot,$$

which by eq. (115):

$$= [\![\cdot\,;\, \Gamma, d : A \vdash a \leftarrow \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a;\ x \leftarrow \mathsf{input}\ a;\ P :: c : C]\!]\bot,$$

which by proposition 9.1.1:

$= [\![\cdot\,;\Gamma, d : A \vdash a \leftarrow a \rightarrow d;\ [v/x]P :: c : C]\!]\bot,$

which by proposition 8.5.8:

$= [\![\cdot\,;\Gamma, d : A \vdash [d, v/a, x]P :: c : C]\!]\bot,$

which by eq. (200):

$= [\![\Gamma, d : A \vdash \mathsf{proc}(c, [d, v/a, x]P) :: cC]\!].$

This is what we wanted to show. □

Lemma 8.6.13. *If $f : A \times X \to B \times X$ is continuous and $\mathrm{Tr}^X(f)(\bot) \neq \bot$, then $f(\bot) \neq \bot$.*

Proof. By corollary 2.3.8,

$$\mathrm{Tr}^X(f)(\bot) = \pi_B^{B\times X}\left(\bigsqcup_{n\in\mathbb{N}}^{\uparrow}\left(\lambda\,(b,x)\,.f\,(\bot,x)\right)^n(\bot_B,\bot_X)\right).$$

Suppose to the contrary that $f(\bot) = \bot$, then an induction shows that

$$\left(\lambda\,(b,x)\,.f\,(\bot,x)\right)^n(\bot_B,\bot_X) = \bot$$

for all n. The result is now obvious from the definitions of least upper bound and of projection. □

Proposition 8.6.14. *If $\Gamma \vdash \mathcal{C} :: \Delta$ is stabilizing, then for all $a : A \in \Gamma, \Delta$, $\mathcal{C} \Downarrow_a$ if and only if $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \Downarrow_a$.*

Proof. Let $\mathcal{D}$ be such that $\mathcal{C} \to^* \mathcal{D}$ and $\mathcal{D}$ is stable. By proposition 5.9.1, $\Gamma \vdash \mathcal{D} :: \Delta$. By induction on $\mathcal{C} \to^* \mathcal{D}$ using proposition 8.6.12, $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \Downarrow_a$ if and only if $[\![\Gamma \vdash \mathcal{D} :: \Delta]\!] \Downarrow_a$. Analogously, by lemma 7.3.11, $\mathcal{C} \Downarrow_a$ if and only if $\mathcal{D} \Downarrow_a$. It is therefore sufficient to show the result for stable configurations. Assume without loss of generality that $\mathcal{C}$ is stable.

We begin with sufficiency. The configuration $\mathcal{C}$ is stable, so no rules can be applied to $\mathcal{C}$. It follows that if $\mathcal{C} \Downarrow_a$, then $\mathcal{C} \downarrow_a$. We proceed by induction on the derivation of $\Gamma \vdash \mathcal{C} :: \Delta$.

Case (Conf-M)**:** Immediate by a case analysis on $\mathcal{C} \downarrow_a$.

Case (Conf-P)**:** A case analysis on $\mathcal{C} \downarrow_a$ shows that this case is vacuously true.

Case (Conf-C)**:** Then $\mathcal{C} = \mathcal{D}, \mathcal{E}$ is the composition of some configurations $\Gamma_1 \vdash \mathcal{D} :: \Delta_1, \Pi$ and $\Pi, \Gamma_2 \vdash \mathcal{E} :: \Delta_2$. A case analysis on $\mathcal{C} \downarrow_a$ reveals that the barb is due to a message fact with carrier a, i.e., it does not involve forwarding processes. The responsible message fact must be contained in one of the two premisses. Assume that it is contained in $\mathcal{D}$; the case where it is in $\mathcal{E}$ will follow by symmetry. By assumption, $a \in \check{\Gamma}, \check{\Delta}$, so $a \in \check{\Gamma}_1, \check{\Delta}_1$. By the induction hypothesis, $[\![\Gamma_1 \vdash \mathcal{D} :: \Delta_1, \Pi]\!] \Downarrow_a$. Monotonicity and the Kleene fixed-point formulation of the trace operator corollary 2.3.8 imply that $\Gamma \vdash \mathcal{C} :: \Delta \Downarrow_a$.

Next, we show necessity. We proceed by induction on the derivation of $\Gamma \vdash \mathcal{C} :: \Delta$.

Case (Conf-M)**:** A case analysis on the message fact gives the result.

Case (Conf-P)**:** A case analysis on the process fact shows that the result is vacuously true. Indeed, if $[\![\Sigma \parallel \Delta \shortmid \cdot \vdash \mathsf{proc}(c, P) :: (c : A)]\!] \Downarrow_a$, then $[\![\cdot\,;\Delta \vdash P :: c : A]\!]\bot\bot \neq \bot$. Because the configuration is stable, we know that the process must be waiting to receive a message or be an instance of Ω. But in each of these cases, $[\![\cdot\,;\Delta \vdash P :: c : A]\!]\bot\bot = \bot$. So it cannot be the case that $[\![\Sigma \parallel \Delta \shortmid \cdot \vdash \mathsf{proc}(c, P) :: (c : A)]\!] \Downarrow_a$.

Case (Conf-C)**:** Then $\mathcal{C} = \mathcal{D}, \mathcal{E}$ is the composition of some configurations $\Gamma_1 \vdash \mathcal{D} :: \Delta_1, \Pi$ and $\Pi, \Gamma_2 \vdash \mathcal{E} :: \Delta_2$. By lemma 8.6.13, $[\![\Gamma_1 \vdash \mathcal{D} :: \Delta_1, \Pi]\!] \Downarrow_a$ or $[\![\Pi, \Gamma_2 \vdash \mathcal{E} :: \Delta_2]\!] \Downarrow_a$. The result follows readily from the induction hypothesis. □

We claim that if a configuration $\mathcal{C}$ has a denotational barb, then there exists a stabilizing configuration $\mathcal{C}'$ that is denotationally below $\mathcal{C}$ and that has the same barb. Intuitively, if $\mathcal{C}$ has a denotational barb on a, then it is because $\mathcal{C}$ sent a message on a after a finite number of steps. In particular, it must be because $\mathcal{C}$ sent a message on a after some finite number n of unrollings of its

fixed points. The stabilizing configuration $\mathcal{C}'$ is then given by replacing all unbounded fixed point operators in $\mathcal{C}$ with bounded fixed point operators allowing n unrollings.

Let the **n-fold truncation** $[\cdot]_n$ of terms, processes, and terms be an assignment of n units of potential to each instance of $\mathsf{fix}\, x.M$. The truncation is inductively defined on the syntax. All cases are structure-preserving except for:

$$[\mathsf{fix}\, x.M]_n = \mathsf{fix}^n\, x.[M]_n.$$

Conversely, let the bound erasure $\mathsf{e}(\cdot)$ replace all occurrences of $\mathsf{fix}^n\, x.M$ in a term, process, or configuration by $\mathsf{fix}\, x.M$. It is defined by induction on the syntax of terms, processes, or configurations in the obvious way.

These operations preserve typing:

Proposition 8.6.15. *If* $\Psi \Vdash M : \tau$, $\Psi \,;\, \Delta \vdash P :: c : C$, *or* $\Gamma \vdash \mathcal{C} :: \Delta$, *then for all* n, $\Psi \Vdash [M]_n : \tau$, $\Psi \,;\, \Delta \vdash [P]_n :: c : C$, *or* $\Gamma \vdash [\mathcal{C}]_n :: \Delta$, *respectively. Conversely, if* $\Psi \Vdash M : \tau$, $\Psi \,;\, \Delta \vdash P :: c : C$, *or* $\Gamma \vdash \mathcal{C} :: \Delta$ *have instances of (*F-Fix^n*) in their derivations, then* $\Psi \Vdash \mathsf{e}(M) : \tau$, $\Psi \,;\, \Delta \vdash \mathsf{e}(P) :: c : C$, *or* $\Gamma \vdash \mathsf{e}(\mathcal{C}) :: \Delta$, *respectively.*

Proof. By induction on the derivation. □

The denotation of an arbitrary configuration $\mathcal{C}$ is the directed supremum of its n-fold truncations. Each of these truncations is stabilizing by corollary 8.6.11.

Proposition 8.6.16. *For all terms* $\Psi \Vdash M : \tau$, *processes* $\Psi \,;\, \Delta \vdash P :: c : C$, *and configurations* $\Gamma \vdash \mathcal{C} :: \Delta$,

$$\begin{aligned}
[\![\Psi \Vdash M : \tau]\!] &= \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Psi \Vdash [M]_n : \tau]\!], \\
[\![\Psi \,;\, \Delta \vdash P :: c : C]\!] &= \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Psi \,;\, \Delta \vdash [P]_n :: c : C]\!], \\
[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] &= \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Gamma \vdash [\mathcal{C}]_n :: \Delta]\!].
\end{aligned}$$

Proof. By induction on the derivation of the term, process, or configuration. For terms and processes, all cases except (F-Fix) follow by continuity and proposition 8.4.17. We give one of these cases to illustrate.

Case (F-Fun)**:** Assume $\Psi \Vdash \lambda x : \tau.M : \tau \to \sigma$ because $\Psi, x : \tau \Vdash M : \sigma$. By the induction hypothesis,

$$[\![\Psi, x : \tau \Vdash M : \sigma]\!] = \bigsqcup_{n\in N}^{\uparrow} [\![\Psi, x : \tau \Vdash [M]_n : \sigma]\!].$$

Let η be the natural interpretation of (F-Fun) given by proposition 8.4.17,

$$\begin{aligned}
&[\![\Psi \Vdash \lambda x : \tau.M : \tau \to \sigma]\!] \\
&= \eta_{[\![\Xi]\!]}\left([\![\Psi, x : \tau \Vdash M : \sigma]\!]\right) \\
&= \eta_{[\![\Xi]\!]}\left(\bigsqcup_{n\in N}^{\uparrow} [\![\Psi, x : \tau \Vdash :]\!][M]_n \sigma\right) \\
&= \bigsqcup_{n\in N}^{\uparrow} \eta_{[\![\Xi]\!]}\left([\![\Psi, x : \tau \Vdash [:]_M n]\!]\sigma\right) \\
&= \bigsqcup_{n\in N}^{\uparrow} [\![\Psi \Vdash \lambda x : \tau.[M]_n : \tau \to \sigma]\!] \\
&= \bigsqcup_{n\in N}^{\uparrow} [\![\Psi \Vdash [\lambda x : \tau.M]_n : \tau \to \sigma]\!].
\end{aligned}$$

Case (F-Fix)**:** Assume that $\Psi \Vdash \mathsf{fix}\, x.M : \tau$ because $\Psi, x : \tau \Vdash M : \tau$. By the induction hypothesis,

$$[\![\Psi, x : \tau \Vdash M : \tau]\!] = \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Psi, x : \tau \Vdash [M]_n : \tau]\!]. \tag{203}$$

By eq. (137) and corollary 2.3.3,

$$\begin{aligned}
&[\![\Psi \Vdash \mathsf{fix}\, x.M : \tau]\!]u \\
&= [\![\Psi, x : \tau \Vdash M : \tau]\!]^{\dagger} u \\
&= \bigsqcup_{m\in\mathbb{N}}^{\uparrow} (\lambda x \in [\![\tau]\!].[\![\Psi, x : \tau \Vdash M : \tau]\!](u, x))^m\, (\bot) \\
&= \bigsqcup_{m\in\mathbb{N}}^{\uparrow} \left(\lambda x \in [\![\tau]\!]. \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Psi, x : \tau \Vdash [M]_n : \tau]\!](u, x)\right)^m\, (\bot) \\
&= \bigsqcup_{m\in\mathbb{N}}^{\uparrow} \bigsqcup_{n\in\mathbb{N}}^{\uparrow} (\lambda x \in [\![\tau]\!].[\![\Psi, x : \tau \Vdash [M]_n : \tau]\!](u, x))^m\, (\bot),
\end{aligned}$$

which by proposition 2.2.11:

$$\begin{aligned}
&= \bigsqcup_{n\in\mathbb{N}}^{\uparrow} (\lambda x \in [\![\tau]\!].[\![\Psi, x : \tau \Vdash [M]_n : \tau]\!](u, x))^n\, (\bot) \\
&= \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Psi \Vdash \mathsf{fix}^n\, [M]_n. : \tau]\!]u \\
&= \bigsqcup_{n\in\mathbb{N}}^{\uparrow} [\![\Psi \Vdash [\mathsf{fix}\, x.M]_n : \tau]\!]u.
\end{aligned}$$

For configurations, the cases (Conf-M) and (Conf-P) are immediate by the induction hypothesis, while (Conf-C) is analogous to (Cut). □

We show that $\mathcal{C}$ has a barb on a whenever one of its n-fold truncations does. Observe first that erasing bounds on bounded fixed-point operators does not affect evaluation (cf. [Gun92, Lemma 4.32]):

Proposition 8.6.17. *If* $M \Downarrow v$, *then* $\mathsf{e}(M) \Downarrow \mathsf{e}(v)$.

Proof. By induction on the derivation of $M \Downarrow v$. The base cases (EV-Fun), (EV-Proc), and (EV-Zero) are obvious. The remaining cases are:

Case (EV-Succ)**:** Assume that $\mathsf{s}(M) \Downarrow \mathsf{s}(n)$ because $M \Downarrow n$. By the induction hypothesis, $\mathsf{e}(M) \Downarrow \mathsf{e}(n)$. By (EV-Succ), $\mathsf{s}(\mathsf{e}(M)) \Downarrow \mathsf{s}(\mathsf{e}(n))$. But this is exactly $\mathsf{e}(\mathsf{s}(M)) \Downarrow \mathsf{e}(\mathsf{s}(n))$.

Case (EV-Fix)**:** Assume that $\mathsf{fix}\, x.M \Downarrow v$ because $[\mathsf{fix}\, x.M/x]M \Downarrow v$. By the induction hypothesis, $\mathsf{e}([\mathsf{fix}\, x.M/x]M) \Downarrow \mathsf{e}(v)$. Observe that $\mathsf{e}([\mathsf{fix}\, x.M/x]M) = [\mathsf{fix}\, x.\mathsf{e}(M)/x]\mathsf{e}(M)$. So by (EV-Fix) again, $\mathsf{e}(\mathsf{fix}\, x.M) \Downarrow \mathsf{e}(v)$.

Case (EV-fix^{n+1})**:** Assume that $\mathsf{fix}^{n+1}\, x.M \Downarrow v$ because $[\mathsf{fix}^n\, x.M/x]M \Downarrow v$. By the induction hypothesis, $\mathsf{e}([\mathsf{fix}^n\, x.M/x]M) \Downarrow \mathsf{e}(v)$. Observe that $\mathsf{e}([\mathsf{fix}^n\, x.M/x]M) = [\mathsf{fix}\, x.\mathsf{e}(M)/x]\mathsf{e}(M)$. Also observe that $\mathsf{e}(\mathsf{fix}^{n+1}\, x.M) = \mathsf{fix}\, x.\mathsf{e}(M)$. So by (EV-Fix), $\mathsf{e}(\mathsf{fix}^{n+1}\, x.M) \Downarrow \mathsf{e}(v)$.

Case (EV-App)**:** Assume that $MN \Downarrow v$ because $M \Downarrow \lambda x : \tau.M'$, $N \Downarrow w$, and $[w/x]M' \Downarrow v$. By the induction hypothesis, $\mathsf{e}(M) \Downarrow \lambda x : \tau.\mathsf{e}(M')$, $\mathsf{e}(N) \Downarrow \mathsf{e}(w)$, and $\mathsf{e}([w/x]M') \Downarrow \mathsf{e}(v)$. Observe that $\mathsf{e}([w/x]M') = [\mathsf{e}(w)/x]\mathsf{e}(M')$, and that $\mathsf{e}(MN) = (\mathsf{e}(M))(\mathsf{e}(N))$. We conclude that $\mathsf{e}(MN) \Downarrow \mathsf{e}(v)$ by (EV-App). □

Erasing bounds also does not affect barbs or multiset rewriting:

Proposition 8.6.18. *If* $\mathcal{C} \downarrow_a$, *then* $\mathsf{e}(\mathcal{C}) \downarrow_a$.

Proof. By case analysis on $\mathcal{C} \downarrow_a$. The only interesting case is $\_ \leftarrow \mathsf{output}\ a\ M;\ P \downarrow_a$ when $M \Downarrow v$ for some v. It follows by proposition 8.6.17. □

Proposition 8.6.19. *If* $\mathcal{C} \to \mathcal{C}'$, *then* $\mathsf{e}(\mathcal{C}) \to^* \mathsf{e}(\mathcal{C}')$.

Proof. By case analysis on the rule used to make the step. The only interesting cases are those involving the functional layer:

Case (73)**:** The rule is

$$\forall a, \overline{a}_i.\mathbf{eval}(M, a \leftarrow \{P\} \leftarrow \overline{a_i}), \mathsf{proc}(a, a \leftarrow \{M\} \leftarrow \overline{a_i}) \to \mathsf{proc}(a, P)$$

The result follows immediately from proposition 8.6.17.

Case (74): The rule is

$$\forall a, \Delta.\mathbf{eval}(M, v), \mathsf{proc}(a, \_ \leftarrow \mathsf{output}\ a\ M;\ P) \to \\ \to \exists d.\mathsf{proc}(d, [d/a]P), \mathsf{msg}(a, \_ \leftarrow \mathsf{output}\ a\ v;\ d \to a)$$

The result follows immediately from proposition 8.6.17.

Case (76): The rule is

$$\forall \Delta, a, d, c.\mathsf{proc}(a, x \leftarrow \mathsf{input}\ a;\ P), \mathsf{msg}(d, \_ \leftarrow \mathsf{output}\ a\ v;\ a \leftarrow d) \to \\ \to \mathsf{proc}(d, [d, v/a, x]P)$$

The result follows immediately from proposition 8.6.17 and a substitution property. □

Corollary 8.6.20. *If* $\mathcal{C} \Downarrow_a$, *then* $\mathsf{e}(\mathcal{C}) \Downarrow_a$.

Proof. The first sentence follows induction on the number of steps needed to produce the barb. The base case is given by proposition 8.6.18, while the inductive step is given by proposition 8.6.19. □

Proposition 8.6.21. *If* $\Gamma \vdash \mathcal{C} :: \Delta$, *then for all* $a : A \in \Gamma, \Delta$, *if* $\mathcal{C} \downarrow_a$, *then* $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \Downarrow_a$.

Proof. By induction on the derivation $\Gamma \vdash \mathcal{C} :: \Delta$.

Case (Conf-M): By a case analysis on $\mathcal{C} \downarrow_a$.

Case (Conf-P): By a case analysis on $\mathcal{C} \downarrow_a$.

Case (Conf-C): Assume first that $\mathcal{C} \downarrow_a$ because $(\mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m^-_{b,c})) \downarrow_a$. By proposition 8.5.8,

$$\Gamma', a : A \vdash \mathsf{proc}(b, a \leftarrow b), \mathsf{msg}(c, m^-_{b,c}) \equiv [a/b]m^-_{b,c} :: c : C.$$

The result then follows by a case analysis on $m^-_{b,c}$.

If $\mathcal{C} \downarrow_a$ because $(\mathsf{msg}(a, m^+), \mathsf{proc}(b, a \to b)) \downarrow_b$, then we can apply an analogous argument.

Otherwise, $\mathcal{C} \downarrow_a$ because $\mathcal{C}$ is of the form $\mathcal{E}[\mathcal{D}]$ and $\mathcal{D} \downarrow_a$. Without loss of generality, $\mathcal{D}$ is contained in one of the two premisses to (Conf-C). The result then follows by the induction hypothesis on that premise, monotonicity, and eq. (202). □

Going forward, we assume that $\Gamma \vdash \mathcal{C} :: \Delta$ is a configuration Polarized SILL processes, i.e., that (F-Fix^n) does not appear in the derivation of $\mathcal{C}$.

Proposition 8.6.22 (Soundness). *If* $\Gamma \vdash \mathcal{C} :: \Delta$, *then for all* $a : A \in \Gamma, \Delta$, $\mathcal{C} \Downarrow_a$ *if and only if* $[\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \Downarrow_a$.

Proof. Sufficiency is immediate by propositions 8.6.12 and 8.6.21. To see necessity, we observe that by proposition 8.6.16, $[\![\Gamma \vdash [\mathcal{C}]_n :: \Delta]\!] \Downarrow_a$ for some n. The configuration $[\mathcal{C}]_n$ is stabilizing by corollary 8.6.11, so $[\mathcal{C}]_n \Downarrow_a$ by proposition 8.6.14. By corollary 8.6.20, $\mathsf{e}([\mathcal{C}]_n) \Downarrow_a$. But $\mathsf{e}([\mathcal{C}]_n) = \mathcal{C}$ because $\mathcal{C}$ was assumed not to contain any bounded fixed point operators. So we conclude that $\mathcal{C} \Downarrow_a$. □

Theorem 8.6.23. *If* $\Gamma \vdash \mathcal{C} \equiv \mathcal{D} :: \Delta$, *then* $\Gamma \vdash \mathcal{C} \approx^c \mathcal{D} :: \Delta$.

Proof. We start by showing that $\equiv$ is a weak barbed bisimulation. The relation $\equiv$ is closed under multiset stepping by proposition 8.6.12. Proposition 8.6.22 implies that for all $a \in \breve{\Gamma}, \breve{\Delta}$,

$$\mathcal{C} \Downarrow_a \iff [\![\Gamma \vdash \mathcal{C} :: \Delta]\!] \Downarrow_a \iff [\![\Gamma \vdash \mathcal{D} :: \Delta]\!] \Downarrow_a \iff \mathcal{D} \Downarrow_a.$$

To see that denotational equivalence is contained in weak barbed congruence, it is then sufficient to observe that denotational equivalence is a congruence. □

8.7. Related Work

Atkey [Atk17] gave a denotational semantics for CP, where types are interpreted as sets and processes are interpreted as relations over these. Because processes in CP are proof terms for classical linear logic, the interpretation of processes is identical to the relational semantics of proofs in classical linear logic [Bar91]. Our jump from sets and relations to domains and continuous functions was motivated by two factors. First, domains provide a natural setting for studying recursion. Second, we believe that monotonicity and continuity are essential properties for a semantics of processes with infinite data, and it is unclear how to capture these properties in a relational setting. Our transition to domains and functions required polarized interpretations of types. In the case of recursive types, defining the relating natural families of embeddings and showing that they satisfied the structural rules required significant generalizations of the techniques found in [SP82]. Atkey interpreted process composition as relational composition. Our use of traces is more complex, but we believe that known trace identities make it tractable. We believe that the extra complexity is justified by SILL's more complex behavioural phenomena.

Our semantics generalizes Kahn's stream-based semantics for deterministic networks [Kah74]. A deterministic network is graph whose nodes are deterministic processes, and whose edges are unidirectional channels. Each channel carries values of a single fixed simple type, e.g., integers or booleans. Semantically, channels denote domains of sequences of values, and processes denote continuous functions from input channels to output channels. Our semantics generalizes this to allow for bidirectional, session-typed communication channels. Satisfactorily generalizing Kahn-style semantics to handle non-determinism is difficult [Bro88; KP85; Pan85; PS92; Sta87; Sta90], partly due to the Keller [Kel77] and Brock and Ackerman [BA81] anomalies.

Castellan and Yoshida [CY19] gave a game semantics interpretation of the session π-calculus with recursion. It is fully abstract relative to a barbed congruence notion of behavioural equivalence. Session types denote event structures that encode games and that are endowed with an ω-cpo structure. Open types denote continuous maps between these and recursive types are interpreted as least fixed points. Open processes are interpreted as continuous maps that describe strategies. We conjecture that our semantics could be related via barbed congruence.

Kokke, Montesi, and Peressotti [KMP19] gave a denotational semantics using Brzozowski derivatives [Brz64] to a proofs-as-processes interpretation between classical linear logic and the π-calculus. It does not handle recursion or the transmission of functional values.

8.8. Summary of Interpretations

For ease of reference, we give all of the semantic clauses (including omitted clauses).

8.8.1. Clauses for Term Formation (section 5.A.1).

Rule (I-{}):

$$[\![\Psi \Vdash a \leftarrow \{P\} \leftarrow \overline{a_i} : \{a : A \leftarrow \overline{a_i : A_i}\}]\!] = \mathsf{up} \circ [\![\Psi \,;\, \overline{a_i : A_i} \vdash P :: a : A]\!] \tag{142}$$

Rule (F-Var):

$$[\![\Psi, x : \tau \Vdash x : \tau]\!]u = \pi_x^{\Psi, x} u \tag{134}$$

Rule (F-Fix):

$$[\![\Psi \Vdash \mathsf{fix}\, x.M : \tau]\!]u = [\![\Psi, x : \tau \Vdash M : \tau]\!]^{\dagger} u \tag{137}$$

Rule (F-Fun):

$$[\![\Psi \Vdash \lambda x : \tau.M : \tau \to \sigma]\!]u = \mathsf{up}\,(\mathsf{strict}\,(\lambda v \in [\![\tau]\!].[\![\Psi, x : \tau \Vdash M : \sigma]\!][u \mid x \mapsto v])) \tag{135}$$

Rule (F-App):

$$[\![\Psi \Vdash MN : \sigma]\!]u = \mathsf{down}\,([\![\Psi \Vdash M : \tau \to \sigma]\!]u)\,([\![\Psi \Vdash N : \tau]\!]u) \tag{136}$$

Rule (F-Z):

$$[\![\Psi \Vdash \mathsf{o} : \mathbf{nat}]\!]u = \mathsf{o} \tag{139}$$

Rule (F-S):

$$[\![\Psi \Vdash s(M) : \mathbf{nat}]\!]u = \begin{cases} \bot & \text{if } [\![\Psi \Vdash M : \mathbf{nat}]\!]u = \bot \\ n+1 & \text{if } [\![\Psi \Vdash M : \mathbf{nat}]\!]u = n \end{cases} \tag{140}$$

8.8.2. Clauses for Process Formation (section 5.A.2).

Rule (FwD$^+$):

$$[\![\Psi\ ;\ a : A \vdash a \to b :: b : A]\!]u = \langle a : \langle A\rangle^p, b : \langle A\rangle^p\rangle \tag{113}$$

Rule (FwD$^-$):

$$[\![\Psi\ ;\ a : A \vdash a \leftarrow b :: b : A]\!]u = \langle a : \langle A\rangle^p, b : \langle A\rangle^p\rangle \tag{114}$$

Rule (Cut):

$$[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C]\!]u = [\![\Psi\ ;\ a : A, \Delta_2 \vdash Q :: c : C]\!]u \circ_a [\![\Psi\ ;\ \Delta_1 \vdash P :: a : A]\!]u \tag{115}$$

Rule (1R):

$$[\![\Psi\ ;\ \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}]\!]u\bot = \mathsf{close} \tag{122}$$

Rule (1L):

$$\begin{aligned} &[\![\Psi\ ;\ \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-) \\ &= \begin{cases} (\delta, \mathsf{close}, c) & \text{if } a^+ = \mathsf{close} \\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise} \end{cases} \\ &\quad \text{where } (\delta, c) = [\![\Psi\ ;\ \Delta \vdash P :: c : C]\!]u(\delta^+, c^-) \end{aligned} \tag{123}$$

Rule (↓R):

$$[\![\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: a : {\downarrow}A]\!]u = (\mathsf{id} \times (a : \mathsf{up})) \circ [\![\Psi\ ;\ \Delta \vdash P :: a : A]\!]u \tag{158}$$

Rule (↓L):

$$\begin{aligned} &[\![\Psi\ ;\ \Delta, a : {\downarrow}A \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-) \\ &= \begin{cases} (\delta, [a], c) & \text{if } a^+ = [a_o^+] \\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise} \end{cases} \\ &\quad \text{where } (\delta, a, c) = [\![\Psi\ ;\ \Delta, a : A \vdash P :: c : C]\!]u(\delta^+, a_o^+, c^-) \end{aligned} \tag{159}$$

Rule (↑R):

$$\begin{aligned} &[\![\Psi\ ;\ \Delta \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P :: a : {\uparrow}A]\!]u(\delta^+, a^-) \\ &= \begin{cases} (\delta, [a]) & \text{if } a^- = [a_o^-] \\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot) & \text{otherwise} \end{cases} \\ &\quad \text{where } (\delta, a) = [\![\Psi\ ;\ \Delta \vdash P :: a :]\!]u(\delta^+, a_o^-) \end{aligned} \tag{204}$$

Rule (↑L):

$$[\![\Psi\ ;\ \Delta, a : {\uparrow}A \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: c : C]\!]u = (\mathsf{id} \times (a : \mathsf{up})) \circ [\![\Psi\ ;\ \Delta, a : A \vdash P :: c : C]\!]u \tag{205}$$

Rule (⊕R):

$$\begin{aligned} &[\![\Psi\ ;\ \Delta \vdash a.k;\ P :: a : \oplus\{l : A_l\}_{l\in L}]\!]u\left(\delta^+, (a_l^-)_{l\in L}\right) = (\delta, (k, [a_k])) \\ &\quad \text{where } [\![\Psi\ ;\ \Delta \vdash P :: a : A_k]\!]u\,(\delta^+, a_k^-) = (\delta, a_k) \end{aligned} \tag{166}$$

Rule (⊕L):

$$\begin{aligned} &[\![\Psi\ ;\ \Delta, a : \oplus\{l : A_l\}_{l\in L} \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: c : C]\!]u(\delta^+, a^+, c^-) \\ &= \begin{cases} (\delta, (l, [a_l]), c) & \text{if } a^+ = (l, [a_l^+]) \\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise} \end{cases} \\ &\quad \text{where } [\![\Psi\ ;\ \Delta, a : A_l \vdash P_l :: c : C]\!]u\,(\delta^+, a_l^+, c^-) = (\delta, a_l, c) \end{aligned} \tag{167}$$

Rule (&R):

$$\begin{aligned}&[\![\Psi\,;\,\Delta \vdash \mathsf{case}\ a\ \{l \Rightarrow P_l\}_{l\in L} :: a : \&\{l : A_l\}_{l\in L}]\!]u(\delta^+, a^-)\\ &= \begin{cases}(\delta, (l, [a_l])) & \text{if } a^- = (l, [a_l^-])\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot) & \text{otherwise}\end{cases}\\ &\quad \text{where } [\![\Psi\,;\,\Delta, a : A_k \vdash P :: c : C]\!]u\,(\delta^+, a_l^-) = (\delta, a_l)\end{aligned} \tag{206}$$

Rule (&L):

$$[\![\Psi\,;\,\Delta, a : \&\{l : A_l\}_{l\in L} \vdash a.k;\ P :: c : C]\!]u(\delta^+, (a_l^+)_{l\in L}, c^-) = (\delta, (k, [a_k])\,, c) \tag{207}$$
$$\text{where } [\![\Psi\,;\,\Delta, a : A_k \vdash P :: c : C]\!]u(\delta^+, a_k^+, c^-) = (\delta, a_k, c)$$

Rule (⊗R):

$$\begin{aligned}&[\![\Psi\,;\,\Delta, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A]\!]u(\delta^+, b^+, (a_B^-, a_A^-))\\ &= (\delta, b, [(b, a)])\ \text{ where } \begin{cases}[\![\Psi\,;\,\Delta \vdash P :: a : A]\!]u(\delta^+, a_A^-) = (\delta, a)\\ \langle B\rangle^p(b^+, a_B^-) = b\end{cases}\end{aligned} \tag{130}$$

Rule (⊗L):

$$\begin{aligned}&[\![\Psi\,;\,\Delta, a : B \otimes A \vdash b \leftarrow \mathsf{recv}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, [(b, a)], c) & \text{if } a^+ = [(b_o^+, a_o^+)]\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise}\end{cases}\\ &\quad \text{where } [\![\Psi\,;\,\Delta, a : A, b : B \vdash P :: c : C]\!]u(\delta^+, a_o^+, b_o^+, c^-) = (\delta, a, b, c)\end{aligned} \tag{131}$$

Rule (⊸R):

$$\begin{aligned}&[\![\Psi\,;\,\Delta \vdash b \leftarrow \mathsf{recv}\ a;\ P :: a : B \multimap A]\!]u(\delta^+, a^-)\\ &= \begin{cases}(\delta, [(b, a)]) & \text{if } a^- = [(b_o^+, a_o^-)]\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot) & \text{otherwise}\end{cases}\\ &\quad \text{where } [\![\Psi\,;\,\Delta, b : B \vdash P :: a : A]\!]u(\delta^+, b_o^+, a_o^-) = (\delta, b, a)\end{aligned} \tag{208}$$

Rule (⊸L):

$$\begin{aligned}&[\![\Psi\,;\,\Delta, b : B, a : B \multimap A \vdash \mathsf{send}\ a\ b;\ P :: c : C]\!]u(\delta^+, b^+, (a_B^-, a_A^+), c^-)\\ &= (\delta, [(b, a)], c)\ \text{ where } \begin{cases}[\![\Psi\,;\,\Delta, a : A \vdash P :: c : C]\!]u(\delta^+, a_A^+, c^-) = (\delta, a)\\ \langle B\rangle^p(b^+, a_B^-) = b\end{cases}\end{aligned} \tag{209}$$

Rule (∧R):

$$\begin{aligned}&[\![\Psi\,;\,\Delta \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]u(\delta^+, a^-)\\ &= \begin{cases}(\delta, (v, [a])) & \text{if } [\![\Psi \Vdash M : \tau]\!]u = v \neq \bot\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \langle\tau \wedge A\rangle^p(\bot, a^-)) & \text{if } [\![\Psi \Vdash M : \tau]\!]u = \bot\end{cases}\\ &\quad \text{where } [\![\Psi\,;\,\Delta \vdash P :: a : A]\!]u(\delta^+, a^-) = (\delta, a)\end{aligned} \tag{150}$$

Rule (∧L):

$$\begin{aligned}&[\![\Psi\,;\,\Delta, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, (v, [a]), c) & \text{if } a^+ = (v, [a_o^+])\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise}\end{cases}\\ &\quad \text{where } [\![\Psi, x : \tau\,;\,\Delta, a : A \vdash P :: c : C]\!][u \mid x \mapsto v](\delta^+, a_o^+, c^-) = (\delta, a, c)\end{aligned} \tag{151}$$

Rule (⊃L):

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, a : \tau \supset A \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, (v, [a])\,, c) & \text{if } [\![\Psi \Vdash M : \tau]\!]u = v \neq \bot\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \langle\tau \supset A\rangle^p(a^+, \bot), \langle C\rangle^p(\bot, c^-)) & \text{if } [\![\Psi \Vdash M : \tau]\!]u = \bot\end{cases}\\ &\quad \text{where } [\![\Psi\ ;\ \Delta, a : A \vdash P :: c : C]\!]u(\delta^+, a^+, c^-) = (\delta, a, c)\end{aligned} \tag{210}$$

Rule (⊃R):

$$\begin{aligned}&[\![\Psi\ ;\ \Delta \vdash x \leftarrow \mathsf{input}\ a;\ P :: a : \tau \supset A]\!]u(\delta^+, a^-)\\ &= \begin{cases}(\delta, (v, [a])) & \text{if } a^- = (v, [a_o^-])\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot) & \text{otherwise}\end{cases}\\ &\quad \text{where } [\![\Psi, x : \tau\ ;\ \Delta \vdash P :: a : A]\!][u \mid x \mapsto v](\delta^+, a_o^-) = (\delta, a)\end{aligned} \tag{211}$$

Rule (ρ^+R):

$$\begin{aligned}&[\![\Psi\ ;\ \Delta \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A]\!]u\\ &= (\mathsf{id} \times (a : \mathsf{Fold} \circ \mathsf{up})) \circ [\![\Psi\ ;\ \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u \circ (\mathsf{id} \times (a^- : \mathsf{Unfold}))\end{aligned} \tag{188}$$

Rule (ρ^+L):

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, a : \rho\alpha.A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: c : C]\!]u(\delta^+, a^+, c^-)\\ &= \begin{cases}(\delta, \mathsf{Fold}([a]), c) & \text{if } a^+ = \mathsf{Fold}([a_o^+])\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot, \langle C\rangle^p(\bot, c^-)) & \text{otherwise}\end{cases}\\ &\quad \text{where } (\delta, a, c) = [\![\Psi\ ;\ \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C]\!]u(\delta^+, a_o^+, c^-)\end{aligned} \tag{189}$$

Rule (ρ^-R):

$$\begin{aligned}&[\![\Psi\ ;\ \Delta \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P :: a : \rho\alpha.A]\!]u(\delta^+, a^-)\\ &= \begin{cases}(\delta, \mathsf{Fold}([a])) & \text{if } a^- = \mathsf{Fold}([a_o^-])\\ ((\langle\Delta\rangle^p(\delta^+, \bot), \bot) & \text{otherwise}\end{cases}\\ &\quad \text{where } (\delta, a) = [\![\Psi\ ;\ \Delta \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u(\delta^+, a_o^-)\end{aligned} \tag{212}$$

Rule (ρ^-L):

$$\begin{aligned}&[\![\Psi\ ;\ \Delta, a : \rho\alpha.A \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: c : C]\!]u\\ &= (\mathsf{id} \times (a : \mathsf{Fold} \circ \mathsf{up})) \circ [\![\Psi\ ;\ \Delta, a : [\rho\alpha.A/\alpha]A \vdash P :: c : C]\!]u \circ (\mathsf{id} \times (a^+ : \mathsf{Unfold}))\end{aligned} \tag{213}$$

Rule (E-{}):

$$[\![\Psi\ ;\ \overline{a_i : A_i} \vdash a \leftarrow \{M\} \leftarrow \overline{a}_i :: a : A]\!] = \mathsf{down} \circ [\![\Psi \Vdash M : \{a : A \leftarrow \overline{a_i : A_i}\}]\!] \tag{143}$$

8.8.3. Clauses for Type Formation (section 5.A.3).

Rule (C1):

$$[\![\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+]\!] = \mathsf{diag}_{[\![\Xi]\!]}\{\bot \Subset \mathsf{close}\} \tag{116}$$

$$[\![\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+]\!]^+ = \mathsf{diag}_{[\![\Xi]\!]}\{\bot \Subset \mathsf{close}\} \tag{117}$$

$$[\![\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+]\!]^- = \mathsf{diag}_{[\![\Xi]\!]}\,\top_{\mathbf{Stab}} \tag{118}$$

$$\langle\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+\rangle^+ = \mathsf{id} \tag{119}$$

$$\langle\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+\rangle^- = \top \tag{120}$$

$$\langle\Xi \vdash \mathbf{1}\ \mathsf{type}_s^+\rangle^p = \pi_1 \tag{121}$$

Rule (CVar):

$$[\![\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p]\!] = \pi_\alpha^{\Xi,\alpha} \tag{168}$$
$$[\![\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p]\!]^+ = \pi_\alpha^{\Xi,\alpha} \tag{169}$$
$$[\![\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p]\!]^- = \pi_\alpha^{\Xi,\alpha} \tag{170}$$
$$\langle\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p\rangle^+ = \mathsf{id} \tag{171}$$
$$\langle\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p\rangle^- = \mathsf{id} \tag{172}$$
$$\langle\Xi, \alpha\ \mathsf{type}_s^p \vdash \alpha\ \mathsf{type}_s^p\rangle^p = \sqcap \tag{173}$$

Rule (Cρ^+):

$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!] = ((-)_\perp [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!])^\dagger \tag{179}$$
$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^+ = ((-)_\perp [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^+)^\dagger \tag{180}$$
$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^- = ([\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^-)^\dagger \tag{181}$$
$$\langle\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+\rangle^+ = ((-)_\perp \langle\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+\rangle^+)^\dagger \tag{182}$$
$$\langle\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+\rangle^- = (\mathsf{down} * \langle\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+\rangle^-)^\dagger \tag{183}$$
$$\langle\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+\rangle^p = \mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A\rangle^p \circ \delta \circ (\mathsf{Unfold} \times \mathsf{Unfold}) \tag{187}$$

Rule (Cρ^-):

$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^-]\!] = ((-)_\perp [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!])^\dagger \tag{214}$$
$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^-]\!]^+ = ([\![\Xi, \alpha\ \mathsf{type}_s^- \vdash A\ \mathsf{type}_s^-]\!]^+)^\dagger \tag{215}$$
$$[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^-]\!]^- = ((-)_\perp [\![\Xi, \alpha\ \mathsf{type}_s^- \vdash A\ \mathsf{type}_s^-]\!]^-)^\dagger \tag{216}$$
$$\langle\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^-\rangle^+ = (\mathsf{down} * \langle\Xi, \alpha\ \mathsf{type}_s^- \vdash A\ \mathsf{type}_s^-\rangle^+)^\dagger \tag{217}$$
$$\langle\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^-\rangle^- = ((-)_\perp \langle\Xi, \alpha\ \mathsf{type}_s^- \vdash A\ \mathsf{type}_s^-\rangle^-)^\dagger \tag{218}$$
$$\langle\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+\rangle^p = \mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A\rangle^p \circ \delta \circ (\mathsf{Unfold} \times \mathsf{Unfold}) \tag{219}$$

Rule (C↓):

$$[\![\Xi \vdash \downarrow A\ \mathsf{type}_s^+]\!] = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]_\perp \tag{152}$$
$$[\![\Xi \vdash \downarrow A\ \mathsf{type}_s^+]\!]^+ = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]_\perp^+ \tag{153}$$
$$[\![\Xi \vdash \downarrow A\ \mathsf{type}_s^+]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]^- \tag{154}$$
$$\langle\Xi \vdash \downarrow A\ \mathsf{type}_s^+\rangle^+ = (-)_\perp \langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^+ \tag{155}$$
$$\langle\Xi \vdash \downarrow A\ \mathsf{type}_s^+\rangle^- = \mathsf{down} * \langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^- \tag{156}$$
$$\langle\Xi \vdash \downarrow A\ \mathsf{type}_s^+\rangle^p = (-)_\perp \langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^p \cdot \delta \tag{157}$$

Rule (C↑):

$$[\![\Xi \vdash \uparrow A\ \mathsf{type}_s^-]\!] = [\![\Xi \vdash A\ \mathsf{type}_s^+]\!]_\perp \tag{220}$$
$$[\![\Xi \vdash \uparrow A\ \mathsf{type}_s^-]\!]^+ = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]^+ \tag{221}$$
$$[\![\Xi \vdash \uparrow A\ \mathsf{type}_s^-]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]_\perp^- \tag{222}$$
$$\langle\Xi \vdash \uparrow A\ \mathsf{type}_s^-\rangle^+ = \mathsf{down} * \langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^+ \tag{223}$$
$$\langle\Xi \vdash \uparrow A\ \mathsf{type}_s^-\rangle^- = (-)_\perp \langle\Xi \vdash A\ \mathsf{type}_s^-\rangle^- \tag{224}$$
$$\langle\Xi \vdash \uparrow A\ \mathsf{type}_s^-\rangle^p = (-)_\perp \langle\Xi \vdash A\ \mathsf{type}_s^+\rangle^p \cdot \delta \tag{225}$$

Rule (C⊕):

$$[\![\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_s^+]\!] = \bigoplus_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_s^+]\!]_\perp \tag{160}$$

$$[\![\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_s^+]\!]^+ = \bigoplus_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_s^+]\!]^+_\perp \tag{161}$$

$$[\![\Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_s^+]\!]^- = \prod_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_s^+]\!]^- \tag{162}$$

$$\langle\!\langle \Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_s^+\rangle\!\rangle^+ = \bigoplus_{l\in L} (-)_\perp \langle\!\langle \Xi \vdash A_l\ \mathsf{type}_s^+\rangle\!\rangle^+ \tag{163}$$

$$\langle\!\langle \Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_s^+\rangle\!\rangle^- = \mathrm{diag}\,(\mathsf{down} * \langle\!\langle \Xi \vdash A_l\ \mathsf{type}_s^+\rangle\!\rangle^-)_{l\in L} \tag{164}$$

$$\langle\!\langle \Xi \vdash \oplus\{l : A_l\}_{l\in L}\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi((k,[a_k^+]),(a_l^-)_{l\in L}) = (k,[\langle\!\langle \Xi \vdash A_k\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(a_k^+,a_k^-)]) \tag{165}$$

Rule (C&):

$$[\![\Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_s^-]\!] = \bigoplus_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_s^-]\!]_\perp \tag{226}$$

$$[\![\Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_s^-]\!]^+ = \prod_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_s^-]\!]^+ \tag{227}$$

$$[\![\Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_s^-]\!]^- = \bigoplus_{l\in L} [\![\Xi \vdash A_l\ \mathsf{type}_s^-]\!]^-_\perp \tag{228}$$

$$\langle\!\langle \Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_s^-\rangle\!\rangle^+ = \mathrm{diag}\,(\mathsf{down} * \langle\!\langle \Xi \vdash A_l\ \mathsf{type}_s^-\rangle\!\rangle^+)_{l\in L} \tag{229}$$

$$\langle\!\langle \Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_s^-\rangle\!\rangle^- = \bigoplus_{l\in L} (-)_\perp \langle\!\langle \Xi \vdash A_l\ \mathsf{type}_s^-\rangle\!\rangle^- \tag{230}$$

$$\langle\!\langle \Xi \vdash \&\{l : A_l\}_{l\in L}\ \mathsf{type}_s^-\rangle\!\rangle^p_\xi((a_l^+)_{l\in L},(k,[a_k^-])) = (k,[\langle\!\langle \Xi \vdash A_k\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(a_k^+,a_k^-)]) \tag{231}$$

Rule (C⊗):

$$[\![\Xi \vdash A \otimes B\ \mathsf{type}_s^+]\!] = ([\![\Xi \vdash A\ \mathsf{type}_s^+]\!] \times [\![\Xi \vdash B\ \mathsf{type}_s^+]\!])_\perp \tag{124}$$

$$[\![\Xi \vdash A \otimes B\ \mathsf{type}_s^+]\!]^+ = ([\![\Xi \vdash A\ \mathsf{type}_s^+]\!]^+ \times [\![\Xi \vdash B\ \mathsf{type}_s^+]\!]^+)_\perp \tag{125}$$

$$[\![\Xi \vdash A \otimes B\ \mathsf{type}_s^+]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_s^+]\!]^- \times [\![\Xi \vdash B\ \mathsf{type}_s^+]\!]^- \tag{126}$$

$$\langle\!\langle \Xi \vdash A \otimes B\ \mathsf{type}_s^+\rangle\!\rangle^+ = (-)_\perp\,(\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^+ \times \langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^+) \tag{127}$$

$$\langle\!\langle \Xi \vdash A \otimes B\ \mathsf{type}_s^+\rangle\!\rangle^- = \mathsf{down} * (\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^- \times \langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^-) \tag{128}$$

$$\begin{aligned}&\langle\!\langle \Xi \vdash A \otimes B\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi([(a^+,b^+)],(a^-,b^-))\\ &= [(\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(a^+,a^-),\langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(b^+,b^-))]\end{aligned} \tag{129}$$

Rule (C⊸):

$$[\![\Xi \vdash B \multimap A\ \mathsf{type}_s^-]\!] = ([\![\Xi \vdash B\ \mathsf{type}_s^+]\!] \times [\![\Xi \vdash A\ \mathsf{type}_s^-]\!])_\perp \tag{232}$$

$$[\![\Xi \vdash B \multimap A\ \mathsf{type}_s^-]\!]^+ = [\![\Xi \vdash B\ \mathsf{type}_s^+]\!]^- \times [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]^+ \tag{233}$$

$$[\![\Xi \vdash B \multimap A\ \mathsf{type}_s^-]\!]^- = (-)_\perp\,([\![\Xi \vdash B\ \mathsf{type}_s^+]\!]^+ \times [\![\Xi \vdash A\ \mathsf{type}_s^-]\!]^-) \tag{234}$$

$$\langle\!\langle \Xi \vdash B \multimap A\ \mathsf{type}_s^-\rangle\!\rangle^+ = \mathsf{down} * (\langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^- \times \langle\!\langle \Xi \vdash A\ \mathsf{type}_s^-\rangle\!\rangle^+) \tag{235}$$

$$\langle\!\langle \Xi \vdash B \multimap A\ \mathsf{type}_s^-\rangle\!\rangle^- = (-)_\perp\,(\langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^+ \times \langle\!\langle \Xi \vdash A\ \mathsf{type}_s^-\rangle\!\rangle^-) \tag{236}$$

$$\begin{aligned}&\langle\!\langle \Xi \vdash B \multimap A\ \mathsf{type}_s^-\rangle\!\rangle^p_\xi((b^-,a+),[(b^+,a^-)])\\ &= [(\langle\!\langle \Xi \vdash A\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(b^+,b^-),\langle\!\langle \Xi \vdash B\ \mathsf{type}_s^+\rangle\!\rangle^p_\xi(a^+,a^-))]\end{aligned} \tag{237}$$

Rule (C∧):

$$[\![\Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+]\!] = [\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!] \otimes [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+]\!]_\perp \tag{144}$$

$$[\![\Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+]\!]^+ = [\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!] \otimes [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+]\!]_\perp^+ \tag{145}$$

$$[\![\Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+]\!]^- = [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+]\!]^- \tag{146}$$

$$\langle \Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+ \rangle^+ = \mathsf{id}_{[\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!]} \otimes (-)_\perp \langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^+ \rangle^+ \tag{147}$$

$$\langle \Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+ \rangle^- = \mathsf{down} * \pi_2 * \langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^+ \rangle^- \tag{148}$$

$$\langle \Xi \vdash \tau \wedge A\ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p((v, [a^+]), a^-) = (v, [\langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^+ \rangle_\xi^p(a^+, a^-)]) \tag{149}$$

Rule (C⊃):

$$[\![\Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^-]\!] = [\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!] \otimes [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^-]\!]_\perp \tag{238}$$

$$[\![\Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^-]\!]^+ = [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^-]\!]^+ \tag{239}$$

$$[\![\Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^-]\!]^- = [\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!] \otimes [\![\Xi \vdash A\ \mathsf{type}_\mathsf{s}^-]\!]_\perp^- \tag{240}$$

$$\langle \Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^- \rangle^+ = \mathsf{down} * \pi_2 * \langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^- \rangle^+ \tag{241}$$

$$\langle \Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^- \rangle^- = \mathsf{id}_{[\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!]} \otimes (-)_\perp \langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^- \rangle^- \tag{242}$$

$$\langle \Xi \vdash \tau \supset A\ \mathsf{type}_\mathsf{s}^- \rangle_\xi^p(a^+, (v, [a^-])) = (v, [\langle \Xi \vdash A\ \mathsf{type}_\mathsf{s}^- \rangle_\xi^p(a^+, a^-)]) \tag{243}$$

Rule (T{}):

$$\begin{aligned}&[\![\Xi \vdash \{a_0 : A_0 \leftarrow a_1 : A_1, \ldots, a_n : A_n\}\ \mathsf{type}_\mathsf{f}]\!] \\ &= (-)_\perp \circ \mathsf{diag}_{[\![\Xi]\!]}\left(\mathbf{JFC}\left[\langle \Xi \vdash A_1\ \mathsf{type}_\mathsf{s}\rangle_\perp, \ldots, \langle \Xi \vdash A_n\ \mathsf{type}_\mathsf{s}\rangle_\perp \to \langle \Xi \vdash A_0\ \mathsf{type}_\mathsf{s}\rangle_\perp\right]\right)\end{aligned} \tag{141}$$

Rule (T→): When the derivation of $\Xi \vdash \tau \to \sigma\ \mathsf{type}_\mathsf{f}$ respectively does and does not use (T{}):

$$[\![\Xi \vdash \tau \to \sigma\ \mathsf{type}_\mathsf{f}]\!] = \mathsf{diag}_{[\![\Xi]\!]}\left(\left(\mathbf{DCPO}_{\perp!}\left[[\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!] \to [\![\Xi \vdash \sigma\ \mathsf{type}_\mathsf{f}]\!]\right]\right)_\perp\right) \tag{133}$$

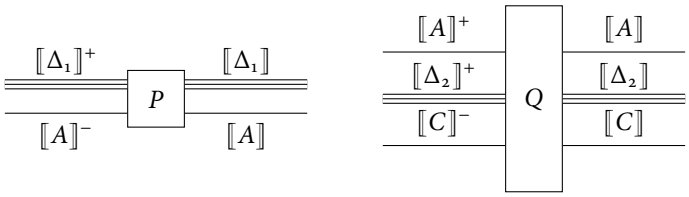
$$[\![\Xi \vdash \tau \to \sigma\ \mathsf{type}_\mathsf{f}]\!] = \mathsf{diag}_{[\![\Xi]\!]}\left(\left(\mathbf{Stab}_{\perp!}\left[[\![\Xi \vdash \tau\ \mathsf{type}_\mathsf{f}]\!] \to [\![\Xi \vdash \sigma\ \mathsf{type}_\mathsf{f}]\!]\right]\right)_\perp\right) \tag{132}$$

CHAPTER 9

Equivalence, Applied

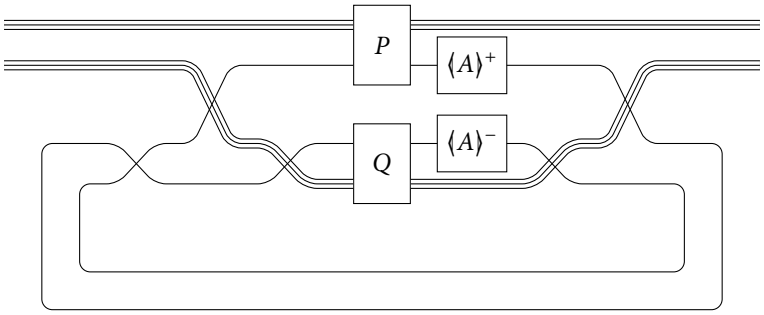
In this chapter, we apply our denotational techniques to show various program equivalences. In particular, we show that our denotational semantics satisfies a collection of η-style properties and commuting conversions in section 9.1. We characterize the denotations of purely positive and negative types in section 9.2. In section 9.3, we revisit example 5.3.10 to show that flipping the bits in a bit stream twice is equivalent to forwarding the bit stream. We show that the computational interpretation of the identity expansion of intuitionistic linear logic coincides with forwarding in section 9.4. Finally, we study binary arithmetic in section 9.5.

Many of the equivalences involve manipulating string diagrams for process compositions in the traced monoidal category $\mathbf{Stab}_{\perp!}$. Given processes $\Psi \mathbin{;} \Delta_1 \vdash P :: a : A$ and $\Psi \mathbin{;} a : A, \Delta_2 \vdash Q :: c : C$ and an environment $u \in [\![\Psi]\!]$, the morphisms $[\![\Psi \mathbin{;} \Delta_1 \vdash P :: a : A]\!]u$ and $[\![\Psi \mathbin{;} a : A, \Delta_2 \vdash Q :: c : C]\!]u$ respectively denote the string diagrams



Here, we've abbreviated the multiple input and output wires associated with each component of Δ_1 and Δ_2 by triple lines. In general, we will group together wires that are not of interest in this manner. We will also generally elide the object labels on wires: they will be clear from context.

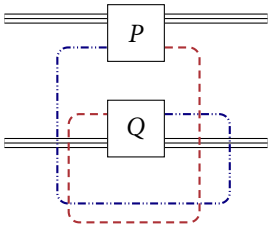
Using these conventions, the composition $\Psi \mathbin{;} \Delta_1, \Delta_2 \vdash a \leftarrow P;\ Q :: c : C$ then denotes the string diagram:[1]



We adopt some convenient notation for string diagrams denoting compositions of processes. We are free to position the positive and negative projections anywhere along the wire thanks to the sliding axiom. As a result, we use red dashed wires "- - - - - -" to represent wires with an (implicit) positive projection, and blue dashed and dotted wires "-·-·-·-·-" for wires with an (implicit) negative projection. We may label these wires with their associated type. In light of theorem 2.3.6, we also

[1] In this diagram, we have implicitly used the vanishing axiom to depict fixing the component $[\![A]\!]^+ \times [\![A]\!]^-$ as fixing the components $[\![A]\!]^+$ and $[\![A]\!]^-$ separately.

allow ourselves a significant liberty with the layout of our string diagrams.[2] Consequently, we may depict the above diagram as follows:



9.1. η-Style Properties

> Règle très appréciée des étudiants, car elle sert à étoffer des thèses peu fournies : un chapitre sur « η » amène son lot de complications techniques prévisibles et fastidieuses, 100% de transpiration, 0% d'inspiration ; bref, cela consomme du papier.
>
> Jean-Yves Girard [Gir06, p. 166]

η-style properties capture the fact that we have enough communication destructors for each communication constructor. They correspond to the principal or key cases of the cut-elimination algorithm that drives communication in Polarized SILL. Though their proof requires no inspiration, it also requires no perspiration (cf. the epigraph). This is because each η-style property follows from an easy manipulation of string diagrams.

Proposition 9.1.1 (η-style Properties). *The following semantic equivalences hold for appropriately typed processes P, Q, P_l, and Q_l:*

$$\Psi\ ;\ \Delta \vdash P \equiv a \leftarrow \mathsf{close}\ a;\ \mathsf{wait}\ a;\ P :: c : C \tag{244}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ Q \equiv a \leftarrow (\mathsf{send}\ a\ \mathsf{shift};\ P);\ (\mathsf{shift} \leftarrow \mathsf{recv}\ a;\ Q) :: c : C \tag{245}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ Q \equiv a \leftarrow (\mathsf{shift} \leftarrow \mathsf{recv}\ a;\ P);\ (\mathsf{send}\ a\ \mathsf{shift};\ Q) :: c : C \tag{246}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ Q_k \equiv a \leftarrow (a.k;\ P);\ \mathsf{case}\ a\ \{l \Rightarrow Q_l\} :: c : C \tag{247}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P_k;\ Q \equiv a \leftarrow \mathsf{case}\ a\ \{l \Rightarrow P_l\};\ (a.k;\ Q) :: c : C \tag{248}$$

$$\Psi\ ;\ \Delta, b : B \vdash a \leftarrow P;\ Q \equiv a \leftarrow (\mathsf{send}\ a\ b;\ P);\ (b \leftarrow \mathsf{recv}\ a;\ Q) :: c : C \tag{249}$$

$$\Psi\ ;\ \Delta, b : B \vdash a \leftarrow P;\ Q \equiv a \leftarrow (b \leftarrow \mathsf{recv}\ a;\ Q);\ (\mathsf{send}\ a\ b;\ P) :: c : C \tag{250}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ [M/x]Q \equiv a \leftarrow (\_ \leftarrow \mathsf{output}\ a\ M;\ P);\ (x \leftarrow \mathsf{input}\ Q;\ ) :: c : C \tag{251}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow [M/x]P;\ Q \equiv a \leftarrow (x \leftarrow \mathsf{input}\ P;\ );\ (\_ \leftarrow \mathsf{output}\ a\ M;\ Q) :: c : C \tag{252}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ Q \equiv a \leftarrow (\mathsf{send}\ a\ \mathsf{unfold};\ P);\ (\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ Q) :: c : C \tag{253}$$

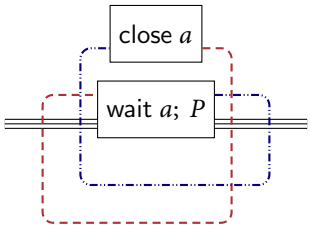
$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ Q \equiv a \leftarrow (\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ P);\ (\mathsf{send}\ a\ \mathsf{unfold};\ Q) :: c : C \tag{254}$$

Equivalences (251) *and* (252) *are subject to the side condition that* $[\![\Psi \Vdash M : \tau]\!]u \neq \bot$ *for all* $u \in [\![\Psi]\!]$.

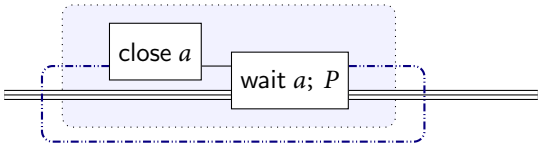
Proof. Each equivalence follows by a manipulation of string diagrams. We show the cases for cuts along positive channels. The cases for cuts along negative channels will follow by symmetry.

[2] In all cases, our diagrams will be morally correct, i.e., their wires and boxes can be rearranged into technically correct diagrams.

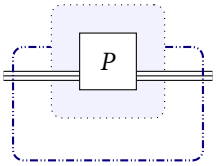
We start with eq. (244). Fix some arbitrary $u \in [\![\Psi]\!]$. The composition $[\![\Psi \;;\; \Delta \vdash a \leftarrow \mathsf{close}\ a;\ \mathsf{wait}\ a;\ P :: c : C]\!]u$ denotes the string diagram



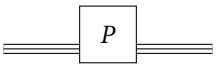
where the boxes respectively represent the morphisms $[\![\Psi \;;\; \cdot \vdash \mathsf{close}\ a :: a : \mathbf{1}]\!]u$ and $[\![\Psi \;;\; \Delta, a : \mathbf{1} \vdash \mathsf{wait}\ a;\ P :: c : C]\!]u$. By eq. (119), the positive wire is the identity morphism, while by eq. (120), the negative wire is the constantly bottom morphism.



By eq. (123), we recognize the composition in the shaded area as $\rho^{-1} \circ [\![\Psi \;;\; \Delta \vdash P :: c : C]\!]u \circ \rho$, so the diagram is equal to:



where the box P is the morphism $[\![\Psi \;;\; \Delta \vdash P :: c : C]\!]u$. But the trace is fixing the monoidal unit, so by vanishing the diagram is equal to:

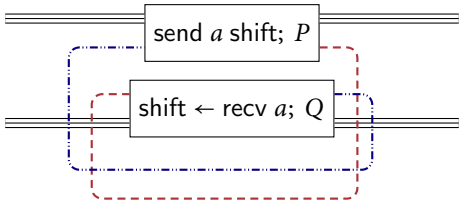


We conclude that

$$[\![\Psi \;;\; \Delta \vdash a \leftarrow \mathsf{close}\ a;\ \mathsf{wait}\ a;\ P :: c : C]\!]u = [\![\Psi \;;\; \Delta \vdash P :: c : C]\!]u$$

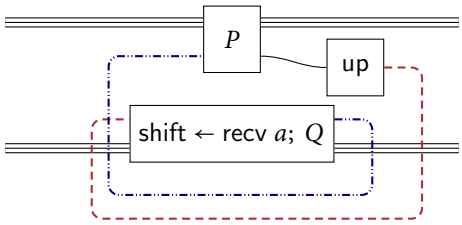
as desired.

Now we show eq. (245). Fix some arbitrary $u \in [\![\Psi]\!]$. The composition $[\![\Psi \;;\; \Delta_1, \Delta_2 \vdash a \leftarrow \mathsf{send}\ a\ \mathsf{shift};\ P;\ \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ Q :: c : C]\!]u$ denotes the string diagram

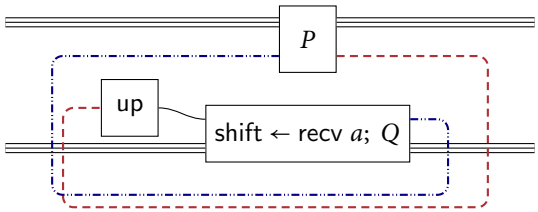


where the boxes respectively represent the morphisms $[\![\Psi \;;\; \Delta_1 \vdash \mathsf{send}\ a\ \mathsf{shift};\ P :: a : \downarrow A]\!]u$ and $[\![\Psi \;;\; \Delta_2, a : \downarrow A \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ Q :: c : C]\!]u$. Expanding eq. (158), we see that this diagram is

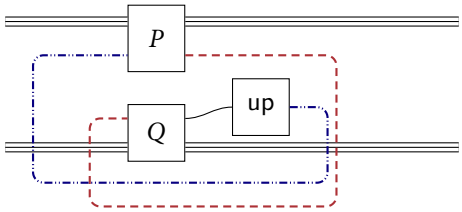
equal to:



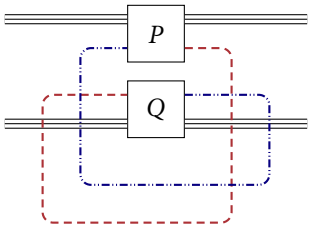
where the box P is the morphism $[\![\Psi\ ;\ \Delta_1 \vdash P :: a : A]\!]u$. By diagram 2 and eq. (155), $\langle\downarrow A\rangle^+ \circ \mathsf{up} = \mathsf{up} \circ \langle A\rangle^+$. Using this fact and rearranging the diagram, we get:



where the positive projection is $\langle A\rangle^+$. Expanding eq. (159), the diagram becomes

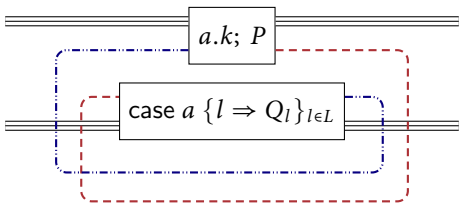


where the box Q is the morphism $[\![\Psi\ ;\ \Delta_2, a : A \vdash Q :: c : C]\!]u$. Taking into account the definition of $\langle\downarrow A\rangle^-$ (eq. (156)), the diagram is equal to:



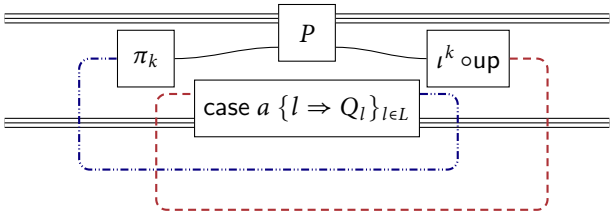
This is what we wanted to show.

Next, we show eq. (247). The composition $[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow a.k;\ P;\ \mathsf{case}\ a\ \{l \Rightarrow Q_l\}_{l\in L} :: c : C]\!]u$ denotes the string diagram



where the boxes respectively represent the morphisms $[\![\Psi\ ;\ \Delta_1 \vdash a.k;\ P :: a : \oplus\{l \Rightarrow A_l\}_{l\in L}]\!]u$ and $[\![\Psi\ ;\ \Delta_2, a : \oplus\{l \Rightarrow A_l\}_{l\in L} \vdash \mathsf{case}\ a\ \{l \Rightarrow Q_l\}_{l\in L} :: c : C]\!]u$. Expanding the definition of $a.k;\ P$,

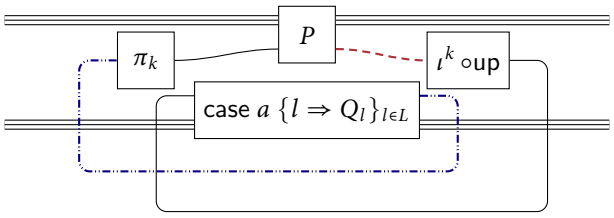
the above diagram is equal to:



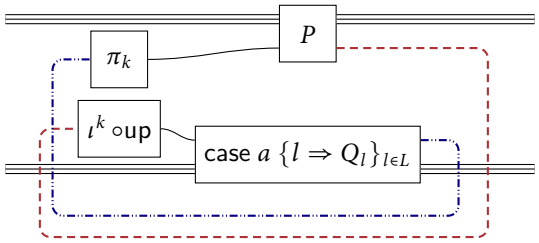
where the box P is $[\![\Psi\ ;\ \Delta_1 \vdash P :: a : A_k]\!]u$. Observe that

$$\langle \oplus\{l \Rightarrow A_l\}_{l \in L} \rangle^+ \circ \iota^k \circ \mathsf{up} = \iota^k \circ \mathsf{up} \circ \langle A_k \rangle^+.$$

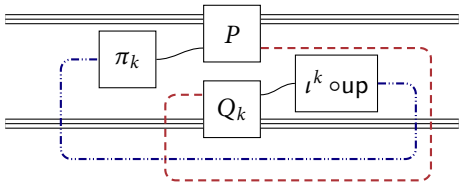
So the diagram is equal to:



where the positive projection used is now $\langle A_k \rangle^+$. Rearranging the diagram gives:



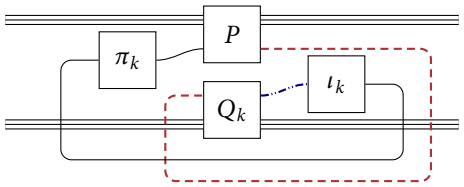
Expanding eq. (167), this diagram is in turn equal to:



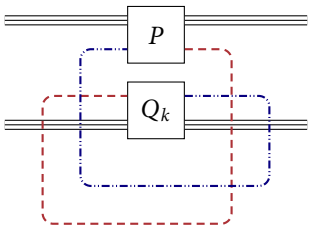
where the box Q_k is $[\![\Psi\ ;\ \Delta_2, a : A_k \vdash Q_k :: c : C]\!]u$. Expanding the definition of the negative projection (eq. (163)) and observing that

$$\langle \oplus\{l \Rightarrow A_l\}_{l \in L} \rangle^- \circ \iota^k \circ \mathsf{up} = \iota_k \circ \langle A_k \rangle^-,$$

the diagram becomes equal to:

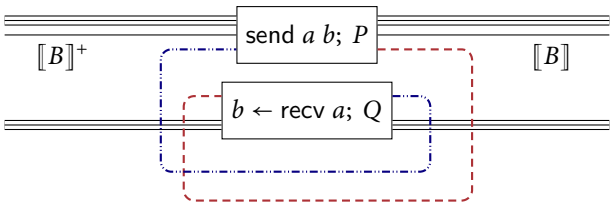


where negative projection is now $\langle A_k \rangle^-$. Rearranging this diagram to place ι_k to the left of π_k, we observe that they cancel out, and the diagram then becomes:



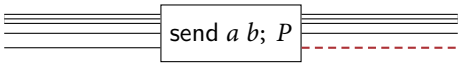
This is what we wanted to show.

We now show the η-style property for tensors, i.e., eq. (249). Fix some arbitrary $u \in [\![\Psi]\!]$. The composition $[\![\Psi \; ; \; \Delta_1, \Delta_2, b : B \vdash a \leftarrow \mathsf{send}\ a\ b;\ P;\ b \leftarrow \mathsf{recv}\ a;\ Q :: c : C]\!]u$ denotes the string diagram

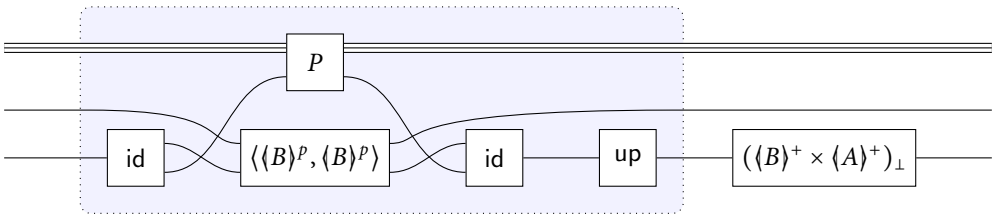


(255)

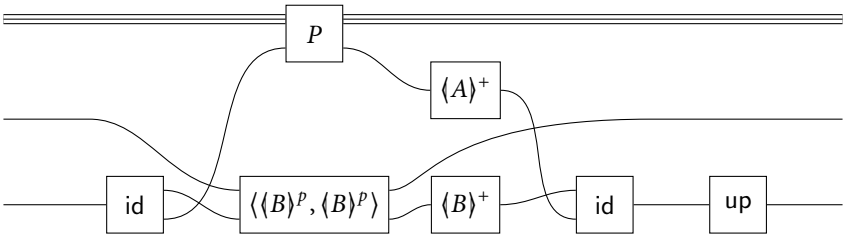
where the boxes respectively represent the morphisms $[\![\Psi \; ; \; \Delta_1, b : B \vdash \mathsf{send}\ a\ b;\ P :: a : B \otimes A]\!]u$ and $[\![\Psi \; ; \; \Delta_2, a : B \otimes A \vdash b \leftarrow \mathsf{recv}\ a;\ Q :: c : C]\!]u$. We begin by expanding the definitions of each box. By eq. (130), the diagram



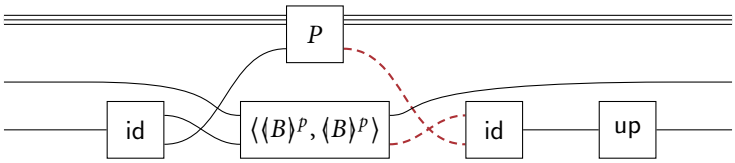
is equal to



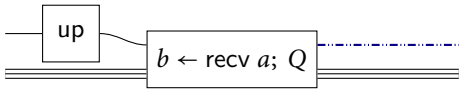
In this diagram, we use the identity morphism as a convenient notation to shuttle between the two roles of the categorical product: $A \times B$ is simulatenously the tensor of A and B (represented by a pair of wires) and the categorical product of A and B (represented by a single wire). By diagram 2, $(\langle B \rangle^+ \times \langle A \rangle^+)_\perp \circ \mathsf{up} = \mathsf{up} \circ (\langle B \rangle^+ \times \langle A \rangle^+)$, so the sequence of three morphisms in the bottom right corner can be rewritten to give:



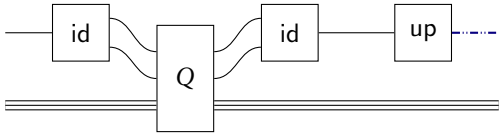
i.e., the diagram



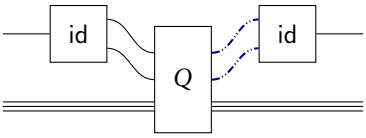
By eq. (131), the diagram



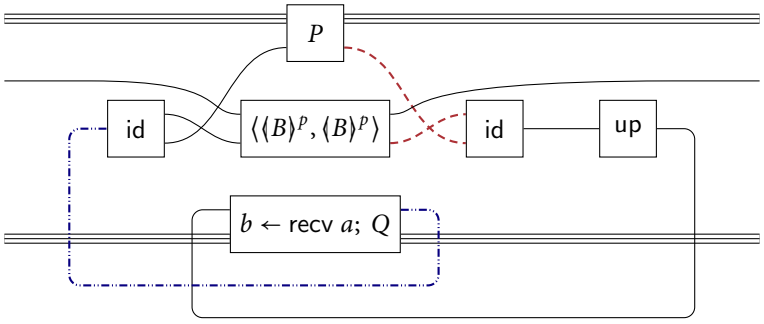
is equal to



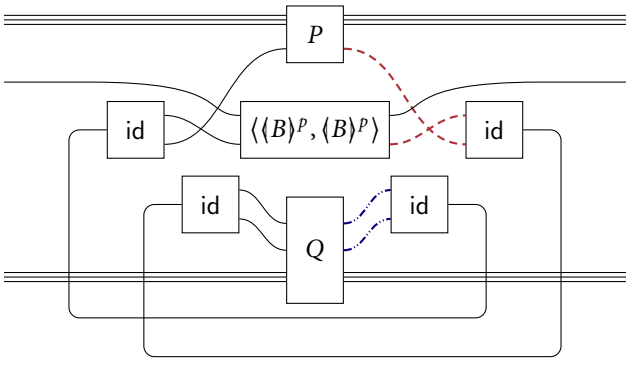
which by eq. (128) is equal to



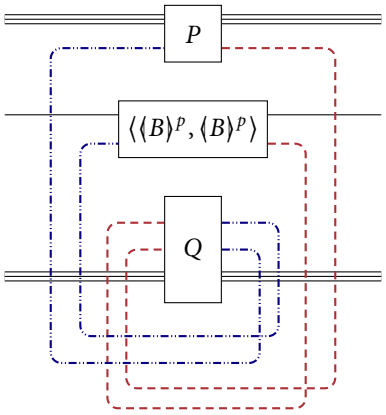
Using these observations, we deduce that diagram 255 is equal to:



By sliding and action, we can move the rightmost up the to left of $b \leftarrow \mathsf{recv}\ a;\ Q$, and the diagram then simplifies to:



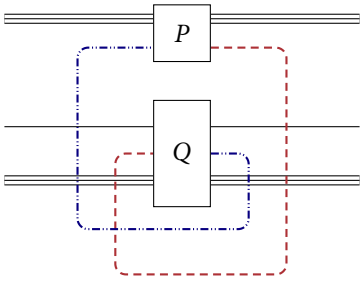
We can now use vanishing and action to shift the identity morphisms to be side by side, and we observe that they cancel out. The diagram becomes:



Recall that $\langle B \rangle$ is well-woven and that $[\![\Psi \;;\; \Delta_2, a : A, b : B \vdash Q :: c : C]\!]u$ is a morphism of **CYO**(**Stab**). This implies that

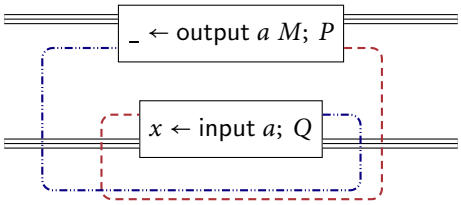
$$[\![\Psi \;;\; \Delta_2, a : A, b : B \vdash Q :: c : C]\!]u \circ \mathsf{id}_{\langle B \rangle} = [\![\Psi \;;\; \Delta_2, a : A, b : B \vdash Q :: c : C]\!]u,$$

i.e., that the above diagram is equal to:

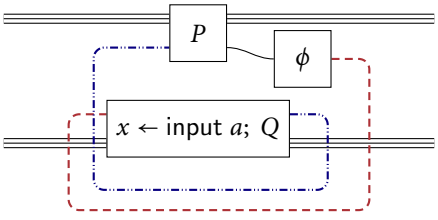


This is exactly what we wanted to show.

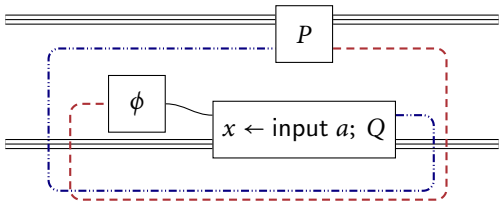
Next, we show eq. (251). Fix some environment $u \in [\![\Psi]\!]$. By assumption, $[\![\Psi \Vdash M : \tau]\!]u = v$ for some $v \neq \bot$. The composition $[\![\Psi \;;\; \Delta_1, \Delta_2 \vdash a \leftarrow \_ \leftarrow \mathsf{output}\ a\ M;\ P;\ x \leftarrow \mathsf{input}\ a;\ Q :: c : C]\!]u$ denotes the string diagram



where the boxes respectively represent the morphisms $[\![\Psi \;;\; \Delta_1 \vdash \_ \leftarrow \mathsf{output}\ a\ M;\ P :: a : \tau \wedge A]\!]u$ and $[\![\Psi \;;\; \Delta_2, a : \tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ Q :: c : C]\!]u$. Expanding eq. (150), this diagram is equal to:



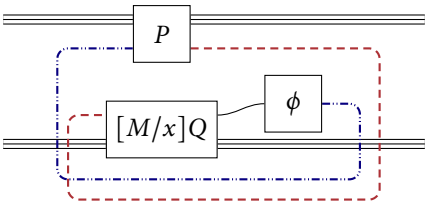
where $\phi = \lambda a \in [\![A]\!].(v, [a])$ and P is the morphism $[\![\Psi \;;\; \Delta_1 \vdash P :: a : A]\!]u$. Observe that $\langle \tau \wedge A \rangle^+ \circ \phi = \phi \circ \langle A \rangle^+$, so after sliding, the diagram is equal to:



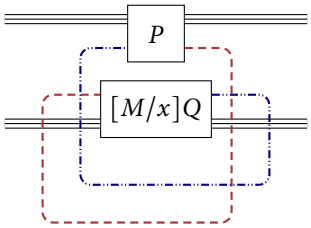
By semantic substitution (proposition 8.5.6),

$$[\![\Psi, x : \tau \;;\; \Delta_2, a : A \vdash Q :: c : C]\!][u \mid x \mapsto v] = [\![\Psi \;;\; \Delta_2, a : A \vdash [M/x]Q :: c : C]\!]u.$$

Expanding eq. (151), we can simplify the bottom portion of the diagram to get:

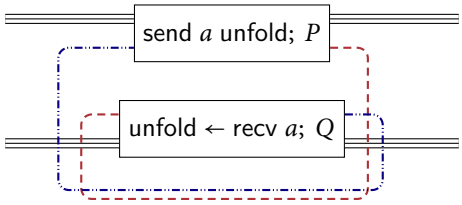


where the box $[M/x]Q$ is the morphism $[\![\Psi \;;\; \Delta_2, a : A \vdash [M/x]Q :: c : C]\!]u$. Finally, observe that $\langle \tau \wedge A \rangle^- \circ \phi = \langle A \rangle^-$, so the diagram is equal to to



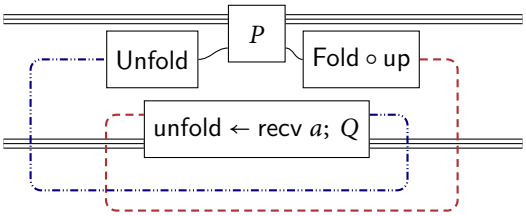
This is what we wanted to show.

Equation (253) is the final η-style property that we must show. Again, fix some arbitrary $u \in [\![\Psi]\!]$. The composition $[\![\Psi \;;\; \Delta_1, \Delta_2 \vdash a \leftarrow \mathsf{send}\ a\ \mathsf{unfold};\ P;\ \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ Q :: c : C]\!]u$ denotes the string diagram



where the boxes respectively represent the morphisms $[\![\Psi \;;\; \Delta_1 \vdash \mathsf{send}\ a\ \mathsf{unfold};\ P :: a : \rho\alpha.A]\!]u$ and $[\![\Psi \;;\; \Delta_2, a : \rho\alpha.A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ Q :: c : C]\!]u$. Expanding eq. (188), this diagram is equal

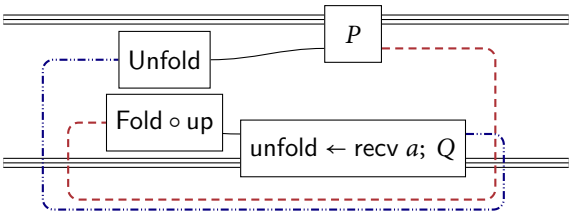
to:



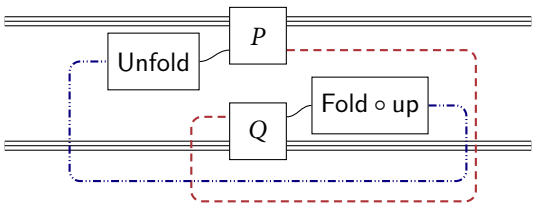
where P is the morphism $[\![\Psi\ ;\ \Delta_1 \vdash P :: a : [\rho\alpha.A/\alpha]A]\!]u$. By diagrams 2 and 185,

$$\begin{aligned}
&\langle \rho\alpha.A\rangle^+ \circ \mathsf{Fold} \circ \mathsf{up} \\
&= \mathsf{Fold} \circ (-)_\perp \langle [\rho\alpha.A/\alpha]A\rangle^+ \circ \mathsf{Unfold} \circ \mathsf{Fold} \circ \mathsf{up} \\
&= \mathsf{Fold} \circ (-)_\perp \langle [\rho\alpha.A/\alpha]A\rangle^+ \circ \mathsf{up} \\
&= \mathsf{Fold} \circ \mathsf{up} \circ \langle [\rho\alpha.A/\alpha]A\rangle^+.
\end{aligned}$$

Combining this fact with sliding, the above diagram is equal to:



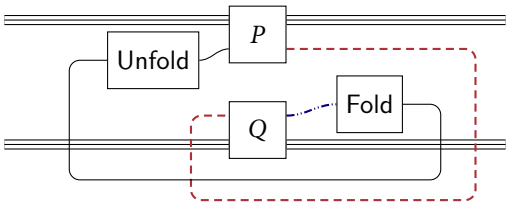
We can simplify the bottom composition by expanding eq. (189) to get:



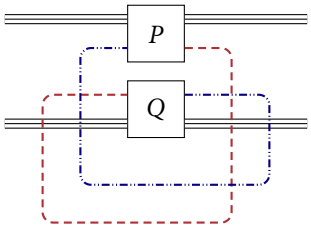
where Q is the morphism $[\![\Psi\ ;\ \Delta_2, a : [\rho\alpha.A/\alpha]A \vdash Q :: c : C]\!]u$. By diagram 185,

$$\begin{aligned}
&\langle \rho\alpha.A\rangle^- \circ \mathsf{Fold} \circ \mathsf{up} \\
&= (\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A\rangle^+ \circ \mathsf{down} \circ \mathsf{Unfold}) \circ \mathsf{Fold} \circ \mathsf{up} \\
&= \mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A\rangle^+.
\end{aligned}$$

So the diagram is equal to



By sliding the fold morphism to the left of the unfold morphism, we see that they cancel out and the diagram becomes:



This is what we wanted to show. □

Proposition 9.1.2 captures the semantic identities that arise from the commutation cases in the cut-elimination proof.

PROPOSITION 9.1.2 (Commuting Conversions). *The following semantic equivalences hold for appropriately typed processes P and Q:*

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ (\mathsf{send}\ c\ \mathsf{shift};\ Q) \equiv \mathsf{send}\ c\ \mathsf{shift};\ a \leftarrow P;\ Q :: c : {\downarrow}C \tag{256}$$

$$\Psi\ ;\ \Delta, b : {\uparrow}B \vdash a \leftarrow (\mathsf{send}\ b\ \mathsf{shift};\ P);\ Q \equiv \mathsf{send}\ b\ \mathsf{shift};\ a \leftarrow P;\ Q :: c : C \tag{257}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ (c.k;\ Q) \equiv c.k;\ a \leftarrow P;\ Q :: c : \oplus\{l : C_l\}_{l \in L} \tag{258}$$

$$\Psi\ ;\ \Delta, b : \&\{l : B_l\}_{l \in L} \vdash a \leftarrow (b.k;\ P);\ Q \equiv b.k;\ a \leftarrow P;\ Q :: c : C \tag{259}$$

$$\Psi\ ;\ \Delta, d : D \vdash a \leftarrow P;\ (\mathsf{send}\ c\ d;\ Q) \equiv \mathsf{send}\ c\ d;\ a \leftarrow P;\ Q :: c : D \otimes C \tag{260}$$

$$\Psi\ ;\ \Delta, d : D, b : D \otimes B \vdash a \leftarrow (\mathsf{send}\ b\ d;\ P);\ Q \equiv \mathsf{send}\ b\ d;\ a \leftarrow P;\ Q :: c : C \tag{261}$$

$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ (\_ \leftarrow \mathsf{output}\ c\ M;\ Q) \equiv \_ \leftarrow \mathsf{output}\ c\ M;\ a \leftarrow P;\ Q :: c : \tau \wedge C \tag{262}$$

$$\Psi\ ;\ \Delta, b : \tau \supset B \vdash a \leftarrow (\_ \leftarrow \mathsf{output}\ b\ M;\ P);\ Q \equiv \_ \leftarrow \mathsf{output}\ b\ M;\ a \leftarrow P;\ Q :: c : C \tag{263}$$

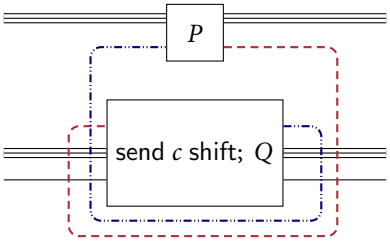
$$\Psi\ ;\ \Delta \vdash a \leftarrow P;\ (\mathsf{send}\ c\ \mathsf{unfold};\ Q) \equiv \mathsf{send}\ c\ \mathsf{unfold};\ a \leftarrow P;\ Q :: c : \rho\alpha.C \tag{264}$$

$$\Psi\ ;\ \Delta, b : \rho\beta.B \vdash a \leftarrow (\mathsf{send}\ b\ \mathsf{unfold};\ P);\ Q \equiv \mathsf{send}\ b\ \mathsf{unfold};\ a \leftarrow P;\ Q :: c : C \tag{265}$$

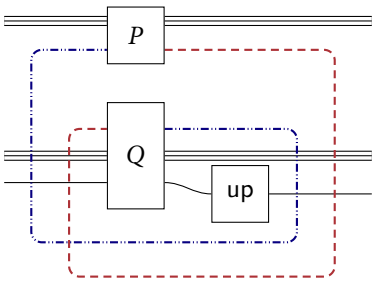
Equivalences (262) *and* (263) *are subject to the side condition that* $[\![\Psi \Vdash M : \tau]\!]u \neq \bot$ *for all* $u \in [\![\Psi]\!]$.

Proof. By string diagram manipulations.

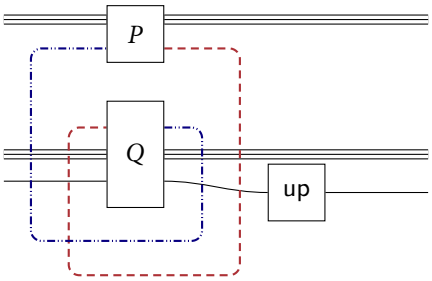
We start with eq. (256). Let $u \in [\![\Psi]\!]$ be arbitrary. The composition $[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ (\mathsf{send}\ c\ \mathsf{shift};\ Q) :: c : {\downarrow}C]\!]u$ represents the string diagram



where the morphisms are respectively $[\![\Psi\ ;\ \Delta_1 \vdash P :: a : A]\!]u$ and $[\![\Psi\ ;\ a : A, \Delta_2 \vdash Q :: c : \downarrow C]\!]u$. Expanding the definition of eq. (158), the diagram is seen to be equal to:



By tightening, this diagram is equal to:



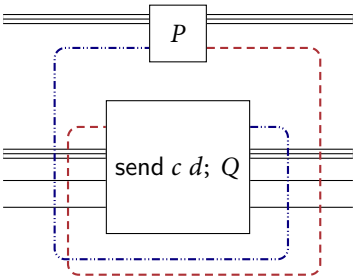
We recognize it as the diagram for $[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash \mathsf{send}\ c\ \mathsf{shift};\ a \leftarrow P;\ Q :: c : \downarrow C]\!]u$. It follows that

$$\begin{aligned}&[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ (\mathsf{send}\ c\ \mathsf{shift};\ Q) :: c : \downarrow C]\!]u\\ &= [\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash \mathsf{send}\ c\ \mathsf{shift};\ a \leftarrow P;\ Q :: c : \downarrow C]\!]u.\end{aligned}$$

But u was arbitrary, so we conclude eq. (256).

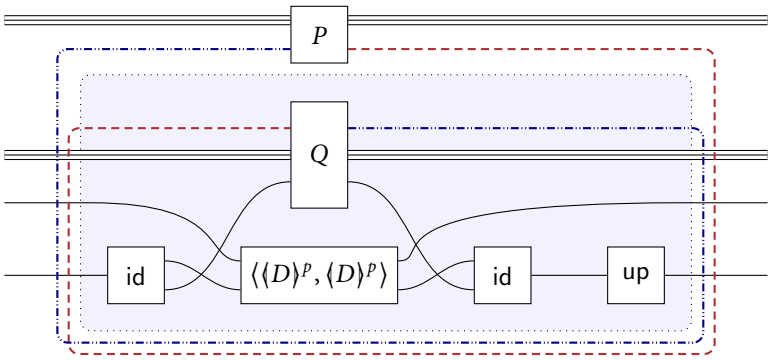
The proof of eq. (258) uses an analogous sequence of diagrams as the proof of eq. (256), except that all up morphisms are replaced by $\lambda c \in [\![C]\!].\, \iota^k([c])$. Analogously, the proof of eq. (262) replaces up morphisms by $\lambda c \in [\![C]\!].(v, [c])$ where $v = [\![\Psi \Vdash M : \tau]\!]u$.

The proof of eq. (260) is only marginally more complex. Fix some arbitrary $u \in [\![\Psi]\!]$. The composition $[\![\Psi\ ;\ \Delta_1, \Delta_2, d : D \vdash a \leftarrow P;\ \mathsf{send}\ c\ d;\ Q :: c : D \otimes C]\!]u$ denotes the string diagram

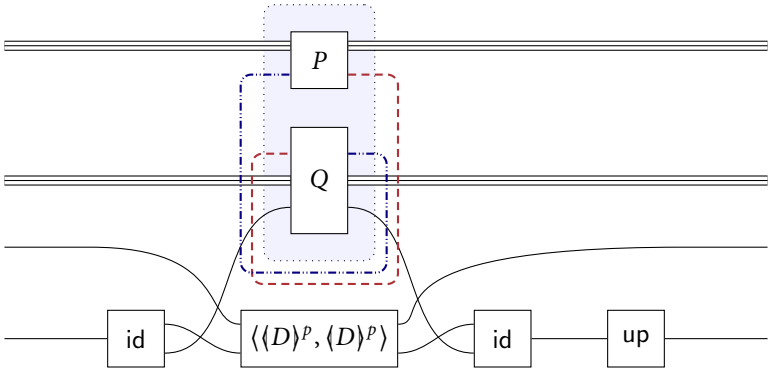


where the boxes respectively represent the morphisms $[\![\Psi\ ;\ \Delta_1 \vdash P :: a : A]\!]u$ and $[\![\Psi\ ;\ \Delta_2, d : D \vdash \mathsf{send}\ c\ d;\ Q :: c : D \otimes C]\!]u$. From top to bottom, the wires on the left and right rides of $\mathsf{send}\ c\ d;\ Q$ correspond to the channels $a : A$, Δ_2, $d : D$, and $c : D \otimes C$. We begin by expanding the definitions

of each box. By eq. (130), the diagram is equal to:

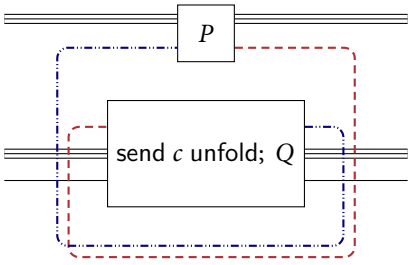


By the tightening and strength axioms, this is diagram is equal to:

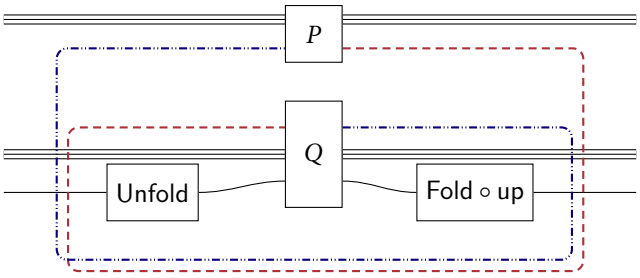


We recognize this as the diagram for $[\![\Psi\ ;\ \Delta_1, \Delta_2, d : D \vdash \mathsf{send}\ c\ d;\ a \leftarrow P;\ Q :: c : D \otimes C]\!]u$. This is what we wanted to show.

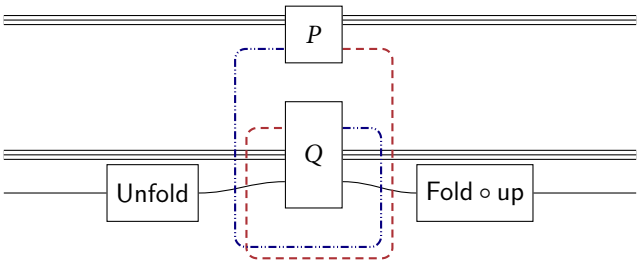
Finally, we show eq. (264). Let $u \in [\![\Psi]\!]$ be arbitrary. The composition $[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash a \leftarrow P;\ (\mathsf{send}\ c\ \mathsf{unfold};\ Q) :: c : \rho\alpha.C]\!]u$ represents the string diagram



where the morphisms are respectively $[\![\Psi\ ;\ \Delta_1 \vdash P :: a : A]\!]u$ and $[\![\Psi\ ;\ \Delta_2 \vdash \mathsf{send}\ c\ \mathsf{unfold};\ Q :: c : \rho\alpha.C]\!]u$. Expanding eq. (188), we get the diagram:



where Q is $[\![\Psi\ ;\ \Delta_2 \vdash Q :: c : [\rho\alpha.C/\alpha]C]\!]u$. By tightening, this diagram is equal to:



We recognize it as the diagram for $[\![\Psi\ ;\ \Delta_1, \Delta_2 \vdash \mathsf{send}\ c\ \mathsf{unfold};\ a \leftarrow P;\ Q :: c : \rho\alpha.C]\!]u$. This is what we wanted to show. □

9.2. Purely Polarized Session Types

We say that a session type A is **purely positive** if it is constructed using only positive types. This means that A is generated by the grammar

$$A, A_l ::= \alpha \mid \rho\alpha.A \mid \mathbf{1} \mid \oplus\{l : A_l\}_{l\in L} \mid A_1 \otimes A_2 \mid \tau \wedge A$$

where all session types are positive, i.e. that its derivation uses only the rules (CVar) (on a positive variable), (Cρ^+), (C$\mathbf{1}$), (C$\oplus$), (C$\otimes$), and (C$\wedge$). A session type A is **purely negative** if it is constructed using only negative types. This means that A is generated by the grammar

$$A, A_l ::= \alpha \mid \rho\alpha.A \mid \&\{l : A_l\}_{l\in L} \mid \tau \supset A$$

where all session types are negative, i.e., if it is generated using only the rules (CVar) (on a negative variable), (Cρ^-), (C&), and (C$\supset$). We remark that $B \multimap A$ is not purely negative because the type B must be positive. We say that a type is purely polarized if it is purely positive or purely negative.

Purely polarized types capture unidirectional communication on channels: the direction of communication never changes. This fact is reflected in our semantics: if a type is purely polarized, then it has a trivial negative aspect, and vice-versa. Recall that we write $\bot$ for the initial object.

Proposition 9.2.1. *If* $\Xi \vdash A\ \mathsf{type_s}$ *is a purely positive session type, then*

$$\langle \Xi \vdash A\ \mathsf{type_s}\rangle^+ = \mathsf{id} : [\![\Xi \vdash A\ \mathsf{type_s}]\!] \to [\![\Xi \vdash A\ \mathsf{type_s}]\!]^+$$

and $[\![\Xi \vdash A\ \mathsf{type_s}]\!]^-(\bot, \ldots, \bot) \cong \bot$. *If* $\Xi \vdash A\ \mathsf{type_s}$ *is a purely negative session type, then*

$$\langle \Xi \vdash A\ \mathsf{type_s}\rangle^- = \mathsf{id} : [\![\Xi \vdash A\ \mathsf{type_s}]\!] \to [\![\Xi \vdash A\ \mathsf{type_s}]\!]^-$$

and $[\![A]\!]^+(\bot, \ldots, \bot) \cong \bot$.

Proof. Assume first that $\Xi \vdash A\ \mathsf{type_s}$ is purely positive. We proceed by induction on the derivation, noting that every subderivation of $\Xi \vdash A\ \mathsf{type_s}$ is of a purely positive type.

Case (CVar)**:** Immediate by eqs. (168) to (171).

Case (Cρ^+): Assume that $\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+$ because $\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+$. By the induction hypothesis, $[\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!] = [\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^+$. It is then immediate by eqs. (179) and (180) that $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!] = [\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^+$. By the induction hypothesis, $\langle \Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+ \rangle^+ = \mathsf{id}$. It then follows from eq. (182) and functoriality that $\langle \Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+ \rangle^+ = \mathsf{id}$. By the induction hypothesis, $[\![\Xi, \alpha\ \mathsf{type}_s^+ \vdash A\ \mathsf{type}_s^+]\!]^-(\bot, \ldots, \bot) = \bot$. It follows that the ω-chain used to construct $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^-(\bot, \ldots, \bot)$ is constantly $\bot$ (see the remarks preceding proposition 4.2.11, and proposition 4.3.1), so its colimit is $\bot$. But this colimit is exactly $[\![\Xi \vdash \rho\alpha.A\ \mathsf{type}_s^+]\!]^-(\bot, \ldots, \bot)$, so we conclude the result.

Case (C**1**): Immediate by eqs. (116) to (119).

Case (C$\oplus$): The first part is immediate by eqs. (160), (161) and (163), the induction hypothesis, and functoriality. The second part is immediate by eq. (162), the induction hypothesis, and the fact that $\bot \times \cdots \times \bot \cong \bot$.

Case (C$\otimes$): The first part is immediate by eqs. (124) and (125), the induction hypothesis, and functoriality. The second part is immediate by eq. (126), the induction hypothesis, and the fact that $\bot \times \bot \cong \bot$.

Case (C$\wedge$): The first part is immediate by eqs. (144) and (145), the induction hypothesis, and functoriality. The second part is immediate by eq. (146) and the induction hypothesis.

The proof for purely negative types is analogous. □

We can use proposition 9.2.1 and eqs. (113) and (114) to characterize forwarding on purely polarized channels:

Corollary 9.2.2. *If A is purely positive, then*

$$[\![\cdot\,;\, a : A \vdash a \to b :: b : A]\!]\bot(a^+, \_) = (a^+, a^+).$$

If A is purely negative, then

$$[\![\cdot\,;\, a : A \vdash a \leftarrow b :: b : A]\!]\bot(\_, b^-) = (b^-, b^-).$$

9.3. Flipping Bit Streams

In example 5.3.10, we defined a process `flip` that flips bits in a bit stream. In this section, we show that flipping bits in a bit stream twice is semantically equivalent to forwarding the bit stream unchanged. Our approach involves a coinduction principle due to Pitts [Pit94], as presented by Abramsky and Jung [AJ95, § 5.4.4]. We illustrate this coinduction principle using a simpler example in section 9.3.1: we show that the positive projection of bit streams is given by the identity function.

Recall that `flip` was given by

$$\begin{aligned}\cdot\,;\, i : \mathtt{bits} \vdash o \leftarrow \{\mathsf{fix}\ f.o \leftarrow \{&\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\\ &\mathsf{send}\ o\ \mathsf{unfold};\\ &\mathsf{case}\ i\ \{\ 0 \Rightarrow o.1;\ o \leftarrow \{f\} \leftarrow i\\ &\qquad\quad \mid 1 \Rightarrow o.0;\ o \leftarrow \{f\} \leftarrow i\}\\ &\}\leftarrow i\} \leftarrow x :: o : \mathtt{bits},\end{aligned}$$

and that the bit stream protocol was specified by the session type

$$\mathtt{bits} = \rho\beta.\oplus\{0 : \beta, 1 : \beta\}.$$

Concretely, we show that[3]

$$\cdot\,;\, i : \mathtt{bits} \vdash c \leftarrow \mathtt{flip};\ \mathtt{flip} \equiv i \to o :: o : \mathtt{bits}.$$

[3]Note that we are using the fact that process judgments are closed under renaming of symbols to implicitly rename the shared channel to c in the composition $c \leftarrow \mathtt{flip};\ \mathtt{flip}$.

9.3.1. Reasoning About Bit Streams. When computing denotations of processes, it is generally useful to first determine the denotations of the session types they use: they will form the domains and codomains of the process denotations. When working with recursive types, this also involves computing the denotations of their unfoldings.

The unfolding of `bits` is the session type

$$\mathtt{BITS} = \oplus\{0 : \mathtt{bits}, 1 : \mathtt{bits}\}.$$

The types `bits` and `BITS` denote the dI-domains:

$$\begin{aligned}
[\![\mathtt{bits}]\!] &= \mathsf{FIX}\left(X \mapsto ((0 : X_\perp) \oplus (1 : X_\perp))_\perp\right) & [\![\mathtt{BITS}]\!] &= (0 : [\![\mathtt{bits}]\!]_\perp) \oplus (1 : [\![\mathtt{bits}]\!]_\perp)\\
[\![\mathtt{bits}]\!]^+ &= \mathsf{FIX}\left(X \mapsto ((0 : X_\perp) \oplus (1 : X_\perp))_\perp\right) & [\![\mathtt{BITS}]\!]^+ &= (0 : [\![\mathtt{bits}]\!]^+_\perp) \oplus (1 : [\![\mathtt{bits}]\!]^+_\perp)\\
[\![\mathtt{bits}]\!]^- &= \{\perp\} & [\![\mathtt{BITS}]\!]^- &= \{\perp\}.
\end{aligned}$$

These dI-domains are equipped with the following canonical isomorphisms:

$$\begin{aligned}
&\mathsf{Unfold} : [\![\mathtt{bits}]\!] \to [\![\mathtt{BITS}]\!]_\perp\\
&\mathsf{Unfold}^+ : [\![\mathtt{bits}]\!]^+ \to [\![\mathtt{BITS}]\!]^+_\perp\\
&\mathsf{Unfold}^- : \{\perp\} \to \{(0 : \perp, 1 : \perp)\}
\end{aligned}$$

Their respective inverses are Fold, Fold^+, and Fold^-.

Remark that $[\![\mathtt{bits}]\!] = [\![\mathtt{bits}]\!]^+$ and $[\![\mathtt{BITS}]\!] = [\![\mathtt{BITS}]\!]^+$. Theorem 2.2.53 and the remarks preceding proposition 4.2.11 explicitly characterizes their elements as infinite tuples. We use the following suggestive notation for the elements of $[\![\mathtt{bits}]\!] = [\![\mathtt{bits}]\!]^+$:

$$::\perp = \mathsf{Fold}([\perp]), \qquad 0::\alpha = \mathsf{Fold}([(0, [\alpha])]), \qquad 1::\alpha = \mathsf{Fold}([(1, [\alpha])]).$$

We show that $\langle\mathtt{bits}\rangle^+ = \mathsf{id}$. This fact is a special case of proposition 9.2.1. However, we prove it directly to illustrate the coinduction principle on dcpos. We will use this coinduction principle in more complex settings later.

Definition 9.3.1 ([Pit96, p. 70])**.** A pointed dcpo constructor $\Phi(\alpha)$ is a formal expression built up from the variable α and from constants K ranging over objects of $\mathbf{DCPO}_\perp$ using operators $(-)_\perp$ (lifting), $\times$ (product), $\otimes$ (smash product), and $\oplus$ (coalesced sum). A pointed dcpo constructor $\Phi(\alpha)$ induces a functor $\Phi : \mathbf{DCPO}_\perp \to \mathbf{DCPO}_\perp$ where the dcpo $\Phi(D)$ is obtained by replacing each occurrence of α by D and interpreting each operator as the obvious corresponding constructor. ◀

Given a dcpo D, write $D_\downarrow$ for the set $D_\downarrow = \{d \in D \mid d \neq \perp\}$ of non-bottom elements.

Definition 9.3.2 ([Pit96, p. 85])**.** Let $\Phi(\alpha)$ be a pointed dcpo constructor, and $\mathfrak{R}$ a binary relation on a pointed dcpo D. The binary relation $\Phi(\mathfrak{R})$ on $\Phi(D)$ is inductively defined on the structure of Φ as follows:

$$(d, d') \in \alpha(\mathfrak{R}) \iff (d, d') \in \mathfrak{R} \tag{266}$$

$$(d, d') \in K(\mathfrak{R}) \iff d \sqsubseteq_K d' \tag{267}$$

$$(d, d') \in \Phi_\perp(\mathfrak{R}) \iff d = [d_0] \supset \exists d'_0.d' = [d'_0] \wedge (d_0, d'_0) \in \Phi(\mathfrak{R}) \tag{268}$$

$$((d, e), (d', e')) \in (\Phi_1 \times \Phi_2)(\mathfrak{R}) \iff (d, d') \in \Phi_1(\mathfrak{R}) \wedge (e, e') \in \Phi_2(\mathfrak{R}) \tag{269}$$

$$\begin{aligned}
(u, u') \in (\Phi_1 \otimes \Phi_2)(\mathfrak{R}) \iff\ & \forall d \in (\Phi_1(D))_\downarrow.\forall e \in (\Phi_2(D))_\downarrow. \\
& u = (d, e) \supset \exists d' \in (\Phi_1(D))_\downarrow.\exists e' \in (\Phi_2(D))_\downarrow. \\
& u' = (d', e') \wedge (d, d') \in \Phi_1(\mathfrak{R}) \wedge (e, e') \in \Phi_2(\mathfrak{R})
\end{aligned} \tag{270}$$

$$\begin{aligned}
(u, u') \in (\Phi_1 \oplus \Phi_2)(\mathfrak{R}) \iff\ & (\forall d \in (\Phi_1(D))_\downarrow.u = \iota^1(d) \supset \exists d' \in (\Phi_1(D))_\downarrow. \\
& \quad u' = \iota^1(d') \wedge (d, d') \in \Phi_1(\mathfrak{R})) \\
& (\forall d \in (\Phi_2(D))_\downarrow.u = \iota^2(d) \supset \exists d' \in (\Phi_2(D))_\downarrow. \\
& \quad u' = \iota^2(d') \wedge (d, d') \in \Phi_2(\mathfrak{R}))
\end{aligned} \tag{271}$$

◀

Definition 9.3.3. Let $\Phi(\alpha)$ be a pointed dcpo constructor. A **Φ-simulation** is a binary relation $\mathfrak{R} \subseteq \mathsf{FIX}(\Phi) \times \mathsf{FIX}(\Phi)$ satisfying $(\mathsf{Unfold}(x), \mathsf{Unfold}(x')) \in \Phi(\mathfrak{R})$ for all $(x, x') \in \mathfrak{R}$. ◂

Theorem 9.3.4 ([Pit96, Corollary 6.13]). *Let $\Phi(\alpha)$ be a pointed dcpo constructor. For any $d, d' \in \mathsf{FIX}(\Phi)$, to prove $d \sqsubseteq d'$ it suffices to show $(d, d') \in \mathfrak{R}$ for some Φ-simulation $\mathfrak{R}$.*

Remark 9.3.5. Pitts's [Pit96] original results also account for mixed-variance pointed dcpo constructors. We have specialized his results to constructors $\Phi(\alpha)$ where the variable α only occurs in positive positions. This special case is sufficient for reasoning about the denotations of session types: assumption 8.1.7 implies that α only appears in positive positions in constructors for session types.

Proposition 9.3.6. *The positive projection of* `bits` *is given by the identity morphism:* $\langle\texttt{bits}\rangle^+ = \mathsf{id}$.

Proof. To show $\langle\texttt{bits}\rangle^+ = \mathsf{id}$, it is sufficient to show that $\langle\texttt{bits}\rangle^+ s = s$ for all $s \in [\![\texttt{bits}]\!]$. The constructor defining the interpretations of $[\![\texttt{bits}]\!] = [\![\texttt{bits}]\!]^+$ is

$$\Phi(\beta) = \left((0 : \beta)_\perp \oplus (1 : \beta)_\perp\right)_\perp.$$

Let $\mathfrak{R} \subseteq [\![\texttt{bits}]\!] \times [\![\texttt{bits}]\!]$ be the relation

$$\mathfrak{R} = \{(\langle\texttt{bits}\rangle^+ s, s) \mid s \in [\![\texttt{bits}]\!]\}.$$

To show our result, it is sufficient by theorem 9.3.4 to show that $\mathfrak{R}$ and $\mathfrak{R}^{\mathrm{op}}$ are Φ-simulations. Let $s \in [\![\texttt{bits}]\!]$ be arbitrary. We proceed by case analysis on s to show that

$$(\mathsf{Unfold}(\langle\texttt{bits}\rangle^+ s), \mathsf{Unfold}(s)) \in \Phi(\mathfrak{R}),$$

i.e., that $\mathfrak{R}$ is a simulation.

Case $s = \perp$**:** It follows from the fact that Unfold is an isomorphism and $\langle\texttt{bits}\rangle$ an embedding that

$$(\mathsf{Unfold}(\langle\texttt{bits}\rangle^+ s), \mathsf{Unfold}(s)) = (\perp, \perp).$$

It is immediate from (268) that $(\perp, \perp) \in \Phi(\mathfrak{R})$.

Case $s = ::\perp$**:** By diagram 185,

$$\langle\texttt{bits}\rangle^+ = \mathsf{Fold} \circ (-)_\perp \langle\texttt{BITS}\rangle^+ \circ \mathsf{Unfold}.$$

But $\langle\texttt{BITS}\rangle^+$ is strict, so $\langle\texttt{bits}\rangle^+ s = s$. It follows that

$$(\mathsf{Unfold}(\langle\texttt{bits}\rangle^+ s), \mathsf{Unfold}(s)) = ([\perp], [\perp]).$$

It is immediate from (268) and (271) that $([\perp], [\perp]) \in \Phi(\mathfrak{R})$.

Case $s = 0::s'$**:** We compute:

$$\begin{aligned}
&\langle\texttt{bits}\rangle^+ s \\
&= (\mathsf{Fold} \circ (-)_\perp \langle\texttt{BITS}\rangle^+ \circ \mathsf{Unfold})(\mathsf{Fold}([(0, [s'])])) \\
&= (\mathsf{Fold} \circ (-)_\perp \langle\texttt{BITS}\rangle^+)[(0, [s'])]) \\
&= \mathsf{Fold}([(0, [\langle\texttt{bits}\rangle^+ s'])]).
\end{aligned}$$

Then

$$(\mathsf{Unfold}(\langle\texttt{bits}\rangle^+ s), \mathsf{Unfold}(s)) = ([(0, [\langle\texttt{bits}\rangle^+ s'])], [(0, [s'])]).$$

It is immediate from the definition of $\mathfrak{R}$, (268), and (271) that

$$([(0, [\langle\texttt{bits}\rangle^+ s'])], [(0, [s'])]) \in \Phi(\mathfrak{R}).$$

Case $s = 1::s'$**:** Analogous to the previous case.

A symmetric argument will give that $\mathfrak{R}^{\mathrm{op}}$ is also a simulation. We conclude the result. □

Corollary 9.3.7. *Forwarding of bit streams is given by:*

$$[\![\cdot\,;\, i : \texttt{bits} \vdash i \to o :: o : \texttt{bits}]\!] \perp (i^+, \_) = (i^+, i^+).$$

9.3.2. Reasoning About Bit Flipping. The next step is to give a typing derivation for the process. This is because the denotations of processes are defined by induction on their typing derivation. Let τ abbreviate $\{i : \mathtt{bits} \leftarrow o : \mathtt{bits}\}$. Let B_0 be the 0 branch:

$$\dfrac{\dfrac{\dfrac{}{f : \tau \Vdash F : \tau}\ (\text{F-Var})}{f : \tau\,;\, i : \mathtt{bits} \vdash o \leftarrow \{f\} \leftarrow i :: o : \mathtt{bits}}\ (\text{E-}\{\})}{f : \tau\,;\, i : \mathtt{bits} \vdash o.1;\ o \leftarrow \{f\} \leftarrow i :: o : \mathtt{BITS}}\ (\oplus\text{R})$$

and let B_1 be the 1 branch. The typing derivation of `flip` is then

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f : \tau\,;\, i : \mathtt{bits} \vdash B_l :: o : \mathtt{BITS} \quad (\forall l \in \{0,1\})}{f : \tau\,;\, i : \mathtt{BITS} \vdash \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}} :: o : \mathtt{BITS}}\ (\oplus\text{L})}{f : \tau\,;\, i : \mathtt{BITS} \vdash \mathsf{send}\ o\ \mathsf{unfold};\ \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}} :: o : \mathtt{bits}}\ (\rho^+\text{R})}{f : \tau\,;\, i : \mathtt{bits} \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ \mathsf{send}\ o\ \mathsf{unfold};\ \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}} :: o : \mathtt{bits}}\ (\rho^+\text{L})}{f : \tau \Vdash o \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ \mathsf{send}\ o\ \mathsf{unfold};\ \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}}\} \leftarrow i : \tau}\ (\text{I-}\{\})}{\Vdash \mathsf{fix}\ f.o \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ \mathsf{send}\ o\ \mathsf{unfold};\ \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}}\} \leftarrow i : \tau}\ (\text{F-Fix})}{\cdot\,;\, i : \mathtt{bits} \vdash o \leftarrow \{\mathsf{fix}\ f.o \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ \cdots\} \leftarrow i\} \leftarrow i :: o : \mathtt{bits}}\ (\text{E-}\{\})$$

Typing derivation in hand, we can now compute the denotation of `flip` in a top-down manner. The denotation of the branch B_0 is:[4]

$$[\![f : \tau\,;\, i : \mathtt{bits} \vdash o.1;\ o \leftarrow \{f\} \leftarrow i :: o : \mathtt{BITS}]\!] f(i^+, o^-) = (I, (1, [O]))$$
$$\text{where } \mathsf{down}(f)(i^+, o^-) = (I, O).$$

The denotation of the 1 branch is analogous. The denotation of the case statement is then:

$$[\![f : \tau\,;\, i : \mathtt{BITS} \vdash \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}} :: o : \mathtt{BITS}]\!] f(i^+, o^-)$$
$$= \begin{cases} (\bot, \bot) & \text{if } i = \bot \\ ((0, [I]), (1, [O])) & \text{if } i = (0, [i']) \\ ((1, [I]), (0, [O])) & \text{if } i = (1, [i']) \end{cases}$$
$$\text{where } \mathsf{down}(f))(i', o^-) = (I, O).$$

Next, we consider the denotation of sending and receiving unfold messages:

$$[\![f : \tau\,;\, i : \mathtt{bits} \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ \mathsf{send}\ o\ \mathsf{unfold};\ \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}} :: o : \mathtt{bits}]\!] f(i^+, o^-)$$
$$= \begin{cases} (\bot, \bot) & \text{if } i^+ = \bot \\ (::\bot, ::\bot) & \text{if } i^+ = ::\bot \\ (0::I, 1::O) & \text{if } i^+ = 0::i' \\ (1::I, 0::O) & \text{if } i^+ = 1::i' \end{cases}$$
$$\text{where } \mathsf{down}(f)(i', o^-) = (I, O).$$

The four cases in the above denotation respectively correspond to:

(1) receiving nothing on i^+;
(2) receiving unfold on i^+ followed by nothing;
(3) receiving unfold on i^+, followed by a bit stream starting with 0; and
(4) receiving unfold on i^+, followed by a bit stream starting with 1.

Next, we compute the denotation of the functional term $\mathsf{fix}\ f.o \leftarrow \{\cdots\} \leftarrow i$. It is given by the least fixed point

$$[\![\Vdash \mathsf{fix}\ f.o \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ i;\ \mathsf{send}\ o\ \mathsf{unfold};\ \mathsf{case}\ i\ \{l \Rightarrow B_l\}_{l\in\{0,1\}}\} \leftarrow i : \tau]\!] \bot = \mathrm{lfp}(\Phi)$$

[4]Strictly speaking, f is an environment $u = (f : v)$. We identify this unary tuple with its single entry for convenience.

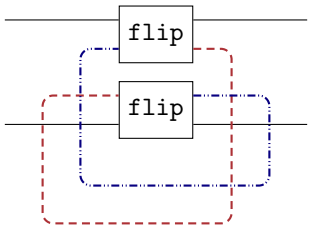
where $\Phi : [\![\tau]\!] \to [\![\tau]\!]$ is the function

$$\Phi(f) = \mathsf{up}\left(\begin{array}{l}\lambda(i^+, o^-) \in [\![\mathtt{bits}]\!]^+ \times [\![\mathtt{bits}]\!]^-. \begin{cases}(\bot, \bot) & \text{if } i^+ = \bot \\ (::\bot, ::\bot) & \text{if } i^+ = ::\bot \\ (0::I, 1::O) & \text{if } i^+ = 0::i' \\ (1::I, 0::O) & \text{if } i^+ = 1::i'\end{cases} \\ \qquad\qquad \text{where } \mathsf{down}(f)(i', o^-) = (I, O)\end{array}\right) \tag{272}$$

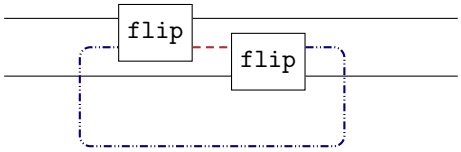
Finally, unquoting this functional term gives us the denotation of `flip`:

$$[\![\cdot \,;\, i : \mathtt{bits} \vdash \mathtt{flip} :: o : \mathtt{bits}]\!]\bot = \mathsf{down}(\mathsf{lfp}(\Phi)).$$

We turn our attention to showing that the composition of `flip` with itself is denotationally equivalent to the forwarding process. We start by computing the denotation of $c \leftarrow \mathtt{flip};\ \mathtt{flip}$. It is given by the string diagram



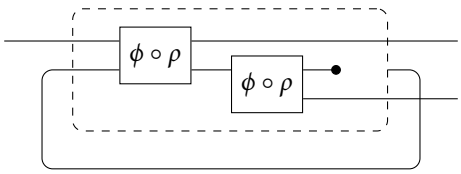
Rearranging the above string diagram, we see that the composition is given by



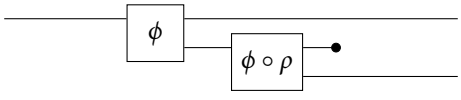
Recall that $\mathbf{Stab}_{\perp!}$ is a symmetric monoidal category with a right unit isomorphism $\rho : [\![\mathtt{bits}]\!]^+ \times [\![\mathtt{bits}]\!]^-$, and that we do not draw monoidal units in string diagrams. Let

$$\phi(i^+) = \mathsf{down}(\mathsf{lfp}(\Phi))(i^+, \bot) : [\![\mathtt{bits}]\!]^+ \to [\![\mathtt{bits}]\!] \times [\![\mathtt{bits}]\!]$$

Then $\mathtt{flip} = \phi \circ \rho$. We also recognize $\langle\mathtt{bits}\rangle^+ = \mathsf{id}$ and $\langle\mathtt{bits}\rangle^- = \bot$. We now recognize the above diagram as:



But the trace is fixing the monoidal unit, so by vanishing, the diagram is equal to:



We conclude that

$$\begin{array}{l}[\![\cdot \,;\, i : \mathtt{bits} \vdash c \leftarrow \mathtt{flip};\ \mathtt{flip} :: o : \mathtt{flip}]\!]\bot(i^+, o^-) = (I, O) \\ \text{where } \phi(i^+) = (I, O') \text{ and } \phi(O') = (\_, O).\end{array} \tag{273}$$

We can characterize ϕ using the fixed-point identity $\mathrm{lfp}(\Phi) = \Phi(\mathrm{lfp}(\Phi))$:

$$\phi(i^+) = \begin{cases} (\bot, \bot) & \text{if } i^+ = \bot \\ (::\bot, ::\bot) & \text{if } i^+ = ::\bot \\ (0::I, 1::O) & \text{if } i^+ = 0::i' \\ (1::I, 0::O) & \text{if } i^+ = 1::i' \end{cases}$$
$$\text{where } \phi(i') = (I, O).$$

Using this characterization and eq. (272), we can directly express eq. (273):

$$[\![\cdot\,;\, i : \texttt{bits} \vdash c \leftarrow \texttt{flip};\ \texttt{flip} :: o : \texttt{flip}]\!]\bot(i^+, o^-) \tag{274}$$
$$= \begin{cases} (\bot, \bot) & \text{if } i^+ = \bot \\ (::\bot, ::\bot) & \text{if } i^+ = ::\bot \\ (0::I, 0::O) & \text{if } i^+ = 0::i' \\ (1::I, 1::O) & \text{if } i^+ = 1::i' \end{cases}$$
$$\text{where } [\![\cdot\,;\, i : \texttt{bits} \vdash c \leftarrow \texttt{flip};\ \texttt{flip} :: o : \texttt{flip}]\!]\bot(i', \bot) = (I, O).$$

We are now in a position to prove our result:

Proposition 9.3.8. *Flipping bits in a bit stream twice is equivalent to forwarding it:*

$$\cdot\,;\, i : \texttt{bits} \vdash c \leftarrow \texttt{flip};\ \texttt{flip} \equiv i \to o :: o : \texttt{bits}.$$

Proof. We must show that

$$[\![\cdot\,;\, i : \texttt{bits} \vdash c \leftarrow \texttt{flip};\ \texttt{flip} :: o : \texttt{bits}]\!]\bot = [\![\cdot\,;\, i : \texttt{bits} \vdash i \to o :: o : \texttt{bits}]\!]\bot.$$

To do so, let (i^+, o^-) be arbitrary in their domain. By corollary 9.2.2, it is sufficient to show that

$$[\![\cdot\,;\, i : \texttt{bits} \vdash c \leftarrow \texttt{flip};\ \texttt{flip} :: o : \texttt{bits}]\!]\bot(i^+, o^-) = (i^+, i^+).$$

Set

$$F = [\![\cdot\,;\, i : \texttt{bits} \vdash c \leftarrow \texttt{flip};\ \texttt{flip} :: o : \texttt{bits}]\!]\bot.$$

We start by showing that if $F(i^+, o^-) = (I, \_)$, then $i^+ = I$. Let

$$\mathfrak{R} = \{(i^+, I) \mid i^+ \in [\![\texttt{bits}]\!]^+, F(i^+) = (I, \_)\}.$$

We show that $\mathfrak{R}$ and $\mathfrak{R}^{\mathrm{op}}$ are Φ-simulations by case analysis on $i^+ \in [\![\texttt{bits}]\!]^+$:

Case $i^+ = \bot$: Then $F(i^+) = (\bot, \_)$ by eq. (274). Observe that

$$(\mathsf{Unfold}(\bot), \mathsf{Unfold}(\bot)) = (\bot, \bot).$$

It is immediate from (268) that $(\bot, \bot) \in \Phi(\mathfrak{R})$ and $(\bot, \bot) \in \Phi(\mathfrak{R}^{\mathrm{op}})$.

Case $i^+ = ::\bot$: Then $F(i^+) = (::\bot, \_)$ by eq. (274). Observe that

$$(\mathsf{Unfold}(::\bot), \mathsf{Unfold}(::\bot)) = ([\bot], [\bot]).$$

It is immediate from (268) and (271) that $([\bot], [\bot]) \in \Phi(\mathfrak{R})$ and $([\bot], [\bot]) \in \Phi(\mathfrak{R}^{\mathrm{op}})$.

Case $i^+ = 0::i'$: Then by eq. (274), $F(i^+) = (0::I, \_$ where $F(i') = (I, \_)$. Observe that

$$(\mathsf{Unfold}(0::i'), \mathsf{Unfold}(0::I)) = ([(0, [i'])], [(0, [I])]).$$

It is immediate from the definition of $\mathfrak{R}$, (268), and (271) that

$$([(0, [i'])], [(0, [I])]) \in \Phi(\mathfrak{R}) \cap \Phi(\mathfrak{R}^{\mathrm{op}}).$$

Case $i^+ = 1::i'$: Analogous to the previous case.

We conclude that $\mathfrak{R}$ and $\mathfrak{R}^{\mathrm{op}}$ are Φ-simulations. We conclude by theorem 9.3.4 that for all $(i^+ \in [\![\mathtt{bits}]\!]^+$, if $F(i^+, o^-) = (I, \_)$, then $i^+ = I$.

An identical argument shows that the relations

$$\mathfrak{S} = \{(i^+, O) \mid i^+ \in [\![\mathtt{bits}]\!]^+, F(i^+) = (\_, O)\}$$

and $\mathfrak{S}^{\mathrm{op}}$ are Φ-simulations. We conclude by theorem 9.3.4 that for all $(i^+ \in [\![\mathtt{bits}]\!]^+$, if $F(i^+, o^-) = (\_, O)$, then $i^+ = O$.

Combining these results, we deduce that $F(i^+, o^-) = (i^+, i^+)$ for all (i^+, o^-). □

9.4. Identity Expansion

Identity expansion theorems for sequent calculi state that if a sequent is provable, then we can give it a proof that only uses the identity rule on atomic propositions. Computationally, this corresponds to reducing channel forwarding at complex types to forwarding at simpler types [CPT12, p. 3].

Lemma 9.4.1. *If* $A\ \mathsf{type}_s^+$, *then* $\langle A \rangle^p(\bot, \_) = \bot$. *If* $A\ \mathsf{type}_s^-$, *then* $\langle A \rangle^p(\_, \bot) = \bot$.

Proof. By case analysis on the last rule of the derivation of $\cdot \vdash A\ \mathsf{type}_s$.

Case (C**1**)**:** Immediate by eq. (121).

Case (Cρ^+)**:** Recall eq. (187). By proposition 2.2.21 and the definition of δ,

$$\begin{aligned}
&\langle \rho\alpha.A\ \mathsf{type}_s^+ \rangle^p(\bot, _) \\
&= (\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A \rangle^p \circ \delta \circ (\mathsf{Unfold} \times \mathsf{Unfold}))\,(\bot, _) \\
&= (\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A \rangle^p \circ \delta)\,(\bot, _) \\
&= (\mathsf{Fold} \circ \langle [\rho\alpha.A/\alpha]A \rangle^p))\bot \\
&= \bot.
\end{aligned}$$

Case (C↓)**:** Recall eq. (157). By definition of δ and the action of the lifting functor,

$$\begin{aligned}
&\langle \downarrow A\ \mathsf{type}_s^+ \rangle^p(\bot, _) \\
&= ((-)_\bot \langle \Xi \vdash A\ \mathsf{type}_s^- \rangle^p \cdot \delta)\,(\bot, _) \\
&= ((-)_\bot \langle \Xi \vdash A\ \mathsf{type}_s^- \rangle^p)\,(\bot, \bot) \\
&= \bot.
\end{aligned}$$

Case (C⊕)**:** Recall eq. (165). It is by definition strict in the positive component.

Case (C∧)**:** Recall eq. (149). It is by definition strict in the positive component.

The remaining cases follow by analogy with those above. □

A session type $\Xi \vdash A\ \mathsf{type}_s$ is *morally recursion-free* if whenever (Cρ^+) or (Cρ^-) appears in its derivation with conclusion $\Xi' \vdash \rho\beta.B\ \mathsf{type}_s$, then β does not appear free in B. We prove identity expansion for the morally recursion-free fragment of Polarized SILL. This fragment contains the logical fragment Polarized SILL, i.e., all session types that correspond to propositions in intuitionistic linear logic.

Theorem 9.4.2. *For all morally recursion-free closed session types* $A\ \mathsf{type}_s^+$, *there exists a cut-free process* I_A *whose derivation does not use (*Fwd$^+$*), (*Fwd$^-$*), or (*E-{}*) such that*

$$\cdot\,;\, a : A \vdash I_A \equiv a \to b :: b : A.$$

For all morally recursion-free closed session types $A\ \mathsf{type}_s^-$, *there exists a cut-free process* I_A *whose derivation does not use (*Fwd$^+$*), (*Fwd$^-$*), or (*E-{}*) such that*

$$\cdot\,;\, a : A \vdash I_A \equiv a \leftarrow b :: b : A.$$

We do not directly prove theorem 9.4.2: it will be an immediate corollary of theorem 9.4.3. We conjecture that we can extend theorem 9.4.2 to support general recursive types. Though a complete proof remains elusive, we present a compelling proof sketch. Then, we then discuss the difficulties involved in completing this proof skeleton.

At a high level, we believe that the expanded forwarding process I_A should be defined by induction on the derivation of A, and that forwarding for $\rho\alpha.A$ should be captured by a recursive process. To define I_A by induction on the derivation of A, we must account for open session types. Indeed, in the case of recursive session types, the rule hypothesis is an open session type, so our induction hypothesis cannot be restricted to closed session types. Here arises the first difficulty: processes cannot communicate over open session types. Put differently, we cannot define a forwarding $\Psi\ ;\ a : A \vdash a \leftarrow b :: b : A$ for an open session type $\Xi \vdash A\ \mathsf{type}_s$.

To address this, we show that for all open session types $\Xi \vdash A\ \mathsf{type}_s$ and closing substitutions $\sigma :_s \cdot \rightsquigarrow \Xi$, there exists an open identity expansion process $I_{[\sigma]A}$ that is (almost) denotationally equivalent to the forwarding process for $[\sigma]A$. In the case of $\Xi \vdash \rho\alpha.A\ \mathsf{type}_s$, we can apply the induction hypothesis for $\Xi, \alpha \vdash A\ \mathsf{type}_s$ to the substitution $(\sigma, \rho\alpha.[\sigma]A) :_s \cdot \rightsquigarrow \Xi, \alpha$ to get an identity expansion process $I_{[\rho\alpha.[\sigma]A/\alpha]([\sigma]A)}$. Wrapping this process in the appropriate fold and unfold process constructors is still insufficient to define the identity expansion process $I_{[\sigma]\rho\alpha.A}$. Indeed, after enough computation, the process $I_{[\rho\alpha.[\sigma]A/\alpha]([\sigma]A)}$ may have to forward communications of type $\rho\alpha.([\sigma]A)$. We escape this circularity by defining $I_{[\sigma](\rho\alpha.A)}$ as a recursive process in terms of $I_{[\rho\alpha.[\sigma]A/\alpha]([\sigma]A)}$. It is defined such every time $I_{[\rho\alpha.[\sigma]A/\alpha]([\sigma]A)}$ has to forward communications of type $[\sigma](\rho\alpha.A)$, it makes a recursive call to $I_{[\sigma](\rho\alpha.A)}$.

We make these vague intuitions clear by explicitly constructing the processes $I_{[\sigma]A}$. Given $\Xi = \alpha_1, \ldots, \alpha_n$ with $n \geq 0$, the context Ψ_σ of functional variables is given by

$$\Psi_\sigma = \hat{\alpha}_1 : \{a : \sigma(\alpha_1) \leftarrow b : \sigma(\alpha_1)\}, \ldots, \hat{\alpha}_n : \{a : \sigma(\alpha_n) \leftarrow b : \sigma(\alpha_n)\}.$$

Let $u_\sigma \in [\![\Psi_\sigma]\!]$ be the environment that maps the functional variable $\hat{\alpha}_i$ to the quoted forwarding process for the type $\sigma(\alpha_i)$. Explicitly, it is the environment such that

$$u_\sigma(\hat{\alpha}_i) = [\![\cdot \Vdash b \leftarrow \{a \to b\} \leftarrow a : \{a : \sigma(\alpha_i) \leftarrow b : \sigma(\alpha_i)\}]\!]\bot$$

whenever α_i is positive, and

$$u_\sigma(\hat{\alpha}_i) = [\![\cdot \Vdash b \leftarrow \{a \leftarrow b\} \leftarrow a : \{a : \sigma(\alpha_i) \leftarrow b : \sigma(\alpha_i)\}]\!]\bot$$

otherwise.

Theorem 9.4.3 (Identity Expansion). *For all session types $\Xi \vdash A\ \mathsf{type}_s$ and closing substitutions $\sigma :_s \cdot \rightsquigarrow \Xi$, there exists a cut-free process $\Psi_\sigma\ ;\ a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A$ whose derivation does not use (Fwd$^+$) or (Fwd$^-$). If A is positive and morally recursion-free, then*

$$[\![\Psi_\sigma\ ;\ a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A]\!]u_\sigma = [\![\cdot\ ;\ a : [\sigma]A \vdash a \to b :: b : [\sigma]A]\!]\bot,$$

and if A is negative and morally recursion-free, then

$$[\![\Psi_\sigma\ ;\ a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A]\!]u_\sigma = [\![\cdot\ ;\ a : [\sigma]A \vdash a \to b :: b : [\sigma]A]\!]\bot.$$

If the derivation of $\cdot \vdash [\sigma]A\ \mathsf{type}_s$ does not use (CVar), then $I_{[\sigma]A}$ does not use (E-{}).

Proof. By well-founded induction on the set of derivations of session-types, ordered by the smallest strict preorder $\prec$ generated by:

- $\mathcal{D}_1 \prec \mathcal{D}_2$ if $\mathcal{D}_1$ is a subderivation[5] of $\mathcal{D}_2$; and
- $\mathcal{D}_1 \prec \mathcal{D}_2$ if $\mathcal{D}_2$ is obtained by weakening $\mathcal{D}_1$.

We proceed by case analysis on the last rule used to form $\Xi \vdash A\ \mathsf{type}_s$. We show the positive cases, and the negative cases will follow by analogy.

[5]That is, a subtree.

Case (CVar): Then conclusion is $\Xi \vdash \alpha_i\ \mathsf{type_s}$ for some $\alpha_i \in \Xi$. By definition, $[\sigma]\alpha_i = \sigma(\alpha_i)$. We show the case for positive α_i; the negative case follows by symmetry. Let $I_{[\sigma]\alpha_i}$ given by

$$\dfrac{\dfrac{}{\Psi_\sigma \Vdash \hat{\alpha}_i : \{a : \sigma(\alpha_i) \leftarrow b : \sigma(\alpha_i)\}}\ (\text{F-Var})}{\Psi_\sigma\ ;\ a : \sigma(\alpha_i) \vdash b \leftarrow \{\hat{\alpha}_i\} \leftarrow a :: b : \sigma(\alpha_i)}\ (\text{E-}\{\})$$

By construction of u_σ:

$$\begin{aligned}
&[\![\Psi_\sigma\ ;\ a : \sigma(\alpha_i) \vdash b \leftarrow \{\hat{\alpha}_i\} \leftarrow a :: b : \sigma(\alpha_i)]\!]u_\sigma \\
&= \mathsf{down} \circ [\![\Psi_\sigma \Vdash \hat{\alpha}_i : \{a : \sigma(\alpha_i) \leftarrow b : \sigma(\alpha_i)\}]\!]u_\sigma \\
&= \mathsf{down}(u_\sigma(\hat{\alpha}_i)) \\
&= \mathsf{down}(\mathsf{up}([\![\cdot \Vdash b \leftarrow \{a \to b\} \leftarrow a : \{a : \sigma(\alpha_i) \leftarrow b : \sigma(\alpha_i)\}]\!]\bot)) \\
&= [\![\cdot \Vdash b \leftarrow \{a \to b\} \leftarrow a : \{a : \sigma(\alpha_i) \leftarrow b : \sigma(\alpha_i)\}]\!]\bot.
\end{aligned}$$

This is what we wanted to show.

Case (C$\mathbf{1}$): Observe that $[\sigma]\mathbf{1} = \mathbf{1}$. Let $I_{[\sigma]\mathbf{1}}$ be given by

$$\dfrac{\dfrac{}{\Psi_\sigma\ ;\ \vdash \mathsf{close}\ b :: b : \mathbf{1}}\ (\mathbf{1}\text{R})}{\Psi_\sigma\ ;\ a : \mathbf{1} \vdash \mathsf{wait}\ a;\ \mathsf{close}\ b :: b : \mathbf{1}}\ (\mathbf{1}\text{L})$$

Recall eqs. (113) and (121) to (123). We must show that

$$[\![\Psi_\sigma\ ;\ a : \mathbf{1} \vdash I_{[\sigma]\mathbf{1}} :: b : \mathbf{1}]\!]u_\sigma = [\![\cdot\ ;\ a : \mathbf{1} \vdash a \to b :: b : \mathbf{1}]\!]\bot$$

are equal functions. It is sufficient to proceed by case analysis on the elements in their domain. There are two possibilities: $(\bot, \bot)$ or $(\mathsf{close}, \bot)$. In the first case,

$$\begin{aligned}
&[\![\Psi_\sigma\ ;\ a : \mathbf{1} \vdash \mathsf{wait}\ a;\ \mathsf{close}\ b :: b : \mathbf{1}]\!]u_\sigma(\bot, \bot) \\
&= (\bot, \bot) \\
&= [\![\cdot\ ;\ a : \mathbf{1} \vdash a \to b :: b : \mathbf{1}]\!]\bot(\bot, \bot).
\end{aligned}$$

In the second case,

$$\begin{aligned}
&[\![\Psi_\sigma\ ;\ a : \mathbf{1} \vdash \mathsf{wait}\ a;\ \mathsf{close}\ b :: b : \mathbf{1}]\!]u_\sigma(\mathsf{close}, \bot) \\
&= (\mathsf{close}, \mathsf{close}) \\
&= [\![\cdot\ ;\ a : \mathbf{1} \vdash a \to b :: b : \mathbf{1}]\!]\bot(\mathsf{close}, \bot).
\end{aligned}$$

This gives the result.

Case (C$\otimes$): Assume that $\Xi \vdash A \otimes B\ \mathsf{type}_\mathsf{s}^+$ because $\Xi \vdash A\ \mathsf{type}_\mathsf{s}^+$ and $\Xi \vdash B\ \mathsf{type}_\mathsf{s}^+$. By the induction hypothesis, there is a process $\Psi_\sigma\ ;\ a : [\sigma]B \vdash I_{[\sigma]B} :: b : [\sigma]B$ satisfying the theorem statement. Observe that $[\sigma](B \otimes A) = ([\sigma]B) \otimes ([\sigma]A)$. Let $I_{[\sigma](A \otimes B)}$ be given by

$$\dfrac{\dfrac{\Psi_\sigma\ ;\ a : [\sigma]A \vdash I_{[\sigma]B} :: b : [\sigma]A}{\Psi_\sigma\ ;\ a : [\sigma]A, c : [\sigma]B \vdash \mathsf{send}\ b\ c;\ I_{[\sigma]B} :: b : [\sigma](B \otimes A)}\ (\otimes\text{R})}{\Psi_\sigma\ ;\ a : [\sigma](B \otimes A) \vdash c \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ c;\ I_{[\sigma]B} :: b : [\sigma](B \otimes A)}\ (\otimes\text{L})$$

Recall eqs. (113) and (129) to (131). Assume that $B \otimes A$ is morally recursion-free. We show that the denotation of $I_{[\sigma](B \otimes A)}$ is suitably equivalent to $a \to b$. As in previous cases, we proceed by case analysis on an arbitrary element in their domain:

Subcase $(\bot, (b^-, a^-))$: We compute:

$$\begin{aligned}
&[\![\Psi_\sigma\ ;\ a : [\sigma](B \otimes A) \vdash c \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ c;\ I_{[\sigma]B} :: b : [\sigma](B \otimes A)]\!]u(\bot, (b^-, a^-)) \\
&= (\bot, \langle [\sigma](B \otimes A) \rangle^p (\bot, (b^-, a^-)))
\end{aligned}$$

which by lemma 9.4.1:

$$= (\bot, \bot)$$
$$= (\langle [\sigma](B \otimes A)\rangle^p(\bot, (b^-, a^-)), \langle [\sigma](B \otimes A)\rangle^p(\bot, (b^-, a^-)))$$
$$= [\![\cdot\,;\, a : [\sigma](B \otimes A) \vdash a \to b :: b : [\sigma](B \otimes A)]\!]\bot(\bot, (b^-, a^-)).$$

Subcase $([(b^+, a^+)], (b^-, a^-))$: Set

$$(\beta, \beta) = [\![\cdot\,;\, a : [\sigma]A \vdash a \to b :: b : [\sigma]A]\!]\bot(b^+, b^-).$$

By the induction hypothesis,

$$[\![\Psi_\sigma\,;\, a : [\sigma]A \vdash I_{[\sigma]B} :: b : [\sigma]A]\!]u_\sigma(b^+, b^-) = (\beta, \beta).$$

It follows that

$$[\![\Psi_\sigma\,;\, a : [\sigma]A, c : [\sigma]B \vdash \mathsf{send}\ b\ c;\ I_{[\sigma]B} :: b : [\sigma](B \otimes A)]\!]u_\sigma(b^+, a^+, (b^-, a^-))$$
$$= (\beta, \alpha, [(\beta, \alpha)])$$

where $\alpha = \langle A\rangle^p(a^+, a^-)$. Using this, we compute that

$$[\![\Psi_\sigma\,;\, a : [\sigma](B \otimes A) \vdash c \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ c;\ I_{[\sigma]B} :: b : [\sigma](B \otimes A)]\!]u_\sigma([(b^+, a^+)], (b^-, a^-))$$
$$= ([(\beta, \alpha)], [(\beta, \alpha)]).$$

But

$$[\![\cdot\,;\, a : [\sigma](B \otimes A) \vdash a \to b :: b : [\sigma](B \otimes A)]\!]\bot([(b^+, a^+)], (b^-, a^-)) = ([(\beta, \alpha)], [(\beta, \alpha)]),$$

so we conclude the result.

Case (C⊕): Assume that $\Xi \vdash \oplus\{l : A_l\}_{l \in L}\ \mathsf{type_s}$ because $\Xi \vdash A_l\ \mathsf{type_s}$ for $l \in L$. By the induction hypothesis, there exist processes $\Psi_\sigma\ ;\ a : [\sigma]A_l \vdash I_{[\sigma]A_l} :: b : [\sigma]A_l$ satisfying the theorem statement for $l \in L$. Observe that $[\sigma](\oplus\{l : A_l\}_{l \in L}) = \oplus\{l : [\sigma]A_l\}_{l \in L}$. Let $I_{[\sigma](\oplus\{l:A_l\}_{l\in L})}$ be given by

$$\dfrac{\dfrac{\Psi_\sigma\ ;\ a : [\sigma]A_l \vdash I_{[\sigma]A_l} :: b : [\sigma]A_l}{\Psi_\sigma\ ;\ a : [\sigma]A_l \vdash b.l;\ I_{[\sigma]A_l} :: b : [\sigma](\oplus\{l : A_l\}_{l \in L})}\ (\oplus\mathrm{R}) \quad (l \in L)}{\Psi_\sigma\ ;\ a : [\sigma](\oplus\{l : A_l\}_{l \in L}) \vdash \mathsf{case}\ a\ \{b.l;\ I_{[\sigma]A_l}\}_{l \in L} :: b : [\sigma](\oplus\{l : A_l\}_{l \in L})}\ (\oplus\mathrm{L})$$

Recall eqs. (113) and (165) to (167). Assume that $\Xi \vdash \oplus\{l : A_l\}_{l \in L}\ \mathsf{type_s}$ is morally recursion-free. We show that $I_{[\sigma](\oplus\{l:A_l\}_{l\in L})}$ is suitably equivalent to $a \to b$. As in previous cases, we proceed by case analysis on an arbitrary element in their domain:

Subcase $\left(\bot, \left(a_l^-\right)_{l \in L}\right)$: We compute:

$$[\![\Psi_\sigma\ ;\ a : [\sigma](\oplus\{l : A_l\}_{l \in L}) \vdash \mathsf{case}\ a\ \{b.l;\ I_{[\sigma]A_l}\}_{l \in L} :: b : [\sigma](\oplus\{l : A_l\}_{l \in L})]\!]u_\sigma\left(\bot, \left(a_l^-\right)_{l \in L}\right)$$
$$= \left(\bot, \langle [\sigma](\oplus\{l : A_l\}_{l \in L})\rangle^p\left(\bot, \left(a_l^-\right)_{l \in L}\right)\right)$$

which by lemma 9.4.1:

$$= (\bot, \bot)$$
$$= \left(\langle [\sigma](\oplus\{l : A_l\}_{l \in L})\rangle^p\left(\bot, \left(a_l^-\right)_{l \in L}\right), \langle [\sigma](\oplus\{l : A_l\}_{l \in L})\rangle^p\left(\bot, \left(a_l^-\right)_{l \in L}\right)\right)$$
$$= [\![\cdot\,;\, a : [\sigma](\oplus\{l : A_l\}_{l \in L}) \vdash a \to b :: b : [\sigma](\oplus\{l : A_l\}_{l \in L})]\!]\bot\left(\bot, \left(a_l^-\right)_{l \in L}\right).$$

Subcase $\left(\left(k, [a_k^+]\right), \left(a_l^-\right)_{l \in L}\right)$: Set

$$(\beta, \beta) = [\![\cdot\,;\, a : [\sigma]A_k \vdash a \to b :: b : [\sigma]A_k]\!]\bot(a_k^+, a_k^-).$$

By the induction hypothesis,

$$[\![\Psi_\sigma\ ;\ a : [\sigma]A_k \vdash I_{[\sigma]A_k} :: b : [\sigma]A_k]\!]u_\sigma(a_k^+, a_k^-) = (\beta, \beta).$$

It follows from eq. (166) that

$$[\![\Psi_\sigma\ ;\ a : [\sigma]A_l \vdash b.l;\ I_{[\sigma]A_l} :: b : [\sigma](\oplus\{l : A_l\}_{l \in L})]\!]u_\sigma\left(a_k^+, \left(a_l^-\right)_{l \in L}\right) = (\beta, (l, [\beta]))$$

for all $l \in L$. Then by eq. (167),

$$\begin{aligned}&[\![\Psi_\sigma \,;\, a : [\sigma](\oplus\{l : A_l\}_{l\in L}) \vdash \\ &\qquad \mathsf{case}\ a\ \{b.l;\ I_{[\sigma]A_l}\}_{l\in L} :: b : [\sigma](\oplus\{l : A_l\}_{l\in L})]\!]u_\sigma\left((k, [a_k^+]), (a_l^-)_{l\in L}\right) \\ &= ((l, [\beta]), (l, [\beta]))\,.\end{aligned}$$

But

$$\begin{aligned}&[\![\cdot \,;\, a : [\sigma](\oplus\{l : A_l\}_{l\in L}) \vdash a \to b :: b : [\sigma](\oplus\{l : A_l\}_{l\in L})]\!]u_\sigma\left((k, [a_k^+]), (a_l^-)_{l\in L}\right) \\ &= ((l, [\beta]), (l, [\beta]))\end{aligned}$$

by eqs. (113) and (165), so we conclude the result.

Case (C↓): Assume that $\Xi \vdash {\downarrow}A\ \mathsf{type_s}$ because $\Xi \vdash A\ \mathsf{type_s}$. By the induction hypothesis, there is a process $\Psi_\sigma \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A$ satisfying the theorem statement. Observe that $[\sigma](\downarrow A) = \downarrow[\sigma]A$. Let $I_{[\sigma](\downarrow A)}$ be given by

$$\dfrac{\dfrac{\Psi_\sigma \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A}{\Psi_\sigma \,;\, a : [\sigma]A \vdash \mathsf{send}\ b\ \mathsf{shift};\ I_{[\sigma]A} :: b : [\sigma](\downarrow A)}\ (\downarrow\mathrm{R})}{\Psi_\sigma \,;\, a : [\sigma](\downarrow A) \vdash \mathsf{shift} \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ \mathsf{shift};\ I_{[\sigma]A} :: b : [\sigma](\downarrow A)}\ (\downarrow\mathrm{L})$$

Recall eqs. (113) and (157) to (159). Assume that $\Xi \vdash {\downarrow}A\ \mathsf{type_s}$ is morally recursion-free. We show that $I_{[\sigma](\downarrow A)}$ is suitably equivalent to $a \to b$. A case analysis analogous to the case analysis for case (C⊕) gives the result.

Case (C∧): Assume that $\Xi \vdash \tau \wedge A\ \mathsf{type_s}$ because $\Xi \vdash A\ \mathsf{type_s}$ and $\tau\ \mathsf{type_f}$. By the induction hypothesis, there is a process $\Psi_\sigma \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A$ satisfying the theorem statement. We can weaken it to a process $\Psi_\sigma, x : \tau \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A$, where we assume without loss of generality that the variable x is fresh. Observe that $[\sigma](\tau \wedge A) = \tau \wedge [\sigma]A$. Let $I_{[\sigma](\tau\wedge A)}$ be given by

$$\dfrac{\dfrac{\Psi_\sigma, x : \tau \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A}{\Psi_\sigma, x : \tau \,;\, a : [\sigma]A \vdash \_ \leftarrow \mathsf{output}\ b\ x;\ I_{[\sigma]A} :: b : [\sigma](\tau \wedge A)}\ (\wedge\mathrm{R})}{\Psi_\sigma \,;\, a : [\sigma](\tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ \_ \leftarrow \mathsf{output}\ b\ x;\ I_{[\sigma]A} :: b : [\sigma](\tau \wedge A)}\ (\wedge\mathrm{L})$$

Recall eqs. (113) and (149) to (151). Assume that $\Xi \vdash \tau \wedge A\ \mathsf{type_s}$ is morally recursion-free. We show that $I_{[\sigma](\tau\wedge A)}$ is suitably equivalent to $a \to b$. As in previous cases, we proceed by case analysis on elements of their domains.

Subcase $(\bot, a^-)$: We compute:

$$\begin{aligned}&[\![\Psi_\sigma \,;\, a : [\sigma](\tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ \_ \leftarrow \mathsf{output}\ b\ x;\ I_{[\sigma]A} :: b : [\sigma](\tau \wedge A)]\!]u_\sigma(\bot, a^-) \\ &= (\bot, \langle\!|[\sigma](\tau \wedge A)|\!\rangle^p(\bot, a^-))\end{aligned}$$

which by lemma 9.4.1:

$$\begin{aligned}&= (\bot, \bot) \\ &= (\langle\!|[\sigma](\tau \wedge A)|\!\rangle^p(\bot, a^-), \langle\!|[\sigma](\tau \wedge A)|\!\rangle^p(\bot, a^-)) \\ &= [\![\cdot \,;\, a : [\sigma](\tau \wedge A) \vdash a \to b :: b : [\sigma](\tau \wedge A)]\!]\bot\,(\bot, a^-)\end{aligned}$$

Subcase $((v, [a^+]), a^-)$: Set

$$(\beta, \beta) = [\![\cdot \,;\, a : [\sigma]A \vdash a \to b :: b : [\sigma]A]\!]\bot(a^+, a^-).$$

By the induction hypothesis,

$$[\![\Psi_\sigma \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A]\!]u_\sigma(a^+, a^-) = (\beta, \beta).$$

By coherence (proposition 8.5.4),

$$[\![\Psi_\sigma, x : \tau \,;\, a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A]\!][u_\sigma \mid x \mapsto v](a^+, a^-) = (\beta, \beta).$$

It follows from eq. (150) that

$$\begin{aligned}&[\![\Psi_\sigma, x : \tau\,;\, a : [\sigma]A \vdash \_ \leftarrow \mathsf{output}\ b\ x;\ I_{[\sigma]A} :: b : [\sigma](\tau \wedge A)]\!][u_\sigma \mid x \mapsto v](a^+, a^-)\\ &= (\beta, (v, [\beta])).\end{aligned}$$

Then by eq. (151),

$$\begin{aligned}&[\![\Psi_\sigma\,;\, a : [\sigma](\tau \wedge A \vdash x \leftarrow \mathsf{input}\ a;\ \_ \leftarrow \mathsf{output}\ b\ x;\ I_{[\sigma]A} :: b : [\sigma](\tau \wedge A)]\!]u_\sigma\ ((v, [a^+])\,, a^-)\\ &= ((v, [\beta])\,, (v, [\beta]))\,.\end{aligned}$$

But

$$[\![\cdot\,;\, a : [\sigma](\tau \wedge A) \vdash a \to b :: b : [\sigma](\tau \wedge A)]\!]_\perp\ ((v, [a^+])\,, a^-) = ((v, [\beta])\,, (v, [\beta]))$$

by eqs. (113) and (149), so we conclude the result.

Case (Cρ^+)**:** Then the conclusion is $\Xi \vdash \rho\alpha.A\ \mathsf{type_s}$. We consider two subcases: when α appears free in A, and when α is does not appear free in A. This case analysis ensures that if the derivation of $\cdot \vdash [\sigma]A\ \mathsf{type_s}$ does not use (CVar), then $I_{[\sigma]A}$ does not use (E-{}).

In the first case, observe that $[\sigma](\rho\alpha.A) = \rho\alpha.[\sigma]A$. Let $\eta = \sigma, \rho\alpha.[\sigma]A :_\mathsf{s} \cdot \rightsquigarrow \Xi, \alpha\ \mathsf{type_s^+}$ be the context morphism that extends σ to substitute $\rho\alpha.[\sigma]A$ for α, so that

$$[\eta]A = [\sigma(\alpha_1), \ldots, \sigma(\alpha_n), \rho\alpha.[\sigma]A/\alpha_1, \ldots, \alpha_n, \alpha]A = [\rho\alpha.[\sigma]A/\alpha]([\sigma]A).$$

By the induction hypothesis, there is a process $\Psi_\eta\ ;\ a : [\eta]A \vdash I_{[\eta]A} :: b : [\eta]A$ satisfying the theorem statement. Let the process

$$I_{[\sigma](\rho\alpha.A)} = b \leftarrow \{\mathsf{fix}\ \hat{\alpha}.b \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\eta]A}\} \leftarrow a\} \leftarrow a$$

be given by:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Psi_\eta\ ;\ a : [\eta]A \vdash I_{[\eta]A} :: b : [\eta]A}{\Psi_\eta\ ;\ a : [\eta]A \vdash \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\eta]A} :: b : \rho\alpha.[\sigma]A}\ (\rho^+\mathrm{R})}{\Psi_\eta\ ;\ a : \rho\alpha.[\sigma]A \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\eta]A} :: b : \rho\alpha.[\sigma]A}\ (\rho^+\mathrm{L})}{\Psi_\eta \Vdash b \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \cdots\} \leftarrow a : \{b : \rho\alpha.[\sigma]A \leftarrow b : \rho\alpha.[\sigma]A\}}\ (\text{I-}\{\})}{\Psi_\sigma \Vdash \mathsf{fix}\ \hat{\alpha}.b \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \cdots\} \leftarrow a : \{b : \rho\alpha.[\sigma]A \leftarrow b : \rho\alpha.[\sigma]A\}}\ (\text{F-Fix})}{\Psi_\sigma\ ;\ a : \rho\alpha.[\sigma]A \vdash b \leftarrow \{\mathsf{fix}\ \hat{\alpha}.b \leftarrow \{\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \cdots\} \leftarrow a\} \leftarrow a :: b : \rho\alpha.[\sigma]A}\ (\text{E-}\{\})$$

At this point, we have nothing left to show: $\rho\alpha.A$ is not morally recursion-free. Nevertheless, we conjecture that $I_{[\sigma](\rho\alpha.A)}$ is suitably equivalent to forwarding $a \to b$. We suspect that a coinduction-style argument could be applied to show this equivalence, but such an argument unfortunately remains elusive.

Assume next that α does not appear free in A. Then the hypothesis $\Xi, \alpha \vdash A\ \mathsf{type_s}$ used to form $\Xi \vdash \rho\alpha.A\ \mathsf{type_s}$ can obtained by weakening $\Xi \vdash A\ \mathsf{type_s}$. By the induction hypothesis, there is a process $\Psi_\sigma\ ;\ a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A$ satisfying the theorem statement. Let $I_{[\sigma](\rho\alpha.A)}$ be given by

$$\dfrac{\dfrac{\Psi_\sigma\ ;\ a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A}{\Psi_\sigma\ ;\ a : [\sigma]A \vdash \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\sigma]A} :: b : [\sigma](\rho\alpha.A)}\ (\rho^+\mathrm{R})}{\Psi_\sigma\ ;\ a : [\sigma](\rho\alpha.A) \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\sigma]A} :: b : [\sigma](\rho\alpha.A)}\ (\rho^+\mathrm{L})$$

Additionally assume that $\rho\alpha.A$ is morally recursion-free. We show that $I_{[\sigma](\rho\alpha.A)}$ is suitably equivalent to $a \to b$. As in previous cases, we proceed by case analysis on elements of their domain. We use the fact that

$$\mathsf{Fold} : (-)_\perp[\![\vdash [\rho\alpha.A/\alpha]A\ \mathsf{type_s}]\!]^+ \to [\![\vdash \rho\alpha.A\ \mathsf{type_s}]\!]^+$$

is an isomorphism of dcpos (see the remarks surrounding eq. (184)) to represent the elements of $[\![\vdash \rho\alpha.A\ \mathsf{type_s}]\!]^+$ in terms of elements of $(-)_\perp[\![\vdash [\rho\alpha.A/\alpha]A\ \mathsf{type_s}]\!]^+$. Similarly, we represent elements of $[\![\vdash \rho\alpha.A\ \mathsf{type_s}]\!]^-$ in terms of elements of $[\![\vdash [\rho\alpha.A/\alpha]A\ \mathsf{type_s}]\!]^-$.

Subcase $(\mathsf{Fold}(\bot), \mathsf{Fold}(a^-))$: Then $\mathsf{Fold}(\bot) = \bot$ and the analysis is identical to previous cases, using eq. (189) and lemma 9.4.1.
Subcase $(\mathsf{Fold}([a^+]), \mathsf{Fold}(a^-))$: Set

$$(\beta, \beta) = [\![\cdot \; ; \; a : [\sigma]A \vdash a \to b :: b : [\sigma]A]\!] \bot (a^+, a^-).$$

By the induction hypothesis,

$$[\![\Psi_\sigma \; ; \; a : [\sigma]A \vdash I_{[\sigma]A} :: b : [\sigma]A]\!] u_\sigma (a^+, a^-) = (\beta, \beta).$$

It follows from eq. (188) that

$$[\![\Psi_\sigma \; ; \; a : [\sigma]A \vdash \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\sigma]A} :: b : [\sigma](\rho\alpha.A)]\!] (a^+, \mathsf{Fold}(a^-)) = (\beta, \mathsf{Fold}([\beta])).$$

Then by eq. (189),

$$\begin{aligned} &[\![\Psi_\sigma \; ; \; a : [\sigma](\rho\alpha.A) \vdash \\ &\qquad \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ \mathsf{unfold};\ I_{[\sigma]A} :: b : [\sigma](\rho\alpha.A)]\!] (\mathsf{Fold}([a^+]), \mathsf{Fold}(a^-)) \\ &= (\mathsf{Fold}([\beta]), \mathsf{Fold}([\beta])). \end{aligned}$$

But

$$\begin{aligned} &[\![\Psi_\sigma \; ; \; a : [\sigma](\rho\alpha.A) \vdash a \to b :: b : [\sigma](\rho\alpha.A)]\!] (\mathsf{Fold}([a^+]), \mathsf{Fold}(a^-)) \\ &= (\mathsf{Fold}([\beta]), \mathsf{Fold}([\beta])) \end{aligned}$$

by eqs. (113) and (187), so we conclude the result. □

9.5. Binary Arithmetic

In this section, we consider an encoding of binary arithmetic. Let binary natural numbers be encoded using the session type

$$\mathtt{bin} = \rho\beta. \oplus \{0 : \beta, 1 : \beta, \$: \mathbf{1}\}.$$

We assume that binary numbers are transmitted with the least significant bit first. Though it is the opposite of the usual network order, in which the most significant bit is sent first, it simplifies the definition of processes. This session type is very similar to the session type `bits` of bit streams given in section 9.3. The key difference is the introduction of the label $, which signals the end of the sequence of bits.

Let $\lceil \cdot \rceil$ be the "ceiling" or greatest integer function. Recall that every natural number n has a unique base-2 representation $(a_k, \ldots, a_0)_2$ where $a_k = 1$ and $a_i \in \{0, 1\}$ for $0 \le i \le k$ are such that $n = \sum_{i=0}^{k} a_i 2^i$. We use this base-2 representation to define a process $\cdot \; ; \cdot \vdash [n]_2 :: b : \mathtt{bin}$ that sends the base two representation of n on the channel b. It is given by

$$\cdot \; ; \cdot \vdash \mathsf{send}\ b\ \mathsf{unfold};\ b.\mathtt{a}_0;\ \cdots;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.\mathtt{a}_k;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.\$;\ \mathsf{close}\ b :: b : \mathtt{bin}$$

where $\mathtt{a}_i$ is the obvious label 0 or 1 corresponding to the coefficient a_i.

Example 9.5.1. The processes $\cdot \; ; \cdot \vdash [4]_2 :: b : \mathtt{bin}$ and $\cdot \; ; \cdot \vdash [5]_2 :: b : \mathtt{bin}$ are respectively given by:

$$\begin{aligned} &\mathsf{send}\ b\ \mathsf{unfold};\ b.0;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.0;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.1;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.\$;\ \mathsf{close}\ b, \\ &\mathsf{send}\ b\ \mathsf{unfold};\ b.1;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.0;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.1;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.\$;\ \mathsf{close}\ b. \end{aligned}$$ ◂

Remark 9.5.2. Binary numbers do not have a unique representation as sequences of bit labels. For example, the following two processes transmit different representations of zero:

$$\begin{aligned} &\cdot \; ; \cdot \vdash \mathsf{send}\ b\ \mathsf{unfold};\ b.0;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.\$;\ \mathsf{close}\ b :: b : \mathtt{bin}, \\ &\cdot \; ; \cdot \vdash \mathsf{send}\ b\ \mathsf{unfold};\ b.0;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.0;\ \mathsf{send}\ b\ \mathsf{unfold};\ b.\$;\ \mathsf{close}\ b :: b : \mathtt{bin}. \end{aligned}$$

Remark 9.5.3. There are communications satisfying the type `bin` that do not represent legitimate base-2 representations of natural numbers, e.g., the infinite stream of labels 1.

Assume now that we want to increment a binary natural number given its base-2 representation $(a_k, \ldots, a_0)_2$. By the grade-school addition algorithm, we proceed from right-to-left along the bits a_i. If a bit is zero, then we set it to one, and copy the remaining bits unchanged; if it is one, then we set it to zero and carry the one over. For example, in grade school, we wrote out the addition of $(1, 0, 1, 1)_2$ and one like so:

$$\begin{array}{rr} & {\scriptstyle 1\ 1} \\ & 1\,0\,1\,1 \\ + & \qquad 1 \\ \hline & 1\,1\,0\,0 \end{array}$$

We can extract a recursive algorithm from this procedure. To do so, we temporarily extend the notion of base-2 representations to allow the empty sequence $()_2$ to also represent zero.[6]

Proposition 9.5.4. *The base-2 representation of the successor of the natural number with base-2 representation $()_2$ is $(1)_2$. The base-2 representation of the successor of the natural number with base-2 representation $(a_k, \ldots, a_0)_2$ is:*

- $(a_k, \ldots, a_1, 1)_2$ *if* $a_0 = 0$*;*
- $(\beta_c, \ldots, \beta_0, 0)_2$ *if* $a_0 = 1$*, where* $(\beta_c, \ldots, \beta_0)_2$ *is the base-2 representation of the successor of the natural number whose base-2 representation is* $(a_k, \ldots, a_1)_2$*.*

Proof. By induction on k. The base case is immediate, so assume the result holds for some k' and consider the base-2 representation $(a_{k+1}, \ldots, a_1, a_0)_2$. It represents the natural number $n = \sum_{i=0}^{k+1} a_i 2^i$. If $a_0 = 0$, then it is immediate that the base-2 representation of $n + 1$ is $(a_k, \ldots, a_1, 1)_2$. Assume now that $a_0 = 1$. Then $n + 1$ is given by:

$$\begin{aligned} &\left(\sum_{i=0}^{k+1} a_i 2^i\right) + 1 \\ &= \left(\sum_{i=1}^{k+1} a_i 2^i\right) + 2 \\ &= \left(2 \sum_{i=0}^{k} a_{i+1} 2^i\right) + 2 \\ &= 2\left(1 + \sum_{i=0}^{k} a_{i+1} 2^i\right) \end{aligned}$$

but we recognize the parenthesized expression as the successor of the natural number whose base-2 representation is $(a_k, \ldots, a_1)_2$, so:

$$\begin{aligned} &= 2\left(\sum_{i=0}^{c} \beta_i 2^i\right) \\ &= \sum_{i=0}^{c} \beta_i 2^{i+1}. \end{aligned}$$

It follows that the base-2 representation of $n + 1$ is $(\beta_c, \ldots, \beta_0, 0)_2$. □

[6] Though this breaks the uniqueness of base-2 representations for zero, it is consistent with the original definition: a nullary sum is equal to zero.

The following process succ implements the recursive algorithm of proposition 9.5.4. It increments a binary natural number received on a and sends the result on b:

$$\begin{aligned}\cdot\,;\, a : \mathtt{bin} \vdash b \leftarrow \{\mathsf{fix}\ s.b \leftarrow \{&\mathsf{unfold} \leftarrow \mathsf{recv}\ a;\\ &\mathsf{send}\ b\ \mathsf{unfold};\\ &\mathsf{case}\ a\ \{\ 0 \Rightarrow b.1;\ a \to b\\ &\qquad \mid 1 \Rightarrow b.0;\ b \leftarrow \{s\} \leftarrow a\\ &\qquad \mid \$ \Rightarrow b.1;\ b.\$;\ a \to b\}\\ &\}\leftarrow a\} \leftarrow a :: b : \mathtt{bin}\end{aligned}$$

Operationally, it implements the grade school addition algorithm. It works by checking each successive component in the bit stream. If a component is zero, then it sets it to one, and then leaves the remaining components unchanged. If a component is one, then it sets it to zero and carries the one using a recursive call.

Our goal is to show that succ correctly implements the successor function, i.e., that

$$\cdot\,;\cdot \vdash a \leftarrow [n]_2\,;\ \mathtt{succ} \equiv [n+1]_2 :: b : \mathtt{bin}$$

for all $n \in \mathbb{N}$.

We begin by computing the denotation of the type bin and of its unfolding

$$\mathtt{BIN} = \oplus\{0 : \mathtt{bin}, 1 : \mathtt{bin}, \$: \mathbf{1}\}.$$

They denote the dI-domains:

$$\begin{aligned}[\![\mathtt{bin}]\!] &= \mathsf{FIX}(X \mapsto ((0 : X_\perp) \oplus (1 : X_\perp) \oplus (\$: [\![\mathbf{1}]\!])))\\ [\![\mathtt{bin}]\!]^+ &= \mathsf{FIX}(X \mapsto ((0 : X_\perp) \oplus (1 : X_\perp) \oplus (\$: [\![\mathbf{1}]\!]^+)))\\ [\![\mathtt{bin}]\!]^- &= \{\perp\}\\ [\![\mathtt{BIN}]\!] &= (0 : [\![\mathtt{bin}]\!]_\perp) \oplus (1 : [\![\mathtt{bin}]\!]_\perp) \oplus (\$: [\![\mathbf{1}]\!])\\ [\![\mathtt{BIN}]\!]^+ &= (0 : [\![\mathtt{bin}]\!]^+_\perp) \oplus (1 : [\![\mathtt{bin}]\!]^+_\perp) \oplus (\$: [\![\mathbf{1}]\!]^+)\\ [\![\mathtt{BIN}]\!]^- &= \{\perp\}.\end{aligned}$$

By proposition 9.2.1, we know that $\langle\mathtt{bin}\rangle^+$ and $\langle\mathtt{BIN}\rangle^+$ are both given by the identity morphism. These dI-domains are equipped with the natural isomorphisms These dI-domains are equipped with the following canonical isomorphisms:

$$\begin{aligned}\mathsf{Unfold} &: [\![\mathtt{bin}]\!] \to [\![\mathtt{BIN}]\!]_\perp\\ \mathsf{Unfold}^+ &: [\![\mathtt{bin}]\!]^+ \to [\![\mathtt{BIN}]\!]^+_\perp\\ \mathsf{Unfold}^- &: \{\perp\} \to \{(0 : \perp, 1 : \perp, \$: \perp)\}\end{aligned}$$

Their respective inverses are Fold, Fold^+, and Fold^-.

We use the following suggestive notation for elements of $[\![\mathtt{bin}]\!] = [\![\mathtt{bin}]\!]^+$:

$$\begin{aligned}((\beta, o))_2 &= \mathsf{Fold}\,([(0, [\beta])])\\ ((\beta, 1))_2 &= \mathsf{Fold}\,([(1, [\beta])])\\ ((\perp))_2 &= \mathsf{Fold}([\perp])\\ ((\$^\perp))_2 &= \mathsf{Fold}([(\$, [\perp])])\\ ((\$))_2 &= \mathsf{Fold}([(\$, [\mathsf{close}])]).\end{aligned}$$

In particular, write $((b_k, \ldots, b_o))_2$ for $((\cdots((((\$))_2, b_k))_2 \cdots, b_o))_2$.

Next, we observe the correspondence between the process that sends n and the base-2 representation of n:

PROPOSITION 9.5.5. *For all $n \in N$,*

$$[\![\cdot\,;\cdot \vdash [n]_2 :: a : \mathtt{bin}]\!]\bot\bot = ((b_k, \ldots, b_0))_2$$

where $(b_k, \ldots, b_0)_2$ is the base-2 representation of n.

Proof. By induction on k. If $k = 0$, then the result follows by computation using eqs. (122), (166) and (188). Assume the result for some k, and consider the case where n has a base-2 representation $(b_{k+1}, \ldots, b_0)_2$. Then $(b_{k+1}, \ldots, b_1)_2$ is also a base-2 representation, and by the induction hypothesis,

$$\left[\!\!\left[\cdot\,;\cdot \vdash \left[\sum_{i=1}^{k+1} b_i 2^i\right]_2 :: a : \mathtt{bin}\right]\!\!\right]\bot\bot = ((b_{k+1}, \ldots, b_1))_2.$$

But

$$[n]_2 = \mathsf{send}\ b\ \mathsf{unfold};\ b.b_0;\ \left[\sum_{i=1}^{k+1} b_i 2^i\right]_2.$$

The result then follows by a computation using eqs. (166) and (188). □

Now we turn our attention to computing the denotation of `succ`. To do so, we first compute the denotations of each branch of the case statement. Let $\tau = \{b : \mathtt{bin} \leftarrow a : \mathtt{bin}\}$. The branch `0` has the derivation

$$\dfrac{\dfrac{}{s : \tau\,;\, a : \mathtt{bin} \vdash a \to b :: b : \mathtt{bin}}\ (\text{FWD}^+)}{s : \tau\,;\, a : \mathtt{bin} \vdash b.1;\ a \to b :: b : \mathtt{BIN}}\ (\oplus\text{R})$$

and, by eqs. (113) and (166) and corollary 9.2.2, its denotation is the stable function

$$[\![s : \tau\,;\, a : \mathtt{bin} \vdash b.1;\ a \to b :: b : \mathtt{BIN}]\!]\_(a^+, \_) = (a^+, (1, [a^+]))\,.$$

Similarly, the denotation of the `$` branch is:

$$[\![s : \tau\,;\, a : \mathtt{bin} \vdash b.1;\ b.\$;\ a \to b :: b : \mathtt{BIN}]\!]\_(a^+, \_) = (a^+, (1, [(\$, [a^+])]))\,.$$

The case of the `1` branch is more interesting because of its interaction with the functional layer. Its derivation is

$$\dfrac{\dfrac{\dfrac{}{s : \tau \Vdash s : \tau}\ (\text{F-VAR})}{s : \tau\,;\, a : \mathtt{bin} \vdash b \leftarrow \{s\} \leftarrow a :: b : \mathtt{bin}}\ (\text{E-}\{\})}{s : \tau\,;\, a : \mathtt{bin} \vdash b.0;\ b \leftarrow \{s\} \leftarrow b :: b : \mathtt{BIN}}\ (\oplus\text{R})$$

Its denotation is

$$[\![s : \tau\,;\, a : \mathtt{bin} \vdash b.0;\ b \leftarrow \{s\} \leftarrow b :: b : \mathtt{BIN}]\!]s(a^+, \_) = (a, (0, [b]))$$
$$\text{where } \mathsf{down}(s)(a^+, \bot) = (a, b).$$

Strictly speaking, the first argument s is an environment $u = (s : v) \in [\![s : \tau]\!]$. We identify this unary tuple with its single entry for convenience.

Write B_l for the branch corresponding to the label l. Combining these three branches, we can compute the denotation of the case statement using eq. (167):

$$[\![s : \tau\,;\, a : \mathtt{BIN} \vdash \mathsf{case}\ a\ \{l \Rightarrow B_l\}_{l \in \{0,1,\$\}} :: b : \mathtt{BIN}]\!]s(a^+, \_)$$
$$= \begin{cases} (\bot, \bot) & \text{if } a^+ = \bot \\ ((0, [\alpha^+]), (1, [\alpha^+])) & \text{if } a^+ = (0, [\alpha^+]) \\ ((1, a), (0, [b])) & \text{if } a^+ = (1, [\alpha^+]) \\ ((\$, \alpha^+), (1, [(\$, [\alpha^+])])) & \text{if } a^+ = (\$, [\alpha^+]) \end{cases}$$
$$\text{where } \mathsf{down}(s)(\alpha^+, \bot) = (a, b).$$

By eqs. (188) and (189), we then deduce:

$$[\![s : \tau\,;\, a : \mathtt{bin} \vdash \mathsf{unfold} \leftarrow \mathsf{recv}\ a;\ \mathsf{send}\ b\ \mathsf{unfold};\ \mathsf{case}\ a\ \{l \Rightarrow B_l\}_{l\in\{0,1,\$\}} :: b : \mathtt{bin}]\!]s(a^+, \_)$$

$$= \begin{cases} (\bot, \bot) & \text{if } a^+ = \bot \\ (((\alpha^+, 0))_2, ((\alpha^+, 1))_2) & \text{if } a^+ = ((\alpha^+, 0))_2 \\ (((a, 1))_2, ((b, 0))_2) & \text{if } a^+ = ((\alpha^+, 1))_2 \\ (((\$^\bot))_2, ((\$^\bot, 1))_2) & \text{if } a^+ = ((\$^\bot))_2 \\ (((\$))_2, ((\$, 1))_2) & \text{if } a^+ = ((\$))_2 \end{cases}$$

where $\mathsf{down}(s)(\alpha^+, \bot) = (a, b)$.

Finally, we compute the denotation of `succ`. By eqs. (137) and (143), it is given

$$[\![\cdot\,;\, a : \mathtt{bin} \vdash \mathtt{succ} :: b : \mathtt{bin}]\!]\bot = \mathsf{down}(\mathrm{lfp}(\Phi))$$

where $\Phi : [\![\tau]\!] \to [\![\tau]\!]$ is the function

$$\Phi(s) = \mathsf{up}\left(\begin{array}{l} \lambda(a^+, \_) \in [\![\mathtt{bin}]\!]^+ \times [\![\mathtt{bin}]\!]^-.\ \begin{cases} (\bot, \bot) & \text{if } a^+ = \bot \\ (((\alpha^+, 0))_2, ((\alpha^+, 1))_2) & \text{if } a^+ = ((\alpha^+, 0))_2 \\ (((a, 1))_2, ((b, 0))_2) & \text{if } a^+ = ((\alpha^+, 1))_2 \\ (((\$^\bot))_2, ((\$^\bot, 1))_2) & \text{if } a^+ = ((\$^\bot))_2 \\ (((\$))_2, ((\$, 1))_2) & \text{if } a^+ = ((\$))_2 \end{cases} \\ \qquad \text{where } \mathsf{down}(s)(\alpha^+, \bot) = (a, b). \end{array}\right).$$

By proposition 9.5.5 and an argument similar to the one establishing eq. (273), we deduce that:

$$[\![\cdot\,;\, \cdot \vdash a \leftarrow [n]_2\,;\ \mathtt{succ} :: a : \mathtt{bin}]\!]\bot\_ = \mathsf{down}(\mathrm{lfp}(\Phi))((b_k, \ldots, b_0))_2 \tag{275}$$

where $(b_k, \ldots, b_0)_2$ is the base-2 representation of n.

Proposition 9.5.6. *The process* `succ` *implements the successor function, i.e.,*

$$\cdot\,;\, \cdot \vdash a \leftarrow [n]_2\,;\ \mathtt{succ} \equiv [n+1]_2 :: b : \mathtt{bin}$$

for all $n \in \mathbb{N}$.

Proof. Consider some arbitrary $n \in \mathbb{N}$. By eq. (275), it is sufficient to show that

$$\mathsf{down}(\mathrm{lfp}(\Phi))((b_k, \ldots, b_0))_2 = ((\beta_m, \ldots, \beta_0))_2$$

where $(b_k, \ldots, b_0)_2$ and $(\beta_m, \ldots, \beta_0)_2$ are respectively the base-2 representations of n and $n+1$.

We know that $(\beta_m, \ldots, \beta_0)_2$ can be computed from $(b_k, \ldots, b_0)_2$ using the recursive algorithm given in proposition 9.5.4. As a result, it is sufficient to show that for any $(b_k, \ldots, b_0)_2$, if

$$\mathsf{down}(\mathrm{lfp}(\Phi))((b_k, \ldots, b_0))_2 = ((\beta_m, \ldots, \beta_0))_2,$$

then $(\beta_m, \ldots, \beta_0)_2$ is the result of applying proposition 9.5.4 to $(b_k, \ldots, b_0)_2$.

We proceed by induction on the number of bits in $(b_k, \ldots, b_0)_2$. If it is the empty sequence, then it corresponds to the element $((\$))_2$, and

$$\mathsf{down}(\mathrm{lfp}(\Phi))((\$))_2 = \mathsf{down}(\Phi(\mathrm{lfp}(\Phi)))((\$))_2 = ((\$, 1))_2,$$

as desired.

Assume the result for sequences of length k, and consider a sequence $(b_k, \ldots, b_0)_2$. We proceed by case analysis on b_0. If $b_0 = 0$, then

$$\mathsf{down}(\mathrm{lfp}(\Phi))((b_k, \ldots, b_1, b_0))_2 = \mathsf{down}(\Phi(\mathrm{lfp}(\Phi)))((b_k, \ldots, b_1, 0))_2 = ((b_k, \ldots, b_1, 1))_2,$$

as desired. If $b_1 = 1$, then

$$\mathsf{down}(\mathrm{lfp}(\Phi))((b_k, \ldots, b_1, b_0))_2 = \mathsf{down}(\Phi(\mathrm{lfp}(\Phi)))((b_k, \ldots, b_1, 1))_2 = ((\delta_c, \ldots, \delta_0, 0))_2$$

where $\mathsf{down}(\mathsf{lfp}(\Phi))((b_k, \ldots, b_1))_2 = ((\delta_c, \ldots, \delta_o))_2$. By the induction hypothesis, $(\delta_c, \ldots, \delta_o)_2$ is the result of applying proposition 9.5.4 to $((b_k, \ldots, b_1))_2$. It follows that $(\delta_c, \ldots, \delta_o, 0)_2$ is the result of applying proposition 9.5.4 to $((b_k, \ldots, b_1, b_o))_2$. This completes the case $b_1 = 1$. We conclude the result. □

CHAPTER 10

Summary and Future Research

In this dissertation, we developed a variety of techniques for reasoning about Polarized SILL and its programs, and we made contributions to the mathematical foundations of programming semantics to support them. We summarize these contributions and we discuss their potential applications to future research. We also discuss remaining open problems that are directly related to our contributions.

In chapter 6, we developed an observed communication semantics for Polarized SILL. We defined the meaning of a session type to be the set of communications it allows, and we showed that this set could be endowed with a notion of approximation. Then, we showed how to observe the communications sent by processes and configurations in the course of an execution. Importantly, we showed that all fair executions of configurations resulted in the same observed communications. This fact reflects the confluence property satisfied by Polarized SILL.

We introduced a framework for extensional, observational notions of equivalence for Polarized SILL in chapter 7. It was inspired by the "testing equivalences" framework of De Nicola and Hennessy [DH84; Hen83; De 85]. Both frameworks are similar in that they deem processes to be equivalent whenever they are indistinguishable through experimentation. The frameworks differ, however, in the notion of experimentation. Subjecting processes to classical experiments could potentially result in a "success" state, and two processes were equivalent if they succeeded the same experiments. Instead of defining experimental indistinguishability using observed states, we defined it in terms of observed communications. In particular, our experiments communicated with processes (strictly speaking, with configurations of processes), and we deemed processes to be equivalent if we could not observe any differences in their communications. We had a certain latitude in choosing which communications to observe, and this latitude resulted in different notions of process equivalence. One of these, "external observational equivalence", coincided with barbed congruence. We showed how to lift observational congruences on configurations to certain restricted forms of congruences on processes.

We introduced *CYO pluricategories* to model systems with bidirectional communication in section 8.2. Intuitively, objects in CYO pluricategories represent bidirectional communication protocols, while morphisms represent communicating processes. Concretely, objects are embeddings $A \to A^+ \times A^-$ in an underlying category, where A captures bidirectional communications allowed by a protocol, and the embedding describes a decomposition of bidirectional communications into unidirectional communications. Morphisms $\Delta \to \Gamma$ represent communicating systems that use communications Δ to provide communications Γ. They are morphisms $\Delta^+ \times \Gamma^- \to \Delta \times \Gamma$ in the underlying category that describe how to complete unidirectional input received on Δ^+ and Γ^- into complete bidirectional communications.

We used CYO pluricategories to give Polarized SILL a denotational semantics in chapter 8. To capture desirable computational properties, we interpreted protocols and processes in a CYO pluricategory over the category **Stab** of dI-domains and stable maps. The functional layer had the usual stable semantics. Our denotational semantics is notable for being the first to handle general recursion at the protocol and process layers, combined with a functional layer and value transmission, and other rich protocols.

The unifying theme of these contributions is that we have defined the meaning of processes in terms of their communications. In doing so, we have stayed faithful to the process abstraction:

communication is the only phenomenon of processes. Together, they serve as compelling proof of the following thesis statement:

> *Communication-based semantics elucidate the structure of session-typed languages and allow us to reason about programs written in these languages.*

These contributions required major extensions to their underlying mathematical foundations. For our observed communication semantics to be well-defined and for it to capture our semantic intuitions, we had to first develop fairness for multiset rewriting systems in chapter 3. We discovered three independent varieties of fairness—rule fairness, fact fairness, and instantiation fairness—and saw how each subdivided along the axis of weak and strong fairness. All six forms of fairness are subsumed by a particularly strong form of fairness called *über fairness*. We studied properties of fair traces, constructed a scheduler, and gave sufficient conditions for multiset rewriting systems to have fair traces. We observed that under certain conditions, all varieties of fairness coincided. We introduced a notion of trace equivalence called "union equivalence" and studied the effects of permutations on fairness. In particular, we showed that, subject to certain conditions, all fair executions are permutations of each other and that all fair executions are union-equivalent.

To define the denotations of recursive session types, we first had to explore the 2-categorical structure of parametrized fixed points of functors in chapter 4. We showed that parametrized fixed points of ω-functors could be given by a Conway operator, and we showed that unfolding parametrized fixed points was given by a modification. We used these facts repeatedly in chapter 8 when reasoning about recursive types.

There are many open questions related to the above contributions. We highlight the most important.

(1) *Do junk-free, frugal, complete functions form a dI-domain?* If so, then we could drop assumption 8.1.8 and allow quoted processes to be sent by session-typed processes. An affirmative answer would also simplify the semantics of the functional layer: we could interpret all types in the functional layer as dI-domains and all functional terms as stable maps, instead of being forced to interpret some types as dcpos and some functions as only continuous.

(2) *How do we lift observational congruences on configurations to (full) congruences on processes?* We showed in section 7.5 that observational congruences on configurations induced certain restricted classes of congruences on processes. However, the subtle interplay between the process and functional layers prevented us from showing that they induced full congruences. We conjecture that we could develop a version of Howe's method [How96] for languages with adjunctions to show this result.

    This question has important implications for practical applications of our observational congruences. Indeed, the reason congruence relations are so sought after is that they allow us to replace equals by equals. If we could do so, then we could use them to reason about, e.g., program optimizations.

(3) *What is the relationship between the partial orders of session-typed communications given in section 6.1 and the dI-domains of complete session-typed communications given in section 8.3?* Are they isomorphic? How are the observed communications of processes related to the complete communications in the image of their denotations? We conjecture that answers to these questions could result in a new soundness proof for our denotational semantics.

(4) *What general structure underlies initial fixed point categories and canonical fixed point categories?* In sections 4.2 and 4.3, we studied initial fixed points and parametrized fixed points of ω-functors. These results carried over, almost unchanged, to locally continuous functors and $\mathbf{O}$-categories in section 4.5. How can we unify these two analyses into a single framework?

(5) *Can we use a graphical language to reason about typing derivations for configurations?* If so, then we would be spared from having to tediously reassociate or commute compositions of configurations using syntactic arguments.

There are also many directions in which our contributions could be extended. The most interesting directions include reasoning about dependent session types and reasoning about computational interpretations of adjoint logic.

Dependent protocols are an important class of real-world protocols unsupported by Polarized SILL. Dependent protocols prescribe communications where some messages may depend on previous messages. An example dependent protocol is the 3-way handshake [Tom75; SD78; RFC793] used to negotiate TCP connections. In the first step of a 3-way handshake, a process sends its peer a natural number n. Its peer must then reply with the natural number $n + 1$, i.e., with a value that depends on a previously received value. The Heartbeat TLS protocol extension is another example of a dependent protocol. It is used by processes to ensure that their peers are still reachable. They do so by exchanging "heartbeat" messages. Roughly speaking, a process sends its peer the message "Here are n bits of data", followed by said data. Its peer replies with "Here are those n bits", followed by the data it received. The dependency arises from the fact that the only data allowed in the reply is the data that it received.

There already exist dependently session-typed languages that support various sorts of dependent protocols [TY18; TV19; TCP11; DP20], but it is unclear how to combine these different kinds of dependency in a single language. It is also unclear how to extend these languages to support general recursion.

Our various semantics provide an ideal framework in which to study these questions. Indeed, program equivalence is the crux of any dependently typed language, and our observational and denotational semantics both provide notions of program equivalence. Moreover, denotational semantics have historically excelled at capturing recursion, and we believe ours could be used to study the interactions between general recursion and various forms of dependency.

Adjoint logic gives a framework for conservatively combining multiple intuitionistic logics with varying structural properties [Pru+18]. Its computational interpretations uniformly combine message-passing concurrency, shared-memory functionality, and sequential computation [PP19b]. Its message-passing interpretation is notable because it permits communication patterns not possible in Polarized SILL or in other languages that use binary session types [PP21]. These include *multicast*, i.e., sending one message to multiple clients, and replicable services, where a service replicates itself on-demand to handle requests from multiple clients. These richer communication patterns are found in real-world software, and we would like to extend our semantics to be able to reason about them.

Bibliography

[81] *Proceedings of the Sixth IBM Symposium on Mathematical Foundations of Computer Science: Logic Aspects of Programs*. 6th IBM Symposium on Mathematical Foundations of Computer Science (Hakone, Japan, May 25–27, 1981). Tokyo, Japan: Corporate & Scientific Programs, IBM Japan, 1981. 431 pp. (cit. on p. 296).

[Abr07] Samson Abramsky. "Event Domains, Stable Functions and Proof-Nets". In: *Electronic Notes in Theoretical Computer Science* 172 (Apr. 1, 2007): *Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin*, pp. 33–67. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2007.02.003 (cit. on p. 23).

[Abr96] Samson Abramsky. "Retracing Some Paths in Process Algebra". In: *CONCUR '96: Concurrency Theory*. Concur '96 : 7th International Conference on Concurrency Theory (Pisa, Italy, Aug. 26–29, 1996). Ed. by Ugo Montanari and Vladimiro Sassone. Lecture Notes in Computer Science 1119. Springer-Verlag Berlin Heidelberg, 1996. ISBN: 978-3-540-70625-0. DOI: 10.1007/3-540-61604-7 (cit. on pp. 4, 177).

[AGM95] S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, eds. *Handbook of Logic in Computer Science*. Vol. 3: *Semantic Structures*. 5 vols. New York: Oxford University Press Inc., June 15, 1995. XV+490 pp. ISBN: 0-19-853762-X.

[AHS02] Samson Abramsky, Esfandiar Haghverdi, and Philip Scott. "Geometry of Interaction and Linear Combinatory Algebras". In: *Mathematical Structures in Computer Science* 12.5 (Oct. 2002), pp. 625–665. ISSN: 1469-8072. DOI: 10.1017/s0960129502003730 (cit. on pp. 29, 177).

[AJ94] S. Abramsky and R. Jagadeesan. "New Foundations for the Geometry of Interaction". In: *Information and Computation* 111.1 (May 15, 1994), pp. 53–119. ISSN: 0890-5401. DOI: 10.1006/inco.1994.1041 (cit. on p. 4).

[AJ95] Samson Abramsky and Achim Jung. "Domain Theory". In: *Handbook of Logic in Computer Science*. Vol. 3: *Semantic Structures*. Ed. by S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum. 5 vols. New York: Oxford University Press Inc., June 15, 1995, pp. 1–168. ISBN: 0-19-853762-X (cit. on pp. 18–23, 25–27, 63, 181, 206, 265).

[AL91] Andrea Asperti and Giuseppe Longo. *Categories, Types, and Structures. An Introduction to Category Theory for the Working Computer Scientist*. Foundations of Computing. Cambridge, Massachusetts: The MIT Press, 1991. xi+306 pp. ISBN: 0-262-01125-5 (cit. on p. 9).

[AMM18] Jiří Adámek, Stefan Milius, and Lawrence S. Moss. "Fixed Points of Functors". In: *Journal of Logical and Algebraic Methods in Programming* 95 (Feb. 2018), pp. 41–81. ISSN: 2352-2208. DOI: 10.1016/j.jlamp.2017.11.003 (cit. on pp. 74, 76).

[AO82] Krzysztof R. Apt and Ernst-Rüdiger Olderog. "Proof Rules Dealing With Fairness". Extended Abstract. In: *Logics of Programs*. Logics of Programs Workshop (Yorktown Heights, New York, May 4–6, 1981). Ed. by Dexter Kozen. Lecture Notes in Computer Science 131. Springer-Verlag Berlin Heidelberg, 1982, pp. 1–8. ISBN: 978-3-540-39047-3. DOI: 10.1007/BFb0025770 (cit. on p. 61).

[Atk17] Robert Atkey. "Observed Communication Semantics for Classical Processes". In: *Programming Languages and Systems*. 26th European Symposium on Programming. ESOP 2017 (Uppsala, Sweden, Apr. 22–29, 2017). Ed. by Hongseok Yang. Lecture

Notes in Computer Science 10201. Berlin, Germany: Springer-Verlag GmbH Germany, 2017, pp. 56–82. ISBN: 978-3-662-54434-1. DOI: 10.1007/978-3-662-54434-1_3 (cit. on pp. iii, 2–4, 133, 137, 138, 154, 158, 243).

[BA81] J. Dean Brock and William B. Ackerman. "Scenarios: A Model of Non-Determinate Computation". In: *Formalization of Programming Concepts*. International Colloquium on the Formalization of Programming Concepts. ICFPC'81 (Peniscola, Spain, Apr. 19–25, 1981). Ed. by Josep Díaz and Isidro Ramos. Lecture Notes in Computer Science 107. Springer-Verlag Berlin Heidelberg, 1981, pp. 252–259. ISBN: 978-3-540-38654-4. DOI: 10.1007/3-540-10699-5_102 (cit. on p. 243).

[Bar91] Michael Barr. "$*$-Autonomous Categories and Linear Logic". In: *Mathematical Structures in Computer Science* 1.2 (July 1991), pp. 159–178. ISSN: 1469-8072. DOI: 10.1017/s0960129500001274 (cit. on pp. 3, 138, 243).

[BÉ93] Stephen L. Bloom and Zoltán Ésik. *Iteration Theories. The Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag Berlin Heidelberg, 1993. XV+630 pp. ISBN: 978-3-642-78034-9. DOI: 10.1007/978-3-642-78034-9 (cit. on p. 88).

[BÉ95] Stephen L. Bloom and Zoltán Ésik. "Some Equational Laws of Initiality in 2CCC's". In: *International Journal of Foundations of Computer Science* 6.2 (1995), pp. 95–118. DOI: 10.1142/S0129054195000081 (cit. on pp. 3, 12, 63, 78, 82, 84, 85, 88).

[BÉ96] Stephen L. Bloom and Zoltán Ésik. "Fixed-Point Operations on ccc's. Part I". In: *Theoretical Computer Science* 155.1 (Feb. 25, 1996), pp. 1–38. ISSN: 0304-3975. DOI: 10.1016/0304-3975(95)00010-0 (cit. on pp. 3, 4, 28, 63, 64, 78, 79, 84–86, 88).

[BÉ98] L. Bernátsky and Z. Ésik. "Semantics of Flowchart Programs and the Free Conway Theories". In: *Informatique théorique et Applications / Theoretical Informatics and Applications* 32.1-2-3 (1998), pp. 35–78. ISSN: 0988-5004 (cit. on p. 64).

[Bek84] Hans Bekić. "Definable Operations in General Algebras, and the Theory of Automata and Flowcharts". In: Hans Bekič. *Programming Languages and Their Definition. Selected Papers*. Ed. by C. B. Jones. With an intro. by Cliff B. Jones. Lecture Notes in Computer Science 177. Springer-Verlag Berlin Heidelberg, 1984, pp. 30–55. ISBN: 978-3-540-38933-0. DOI: 10.1007/bfb0048939 (cit. on p. 63).

[Ben94] P. N. Benton. *A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models*. Preliminary Report. Tech. rep. UCAM-CL-TR-352. Cambridge, United Kingdom: Computer Laboratory, University of Cambridge, Oct. 1994. 65 pp. (cit. on p. 88).

[Ben95] P. N. Benton. "A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models". Extended Abstract. In: *Computer Science Logic*. 8th Workshop, CSL '94. Annual Conference of the European Association for Computer Science Logic, CSL '94 (Kazimierz, Poland, Sept. 25–30, 1994). Ed. by Leszek Pacholski and Jerzy Tiuryn. Lecture Notes in Computer Science 933. Springer-Verlag Berlin Heidelberg, 1995, pp. 121–135. ISBN: 978-3-540-49404-1. DOI: 10.1007/BFb0022251 (cit. on p. 88).

[Ber94] Claude Bertrand. "A Natural Semantics of First-Order Type Dependency". In: *Theoretical Computer Science* 123.1 (Jan. 17, 1994), pp. 31–53. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90067-1 (cit. on p. 26).

[BH03] Nick Benton and Martin Hyland. "Traced Premonoidal Categories". In: *RAIRO - Theoretical Informatics and Applications* 37.4 (Oct.–Dec. 2003), pp. 273–299. ISSN: 1290-385X. DOI: 10.1051/ita:2003020 (cit. on pp. 28, 29, 64, 85).

[Boc78] Gregor V. Bochmann. "Finite State Description of Communication Protocols". In: *Computer Networks* 2.4-5 (Sept.–Oct. 1978), pp. 361–372. ISSN: 0376-5075. DOI: 10.1016/0376-5075(78)90015-6 (cit. on p. 1).

[BP17] Stephanie Balzer and Frank Pfenning. "Manifest Sharing With Session Types". In: *Proceedings of the ACM on Programming Languages* 1.ICFP, 37 (Sept. 2017). ISSN: 2475-1421. DOI: 10.1145/3110281 (cit. on p. 60).

[Bro88] Manfred Broy. "Nondeterministic Data Flow Programs: How To Avoid the Merge Anomaly". In: *Science of Computer Programming* 10.1 (Feb. 1988), pp. 65–85. ISSN: 0167-6423. DOI: 10.1016/0167-6423(88)90016-0 (cit. on p. 243).

[BRW85] Stephen D. Brookes, Andrew William Roscoe, and Glynn Winskel, eds. *Seminar on Concurrency*. Seminar on Semantics of Concurrency (Carnegie-Mellon University, Pittsburgh, Pennsylvania, July 9–11, 1984). Lecture Notes in Computer Science 197. Springer-Verlag Berlin Heidelberg, 1985. x+523 pp. ISBN: 978-3-540-39593-5. DOI: 10.1007/3-540-15670-4.

[Brz64] Janusz A. Brzozowski. "Derivatives of Regular Expressions". In: *Journal of the ACM* 11.4 (Nov. 1964), pp. 481–494. ISSN: 0004-5411. DOI: 10.1145/321239.321249 (cit. on pp. 133, 243).

[BW96] Nick Benton and Philip Wadler. "Linear Logic, Monads and the Lambda Calculus". In: *Proceedings*. 11th Annual IEEE Symposium on Logic in Computer Science. LICS'96 (New Brunswick, New Jersey, July 27–30, 1996). IEEE Computer Society Technical Committee on Mathematical Foundations of Computing. Los Alamitos, California: IEEE Computer Society Press, 1996, pp. 420–431. ISBN: 0-8186-7463-6. DOI: 10.1109/LICS.1996.561458 (cit. on p. 88).

[BW99] Michael Barr and Charles Wells. *Category Theory for Computing Science*. 3rd ed. Montreal, Quebec: Les Publications CRM, 1999. xvii+526 pp. ISBN: 2-921120-31-3 (cit. on pp. 9, 13).

[CAA84] J. P. Courtiat, J. M. Ayache, and B. Algayres. "Petri Nets Are Good for Protocols". In: *ACM SIGCOMM Computer Communication Review* 14.2 (June 1984), pp. 66–74. ISSN: 0146-4833. DOI: 10.1145/639624.802062 (cit. on p. 1).

[CBB54] T. W. Chaundy, P. R. Barrett, and Charles Batey. *The Printing of Mathematics. Aids for Authors and Editors and Rules for Compositors and Readers at the University Press, Oxford*. London, United Kingdom: Oxford University Press, 1954. ix+105 pp. (cit. on p. 5).

[Cer+00] I. Cervesato et al. "Interpreting Strands in Linear Logic". In: *2000 Workshop on Formal Methods and Computer Security* (Chicago, Illinois, July 2000). 2000 (cit. on p. 60).

[Cer+03] Iliano Cervesato et al. *A Concurrent Logical Framework II: Examples and Applications*. Research rep. CMU-CS-02-102. Pittsburgh, Pennsylvania: School of Computer Science, Carnegie Mellon University, May 2003. 74 pp. (cit. on p. 49).

[Cer+05] Iliano Cervesato et al. "A Comparison Between Strand Spaces and Multiset Rewriting for Security Protocol Analysis". In: *Journal of Computer Security* 13.2 (Apr. 1, 2005), pp. 265–316. ISSN: 0926-227X. DOI: 10.3233/JCS-2005-13203 (cit. on pp. 39, 41, 43, 60).

[Cer+99] I. Cervesato et al. "A Meta-Notation for Protocol Analysis". In: *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. 12th IEEE Computer Security Foundations Workshop. CSFW'99 (Mordano, Italy, June 28–30, 1999). Los Alamitos, California: IEEE Computer Society, 1999, pp. 55–69. ISBN: 0-7695-0201-6. DOI: 10.1109/CSFW.1999.779762 (cit. on p. 60).

[Cer01] Iliano Cervesato. "Typed Multiset Rewriting Specifications of Security Protocols". In: *Electronic Notes in Theoretical Computer Science* 40 (Mar. 2001): *MFCSIT2000, The First Irish Conference on the Mathematical Foundations of Computer Science and Information Technology*, pp. 8–51. ISSN: 1571-0661. DOI: 10.1016/s1571-0661(05)80035-0 (cit. on pp. 39, 47).

[CGW88] Thierry Coquand, Carl Gunter, and Glynn Winskel. "dI-Domains As a Model of Polymorphism". In: *Mathematical Foundations of Programming Language Semantics*. 3rd Workshop on the Mathematical Foundations of Programming Language Semantics. MFPS'87 (New Orleans, Louisiana, Apr. 8–10, 1987). Lecture Notes

in Computer Science 298. Springer Berlin Heidelberg, 1988, pp. 344–363. ISBN: 9783540389200. DOI: 10.1007/3-540-19020-1_18 (cit. on p. 26).

[Chu40] Alonzo Church. "A Formulation of the Simple Theory of Types". In: *Journal of Symbolic Logic* 5.2 (June 1940), pp. 56–68. ISSN: 1943-5886. DOI: 10.2307/2266170 (cit. on pp. 9, 31).

[CP10] Luís Caires and Frank Pfenning. "Session Types as Intuitionistic Linear Propositions". In: *CONCUR 2010 — Concurrency Theory*. 21st International Conference, CONCUR 2010 (Paris, France, Aug. 31–Sept. 3, 2010). Ed. by Paul Gastin and François Laroussinie. Lecture Notes in Computer Science 6269. Springer-Verlag Berlin Heidelberg, 2010, pp. 222–236. ISBN: 978-3-642-15374-7. DOI: 10.1007/978-3-642-15375-4_16 (cit. on pp. 97, 133).

[CPT12] Luís Caires, Frank Pfenning, and Bernardo Toninho. "Towards Concurrent Type Theory". In: *TLDI'12*. 8th ACM SIGPLAN Workshop on Types in Language Design and Implementation (Philadelphia, Pennsylvania, Jan. 28, 2012). New York, New York: Association for Computing Machinery, Inc., 2012, pp. 1–12. ISBN: 978-1-4503-1120-5. DOI: 10.1145/2103786.2103788 (cit. on p. 271).

[Cro93] Roy L. Crole. *Categories for Types*. Cambridge, United Kingdom: Cambridge University Press, 1993. xvii+335 pp. ISBN: 0-521-45701-7 (cit. on pp. 63, 179, 199).

[CS09] Iliano Cervesato and Andre Scedrov. "Relating State-Based and Process-Based Concurrency Through Linear Logic (Full-Version)". In: *Information and Computation* 207.10 (Oct. 2009): *Special Issue: 13th Workshop on Logic, Language, Information and Computation (WoLLIC 2006)*, pp. 1044–1077. ISSN: 0890-5401. DOI: 10.1016/j.ic.2008.11.006 (cit. on pp. 2, 39, 41, 42, 45, 60). "Relating State-Based and Process-Based Concurrency Through Linear Logic". In: *Electronic Notes in Theoretical Computer Science* 165 (Nov. 22, 2006): *Proceedings of the 13th Workshop on Logic, Language, Information and Computation (WoLLIC 2006)*, pp. 145–176. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2006.05.043.

[CS87] Gerardo Costa and Colin Stirling. "Weak and Strong Fairness in CCS". In: *Information and Computation* 73.3 (June 1987), pp. 207–244. ISSN: 0890-5401. DOI: 10.1016/0890-5401(87)90013-7 (cit. on p. 61).

[CŞ90] Virgil Emil Căzănescu and Gheorghe Ştefănescu. "Towards a New Algebraic Foundation of Flowchart Scheme Theory". In: *Fundamenta Informaticae* 13.2 (June 1990), pp. 171–210 (cit. on p. 28). Repr. of Virgil-Emil Căzănescu and Gheorghe Ştefănescu. "Towards a New Algebraic Foundation of Flowchart Scheme Theory". In: INCREST Preprint Series in Mathematics 43 (Dec. 1987). ISSN: 0250-3638.

[CY19] Simon Castellan and Nobuko Yoshida. "Two Sides of the Same Coin: Session Types and Game Semantics. A Synchronous Side and an Asynchronous Side". In: *Proceedings of the ACM on Programming Languages* 3.POPL, 27 (Jan. 2019), p. 27. DOI: 10.1145/3290340 (cit. on pp. iii, 3, 243).

[Dar82] Ph. Darondeau. "An Enlarged Definition and Complete Axiomatization of Observational Congruence of Finite Processes". In: *International Symposium on Programming*. Fifth International Symposium on Programming (Turin, Italy, Apr. 6–8, 1982). Ed. by Mariangiola Dezani-Ciancaglini and Ugo Montanari. Lecture Notes in Computer Science 137. Springer-Verlag Berlin Heidelberg, 1982, pp. 47–62. ISBN: 9783540391845. DOI: 10.1007/3-540-11494-7_5 (cit. on pp. 4, 154).

[De 85] Rocco De Nicola. "Testing Equivalences and Fully Abstract Models for Communication Processes". PhD thesis. University of Edinburgh, 1985. vi+213 pp. HDL: 1842/16979 (cit. on pp. iii, 2–4, 153, 283).

[DH84] R. De Nicola and M. C. B. Hennessy. "Testing Equivalences for Processes". In: *Theoretical Computer Science* 34.1-2 (1984), pp. 83–133. ISSN: 0304-3975. DOI: 10.1016/0304-3975(84)90113-0 (cit. on pp. iii, 2–4, 153, 283).

[DJP03] Nachum Dershowitz, D. N. Jayasimha, and Seungjoon Park. "Bounded Fairness". In: *Verification: Theory and Practice. Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*. Ed. by Nachum Dershowitz. 2772. Springer-Verlag Berlin Heidelberg, 2003, pp. 304–317. ISBN: 9783540399100. DOI: 10.1007/978-3-540-39910-0_14 (cit. on p. 61).

[DK19] Joshua Dunfield and Neel Krishnaswami. *Bidirectional Typing*. Aug. 16, 2019. arXiv: 1908.05839v1 [cs.PL] (cit. on p. 38).

[DP19] Farzaneh Derakhshan and Frank Pfenning. *Circular Proofs as Session-Typed Processes: A Local Validity Condition*. Aug. 6, 2019. arXiv: 1908.01909v1 [cs.LO] (cit. on p. iii).

[DP20] Ankush Das and Frank Pfenning. *Session Types with Arithmetic Refinements and Their Application to Work Analysis*. Jan. 23, 2020. arXiv: 2001.04439v3 [cs.PL] (cit. on p. 285).

[Dur+14] Zakir Durumeric et al. "The Matter of Heartbleed". In: *IMC'14*. 2014 ACM Internet Measurement Conference. IMC'14 (Vancouver, British Columbia, Nov. 5–7, 2014). ACM SIGCOMM and ACM SIGMETRICS. New York, New York: The Association for Computing Machinery, Inc., 2014, pp. 475–488. ISBN: 9781450332132. DOI: 10.1145/2663716.2663755 (cit. on p. 1).

[Egg21] L. C. Eggan. "Rev. of *A New Approach To the Real Numbers (Motivated By Continued Fractions)*." In: *Mathematical Reviews* MR693180 (2021) (cit. on p. 199). Rev. of G. J. Rieger. "A New Approach To the Real Numbers (Motivated By Continued fractions)". In: *Abhandlungen der Braunschweigischen Wissenschaftlichen Gesellschaft* 33 (1982), pp. 205–217. ISSN: 0068-0737.

[Esc93] Martín Hötzel Escardó. "On Lazy Natural Numbers With Applications To Computability Theory and Functional Programming". In: *ACM SIGACT News* 24.1 (Jan. 1993), pp. 61–67. DOI: 10.1145/152992.153008 (cit. on p. 138).

[Eti+15] Pavel Etingof et al. *Tensor Categories*. Mathematical Surveys and Monographs 2015. Providence, Rhode Island: American Mathematical Society, 2015. xvi+343 pp. ISBN: 978-1-4704-2024-6 (cit. on p. 13).

[Fio94] Marcelo P. Fiore. "Axiomatic Domain Theory in Categories of Partial Maps". PhD thesis. The University of Edinburgh Department of Computer Science, Oct. 1994. v+282 pp. (cit. on pp. 11, 12, 27, 82, 87, 88).

[FOCS'7777] *18th Annual Symposium on Foundations of Computer Science*. Formerly called the Annual Symposium on Switching and Automata Theory. 18th Annual Symposium on Foundations of Computer Science. FOCS'77 (Providence, Rhode Island, Oct. 31–Nov. 2, 1977). IEEE 77 CH1278-1 C. IEEE Computer Society's Technical Committee on Mathematical Foundations of Computing. Long Beach, California: Institute of Electrical and Electronics Engineers, 1977. v+269 pp.

[Fra86] Nissim Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag New York Inc., 1986. xiii+295 pp. ISBN: 978-1-4612-4886-6. DOI: 10.1007/978-1-4612-4886-6 (cit. on pp. 47, 60).

[Fre90] P. Freyd. "Recursive Types Reduced To Inductive Types". In: *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*. Fifth Annual IEEE Symposium on Logic in Computer Science. LICS'90 (Philadelphia, Pennsylvania, June 4–7, 1990). IEEE Computer Society Press, 1990, pp. 498–507. DOI: 10.1109/LICS.1990.113772 (cit. on p. 138).

[Fre91] Peter Freyd. "Algebraically Complete Categories". In: *Category Theory*. Category Theory '90 (Como, Italy, July 22–28, 1990). Ed. by Aurelio Carboni, Maria Cristina Pedicchio, and Giuseppe Rosolini. Lecture Notes in Mathematics 1488. Springer-Verlag Berlin Heidelberg, 1991, pp. 95–104. ISBN: 978-3-540-46435-8. DOI: 10.1007/BFb0084215 (cit. on p. 88).

[Fre92] Peter Freyd. "Remarks on Algebraically Compact Categories". In: *Applications of Categories in Computer Science* (Durham, United Kingdom). Ed. by M. P. Fourman, P. T. Johnstone, and A. M. Pitts. London Mathematical Society Lecture Note Series 177. Cambridge, United Kingdom: Cambridge University Press, 1992, pp. 95–106. ISBN: 0-521-42726-6. DOI: 10.1017/CBO9780511525902.006 (cit. on p. 88).

[GFK84] Orna Grumberg, Nissim Francez, and Shmuel Katz. "Fair Termination of Communicating Processes". In: *Proceedings of the Third Annual Acm Symposium on Principles of Distributed Computing*. Third Annual ACM Symposium on Principles of Distributed Computing. PODC'84 (Vancouver, British-Columbia, Aug. 27–29, 1984). New York, New York: Association for Computing Machinery, 1984, pp. 254–265. ISBN: 0-89791-143-1. DOI: 10.1145/800222.806752 (cit. on p. 61).

[Gie+03] G. Gierz et al. *Continuous Lattices and Domains*. Encyclopedia of Mathematics and its Applications 93. Cambridge, United Kingdom: Cambridge University Press, 2003. xxxvi+591 pp. ISBN: 0-521-80338-1. DOI: 10.1017/CBO9780511542725 (cit. on pp. 18, 21).

[Gie+80] G. Gierz et al. *A Compendium of Continuous Lattices*. Springer-Verlag Berlin Heidelberg, 1980. xix+371 pp. ISBN: 978-3-642-67678-9. DOI: 10.1007/978-3-642-67678-9 (cit. on p. 18).

[Gir06] Jean-Yves Girard. *Le Point Aveugle. Cours de logique*. Vol. 1: *Vers la perfection*. French. 2 vols. Visions des sciences. Paris, France: Hermann Éditeurs, May 2006. xvi+280 pp. ISBN: 2 7056 6633 X (cit. on pp. 24, 252).

[Gir86] Jean-Yves Girard. "The System *F* of Variable Types, Fifteen Years Later". In: *Theoretical Computer Science* 45 (1986), pp. 159–192. ISSN: 0304-3975. DOI: 10.1016/0304-3975(86)90044-7 (cit. on pp. 23, 24).

[GJP18] Hannah Gommerstadt, Limin Jia, and Frank Pfenning. "Session-Typed Concurrent Contracts". In: *Programming Languages and Systems*. 27th European Symposium on Programming. ESOP 2018 (Thessaloniki, Greece, Apr. 14–20, 2018). Ed. by Amal Ahmed. Lecture Notes in Computer Science 10801. Cham: Springer, 2018, pp. 771–798. ISBN: 978-3-319-89884-1. DOI: 10.1007/978-3-319-89884-1 (cit. on pp. iii, 60, 99, 133).

[GM89] Carl A. Gunter and Dana S. Mosses Peter D.and Scott. *Semantic Domains and Denotational Semantics*. Tech. rep. MS-CIS-89-16. Philadelphia, Pennsylvania: University of Pennsylvania Department of Computer and Information Science, Feb. 1989 (cit. on p. 18).

[Gun92] Carl A. Gunter. *Semantics of Programming Languages. Structures and Techniques*. Cambridge, Massachusetts: The MIT Press, 1992. 419 pp. ISBN: 0-262-07143-6 (cit. on pp. 18, 22–24, 27, 179, 193, 228, 235, 237, 241).

[GV10] Simon J. Gay and Vasco T. Vasconcelos. "Linear Type Theory for Asynchronous Session Types". In: *Journal of Functional Programming* 20.1 (Jan. 2010), pp. 19–50. ISSN: 1469-7653. DOI: 10.1017/s0956796809990268 (cit. on p. 133).

[Har16] Robert Harper. *Practical Foundations for Programming Languages*. New York, New York: Cambridge University Press, Mar. 2016. xviii+494 pp. ISBN: 978-1-107-15030-0. DOI: 10.1017/CBO9781316576892 (cit. on pp. 9, 31–33, 113).

[Has99] Masahito Hasegawa. "Recursion from Cyclic Sharing". In: *Models of Sharing Graphs. A Categorical Semantics of `let` and `letrec`*. Distinguished Dissertations. Springer-Verlag London Limited, June 1999. Chap. 7, pp. 83–101. ISBN: 978-1-4471-0865-8. DOI: 10.1007/978-1-4471-0865-8_7 (cit. on pp. 29, 64, 85).

[Hen83] M. Hennessy. "Synchronous and Asynchronous Experiments on Processes". In: *Information and Control* 59.1-3 (1983), pp. 36–83. ISSN: 0019-9958. DOI: 10.1016/s0019-9958(83)80029-1 (cit. on pp. iii, 2–4, 153, 162, 283).

[Hen84] G. J. Henry. "The UNIX System: The Fair Share Scheduler". In: *AT&T Bell Laboratories Technical Journal* 63.8 (Oct. 1984), pp. 1845–1857. ISSN: 0748-612X. DOI: 10.1002/j.1538-7305.1984.tb00068.x (cit. on p. 61).

[Hen87] Matthew Hennessy. "An Algebraic Theory of Fair Asynchronous Communicating Processes". In: *Theoretical Computer Science* 49.2-3 (1987), pp. 121–143. ISSN: 0304-3975. DOI: 10.1016/0304-3975(87)90004-1 (cit. on p. 61).

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. With a forew. by Edsger W. Dijkstra. London, United Kingdom: Prentice-Hall International, UK, Ltd., 1985. viii+256 pp. ISBN: 0-13-153271-5 (cit. on pp. 1, 153).

[Hon93] Kohei Honda. "Types for Dyadic Interaction". In: *CONCUR'93*. 4th International Conference on Concurrency Theory (Hildesheim, Germany, Aug. 23–26, 1993). Ed. by Eike Best. Lecture Notes in Computer Science 715. Berlin: Springer-Verlag Berlin Heidelberg, 1993, pp. 509–523. ISBN: 978-3-540-47968-0. DOI: 10.1007/3-540-57208-2_35 (cit. on pp. iii, 1, 97, 133).

[How96] Douglas J. Howe. "Proving Congruence of Bisimulation in Functional Programming Languages". In: *Information and Computation* 124.2 (Feb. 1, 1996), pp. 103–112. ISSN: 0890-5401. DOI: 10.1006/inco.1996.0008 (cit. on pp. 170, 172, 284).

[Hun74] Thomas W. Hungerford. *Algebra*. Graduate Texts in Mathematics 73. Springer-Verlag New York, Inc., 1974. xxiii+502 pp. ISBN: 978-0-387-90518-1. DOI: 10.1007/978-1-4612-6101-8 (cit. on p. 55).

[HVK98] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. "Language Primitives and Type Discipline for Structured Communication-Based Programming". In: *Programming Languages and Systems*. 7th European Symposium on Programming. ESOP'98 (Lisbon, Portugal, Mar. 28–Apr. 4, 1998). Ed. by Chris Hankin. Lecture Notes in Computer Science 1381. Joint European Conferences on Theory and Practice of Software, ETAPS'98. Springer-Verlag Berlin Heidelberg, 1998, pp. 122–138. ISBN: 978-3-540-69722-0. DOI: 10.1007/BFb0053567 (cit. on pp. iii, 1).

[JR12] Bart Jacobs and Jan Rutten. "An Introduction to (Co)algebra and (Co)induction". In: *Advanced Topics in Bisimulation and Coinduction*. Ed. by Davide Sangiorgi and Jan Rutten. Cambridge Tracts in Theoretical Computer Science 52. Cambridge, United Kingdom: Cambridge University Press, 2012, pp. 38–99. ISBN: 978-1-107-00497-9. DOI: 10.1017/CBO9780511792588.003 (cit. on pp. 140, 144).

[JS91] André Joyal and Ross Street. "The Geometry of Tensor Calculus, I". In: *Advances in Mathematics* 88.1 (July 1991), pp. 55–112. ISSN: 0001-8708. DOI: 10.1016/0001-8708(91)90003-p (cit. on pp. 14, 112).

[JSV96] André Joyal, Ross Street, and Dominic Verity. "Traced Monoidal Categories". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 119.3 (Apr. 1996), pp. 447–468. ISSN: 1469-8064. DOI: 10.1017/s0305004100074338 (cit. on pp. 14, 28, 64, 85).

[Kah74] Gilles Kahn. "The Semantics of a Simple Language for Parallel Programming". In: *Information Processing 74*. 6th IFIP Congress 1974 (Stockholm, Sweden, Aug. 5–10, 1974). Ed. by Jack L. Rosenfeld. International Federation for Information Processing. North-Holland Publishing Company, 1974, pp. 471–475. ISBN: 0-7204-2803-3 (cit. on pp. iii, 3, 4, 175, 176, 243).

[Kav20a] Ryan Kavanagh. "Substructural Observed Communication Semantics". In: *Proceedings*. Combined 27th International Workshop on Expressiveness in Concurrency and 17th Workshop on Structural Operational Semantics. EXPRESS/SOS 2020 (Online, Aug. 31, 2020). Ed. by Ornela Dardha and Jurriaan Rot. Electronic Proceedings in Theoretical Computer Science 322. Aug. 27, 2020, pp. 69–87. DOI: 10.4204/EPTCS.322.7 (cit. on pp. 3, 4, 39, 55, 99).

[Kav20b] Ryan Kavanagh. "Parametrized Fixed Points and Their Applications To Session Types". In: *Electronic Notes in Theoretical Computer Science* 352 (Oct. 2020): *The 36th*

Mathematical Foundations of Programming Semantics Conference, 2020, pp. 149–172. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2020.09.008 (cit. on p. 4).

[Kel77] Robert M. Keller. *Denotational Models for Parallel Programs With Indeterminate Operators*. Research rep. UUCS-77-103. School of Computing, University of Utah, 1977. 27 pp. (cit. on p. 243).

[Ker14] Sean Michael Kerner. *Heartbleed SSL Flaw's True Cost Will Take Time to Tally*. Apr. 19, 2014. URL: https://www.eweek.com/security/heartbleed-ssl-flaw-s-true-cost-will-take-time-to-tally (visited on 01/07/2021) (cit. on p. 1).

[KMP19] Wen Kokke, Fabrizio Montesi, and Marco Peressotti. "Better Late Than Never. A Fully-Abstract Semantics for Classical Processes". In: *Proceedings of the ACM on Programming Languages* 4.POPL, 24 (Jan. 2019), p. 24. DOI: 10.1145/3290337 (cit. on pp. iii, 3, 133, 160, 243).

[KP85] Robert M. Keller and Prakash Panangaden. "Semantics of Networks Containing Indeterminate Operators". In: *Seminar on Concurrency*. Seminar on Semantics of Concurrency (Carnegie-Mellon University, Pittsburgh, Pennsylvania, July 9–11, 1984). Ed. by Stephen D. Brookes, Andrew William Roscoe, and Glynn Winskel. Lecture Notes in Computer Science 197. Springer-Verlag Berlin Heidelberg, 1985, pp. 479–596. ISBN: 978-3-540-39593-5. DOI: 10.1007/3-540-15670-4_23 (cit. on p. 243).

[KP93] G. Kahn and G.D. Plotkin. "Concrete Domains". In: *Theoretical Computer Science* 121.1-2 (Dec. 6, 1993), pp. 187–277. ISSN: 0304-3975. DOI: 10.1016/0304-3975(93)90090-g (cit. on p. 26).

[KPY17] Dimitrios Kouzapas, Jorge A. Pérez, and Nobuko Yoshida. "Characteristic Bisimulation for Higher-Order Session Processes". In: *Acta Informatica* 54.3 (May 2017): *Selected Papers from the 26th International Conference on Concurrency Theory (CONCUR 2015) — Part 3*, pp. 271–341. ISSN: 1432-0525. DOI: 10.1007/s00236-016-0289-7 (cit. on p. iii).

[KS74] G. M. Kelly and Ross Street. "Review of the Elements of 2-categories". In: *Category Seminar*. Sydney Category Theory Seminar (Sydney, NSW, Australia, 1972–1973). Ed. by Gregory M. Kelly. Lecture Notes in Mathematics 420. Springer-Verlag Berlin Heidelberg, 1974, pp. 75–103. ISBN: 978-3-540-37270-7. DOI: 10.1007/BFb0063101 (cit. on p. 11).

[Kwi89] M.Z. Kwiatkowska. "Survey of Fairness Notions". In: *Information and Software Technology* 31.7 (Sept. 1989), pp. 371–386. ISSN: 0950-5849. DOI: 10.1016/0950-5849(89)90159-6 (cit. on pp. 47, 61).

[Lah+18] Shuvendu K. Lahiri et al. "Program Equivalence (Dagstuhl Seminar 18151)". In: *Dagstuhl Reports* 8.4 (Oct. 2, 2018). ISSN: 2192-5283. DOI: 10.4230/DagRep.8.4.1 (cit. on p. 2).

[Lah+20] Ori Lahav et al. *Making Weak Memory Models Fair*. Dec. 2, 2020. arXiv: 2012.01067 [cs.PL] (cit. on p. 61).

[Lam69] Joachim Lambek. "Deductive Systems and Categories II. Standard constructions and closed categories". In: *Category Theory, Homology Theory and Their Applications*. Conference on Category Theory, Homology Theory and Their Applications (Seattle Research Center of the Battelle Memorial Institute, Seattle, Washington, June 24–July 19, 1968). Ed. by Peter J. Hilton. Vol. 2. Lecture Notes in Mathematics 92. Springer-Verlag Berlin Heidelberg, 1969, pp. 76–122. ISBN: 978-3-540-36101-5. DOI: 10.1007/BFb0079385 (cit. on p. 14).

[Lam77] L. Lamport. "Proving the Correctness of Multiprocess Programs". In: *IEEE Transactions on Software Engineering* SE-3.2 (Mar. 1977), pp. 125–143. ISSN: 0098-5589. DOI: 10.1109/tse.1977.229904 (cit. on p. 60).

[Leh76a] Daniel J. Lehmann. "Categories for Fixpoint Semantics". PhD thesis. Coventry, United Kingdom: Department of Computer Science, University of Warwick, 1976. 75 pp. (cit. on pp. 64, 91, 93).

[Leh76b] Daniel J. Lehmann. "Categories for Fixpoint-Semantics". In: *17th Annual Symposium on Foundations of Computer Science*. Formerly called the Annual Symposium on Switching and Automata Theory. 17th Annual Symposium on Foundations of Computer Science. FOCS'76 (Houston, Texas, Oct. 25–27, 1976). IEEE CH1133-8 C. IEEE Computer Society's Technical Committee on Mathematical Foundations of Computing. Long Beach, California: Institute of Electrical and Electronics Engineers, 1976, pp. 122–126. DOI: 10.1109/SFCS.1976.9 (cit. on p. 88).

[Lei04] Tom Leinster. *Higher Operads, Higher Categories*. London Mathematical Society Lecture Note Series 298. Cambridge University Press, Aug. 2004. ISBN: 9780521532150 (cit. on p. 14).

[Leu+88] D. Leu et al. "Interrelationships Among Various Concepts of Fairness for Petri Nets". In: *31st Midwest Symposium on Circuits and Systems* (St. Louis, Missouri, Aug. 9–12, 1988). IEEE Computer Society Press, 1988 (cit. on pp. 48, 61).

[LM16] Sam Lindley and J. Garrett Morris. "Talking Bananas: Structural Recursion for Session Types". In: *ICFP'16*. 21st ACM SIGPLAN International Conference on Functional Programming (Nara, Japan, Sept. 18–24, 2016). Ed. by Jacques Garrigue, Gabriele Keller, and Eijiro Sumii. ACM SIGPLAN. New York, New York: The Association for Computing Machinery, Inc., 2016, pp. 434–447. ISBN: 978-1-4503-4219-3. DOI: 10.1145/2951913.2951921 (cit. on p. iii).

[LMZ19] Bert Lindenhovius, Michael Mislove, and Vladimir Zamdzhiev. "Mixed Linear and Non-Linear Recursive Types". In: *Proceedings of the ACM on Programming Languages* 3.ICFP, 111 (Aug. 2019), pp. 1–29. ISSN: 2475-1421. DOI: 10.1145/3341715 (cit. on pp. 81, 88).

[LPS81] D. Lehmann, A. Pnueli, and J. Stavi. "Impartiality, Justice and Fairness: The Ethics of Concurrent Termination". In: *Automata, Languages and Programming*. Eigth International Colloquium on Automata, Languages and Programming. ICALP'81 (Acre (Akko), Israel, July 13–17, 1981). Ed. by Shimon Even and Oded Kariv. Lecture Notes in Computer Science 115. Springer-Verlag Berlin Heidelberg, 1981, pp. 264–277. ISBN: 978-3-540-38745-9. DOI: 10.1007/3-540-10843-2_22 (cit. on p. 60).

[LS77] Daniel J. Lehmann and Michael B. Smyth. "Data Types". Extended Abstract. In: *18th Annual Symposium on Foundations of Computer Science*. Formerly called the Annual Symposium on Switching and Automata Theory. 18th Annual Symposium on Foundations of Computer Science. FOCS'77 (Providence, Rhode Island, Oct. 31–Nov. 2, 1977). IEEE 77 CH1278-1 C. IEEE Computer Society's Technical Committee on Mathematical Foundations of Computing. Long Beach, California: Institute of Electrical and Electronics Engineers, 1977, pp. 7–12. DOI: 10.1109/SFCS.1977.10 (cit. on pp. 67, 76, 88).

[LS81] Daniel J. Lehmann and Michael B. Smyth. "Algebraic Specification of Data Types: a Synthetic Approach". In: *Mathematical Systems Theory* 14.1 (Dec. 1981), pp. 97–139. ISSN: 1433-0490. DOI: 10.1007/bf01752392 (cit. on pp. 64, 65, 67, 69, 71, 74, 88).

[Mac98] Saunders Mac Lane. *Categories for the Working Mathematician*. 2nd ed. Graduate Texts in Mathematics 5. New York, Berlin, and Heidelberg: Springer-Verlag New York, Inc., 1998. xii+314 pp. ISBN: 0-387-98403-8 (cit. on pp. 9, 91, 208).

[MAH18] Stefan K. Muller, Umut A. Acar, and Robert Harper. "Competitive Parallelism: Getting Your Priorities Right". In: *Proceedings of the ACM on Programming Languages* 2.ICFP, 95 (July 2018). ISSN: 2475-1421. DOI: 10.1145/3236790 (cit. on p. 61).

[Mal10] Octavio Malherbe. "Categorical Models of Computation: Partially Traced Categories and Presheaf Models of Quantum Computation". PhD thesis. Ottawa,

Ontario: Department of Mathematics and Statistics, University of Ottawa, 2010. vii+205 pp. (cit. on pp. 14, 29).

[Mar96] Per Martin-Löf. "On the Meanings of the Logical Constants and the Justifications of the Logical Laws". In: *Nordic Journal of Philosophical Logic* 1.1 (May 1996). Three lectures given in the form of a short course at the meeting Teoria della Dimostrazione e Filosofia della Logica, organized in Siena, 6–9 April 1983., pp. 11–60. URL: https://www.hf.uio.no/ifikk/forskning/publikasjoner/tidsskrifter/njpl/vol1no1/meaning.pdf (cit. on p. 34).

[Mil77] Robin Milner. "Fully Abstract Models of Typed λ-calculi". In: *Theoretical Computer Science* 4.1 (Feb. 1977), pp. 1–22. ISSN: 0304-3975. DOI: 10.1016/0304-3975(77)90053-6 (cit. on pp. 124, 155, 156).

[Mil80] Robin Milner. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science 92. Springer-Verlag Berlin Heidelberg, 1980. vi+171 pp. ISBN: 978-3-662-17142-4. DOI: 10.1007/978-3-540-38311-6 (cit. on pp. 1, 99, 100, 124, 137, 153).

[Mit90] John C. Mitchell. "Type Systems for Programming Languages". In: *Handbook of Theoretical Computer Science*. Vol. B: *Formal Models and Semantics*. Ed. by Jan van Leeuwen. 2 vols. Amsterdam, The Netherlands and Cambridge, Massachusetts: Elsevier Science Publishers B.V. and The MIT Press, May 5, 1990, pp. 365–458. ISBN: 0-262-22039-3 (cit. on p. 20).

[Mog91] Eugenio Moggi. "Notions of Computation and Monads". In: *Information and Computation* 93.1 (July 1991), pp. 55–92. ISSN: 0890-5401. DOI: 10.1016/0890-5401(91)90052-4 (cit. on p. 88).

[MPW92a] Robin Milner, Joachim Parrow, and David Walker. "A Calculus of Mobile Processes, I". In: *Information and Computation* 100.1 (Sept. 1992), pp. 1–40. ISSN: 0890-5401. DOI: 10.1016/0890-5401(92)90008-4 (cit. on p. 1).

[MPW92b] Robin Milner, Joachim Parrow, and David Walker. "A Calculus of Mobile Processes, II". In: *Information and Computation* 100.1 (Sept. 1992), pp. 41–77. ISSN: 0890-5401. DOI: 10.1016/0890-5401(92)90009-5 (cit. on p. 1).

[MS92] Robin Milner and David Sangiorgi. "Barbed Bisimulation". In: *Automata, Languages and Programming*. 19th International Colloquium on Automata, Languages and Programming (Wien, Austria, July 13–17, 1992). Ed. by Werner Kuich. Lecture Notes in Computer Science 623. Springer-Verlag Berlin Heidelberg, 1992, pp. 685–695. ISBN: 978-3-540-47278-0. DOI: 10.1007/3-540-55719-9_114 (cit. on pp. 2, 159, 160).

[MWA19] Stefan K. Muller, Sam Westrick, and Umut A. Acar. "Fairness in Responsive Parallelism". In: *Proceedings of the ACM on Programming Languages* 3.ICFP, 81 (July 2019). ISSN: 2475-1421. DOI: 10.1145/3341685 (cit. on p. 61).

[Pan85] Prakash Panangaden. "Abstract Interpretation and Indeterminacy". In: *Seminar on Concurrency*. Seminar on Semantics of Concurrency (Carnegie-Mellon University, Pittsburgh, Pennsylvania, July 9–11, 1984). Ed. by Stephen D. Brookes, Andrew William Roscoe, and Glynn Winskel. Lecture Notes in Computer Science 197. Springer-Verlag Berlin Heidelberg, 1985, pp. 495–511. ISBN: 978-3-540-39593-5. DOI: 10.1007/3-540-15670-4_24 (cit. on p. 243).

[Par80] David Park. "On the Semantics of Fair Parallelism". In: *Abstract Software Specification*. Winter School (Technical University of Denmark, Copenhagen, Denmark, Jan. 22–Feb. 2, 1979). Ed. by Dines Bjørner. Lecture Notes in Computer Science 86. Springer-Verlag Berlin Heidelberg, 1980. ISBN: 978-3-540-38136-5. DOI: 10.1007/3-540-10007-5_47 (cit. on pp. 3, 60).

[Par82] David Park. "A Predicate Transformer for Weak Fair Iteration". In: *RIMS Kôkyûroku* 454 (Apr. 1982). Also appears in [81], pp. 211–228. ISSN: 1880-2818. HDL: 2433/103001 (cit. on p. 61).

[Pér+12] Jorge A. Pérez et al. "Linear Logical Relations for Session-Based Concurrency". In: *Programming Languages and Systems*. 21st European Symposium on Programming, ESOP 2012 (Tallinn, Estonia, Mar. 24–Apr. 1, 2012). Ed. by Helmut Seidl. Lecture Notes in Computer Science 7211. Heidelberg: Springer-Verlag Berlin Heidelberg, 2012, pp. 539–558. ISBN: 978-3-642-28869-2. DOI: 10.1007/978-3-642-28869-2_27 (cit. on p. 3).

[Pér+14] Jorge A. Pérez et al. "Linear Logical Relations and Observational Equivalences for Session-Based Concurrency". In: *Information and Computation* 239 (Dec. 2014), pp. 254–302. ISSN: 0890-5401. DOI: 10.1016/j.ic.2014.08.001 (cit. on pp. iii, 3).

[Pet77] James L. Peterson. "Petri Nets". In: *ACM Computing Surveys* 9.3 (Sept. 1977), pp. 223–252. ISSN: 1557-7341. DOI: 10.1145/356698.356702 (cit. on p. 47).

[Pet80] C. A. Petri. "Introduction To General Net Theory". In: *Net Theory and Applications*. Advanced Course on General Net Theory of Processes and Systems (Hamburg, Federal Republic of Germany, Oct. 8–19, 1979). Lecture Notes in Computer Science 84. Springer-Verlag Berlin Heidelberg, 1980, pp. 1–19. ISBN: 978-3-540-39322-1. DOI: 10.1007/3-540-10001-6_21 (cit. on p. 47).

[Pfe95] Frank Pfenning. "Structural Cut Elimination". In: *Tenth Annual IEEE Symposium on Logic in Computer Science*. Tenth Annual IEEE Symposium on Logic in Computer Science. LICS'95 (San Diego, California, June 26–29, 1995). 1995, pp. 156–166. ISBN: 0-8186-7050-9. DOI: 10.1109/LICS.1995.523253 (cit. on p. 42).

[PG15] Frank Pfenning and Dennis Griffith. "Polarized Substructural Session Types". In: *Foundations of Software Science and Computation Structures*. 18th International Conference on Foundations of Software Science and Computation Structures. FoSSaCS 2015 (London, United Kingdom, Apr. 11–18, 2015). Ed. by Andrew Pitts. Lecture Notes in Computer Science 9034. Springer-Verlag GmbH Berlin Heidelberg, 2015, pp. 3–32. ISBN: 978-3-662-46678-0. DOI: 10.1007/978-3-662-46678-0_1 (cit. on pp. iii, 2, 97, 105, 133).

[Pie02] Benjamin Pierce. *Types and Programming Languages*. Cambridge, Massachusetts: The MIT Press, 2002. xxi+623 pp. ISBN: 0-262-16209-1 (cit. on p. 198).

[Pit94] Andrew M Pitts. "A Co-Induction Principle for Recursively Defined Domains". In: *Theoretical Computer Science* 124.2 (Feb. 28, 1994), pp. 195–219. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90014-0 (cit. on p. 265).

[Pit96] Andrew M. Pitts. "Relational Properties of Domains". In: *Information and Computation* 127.2 (June 15, 1996), pp. 66–90. ISSN: 0890-5401. DOI: 10.1006/inco.1996.0052 (cit. on pp. 266, 267).

[PP19a] Klaas Pruiksma and Frank Pfenning. "A Message-Passing Interpretation of Adjoint Logic". In: *Proceedings: Programming Language Approaches to Concurrency- and Communication-cEntric Software*. Programming Language Approaches to Concurrency- and Communication-cEntric Software (PLACES) (Prague, Czech Republic, Apr. 7, 2019). Ed. by Francisco Martins and Dominic Orchard. Electronic Proceedings in Theoretical Computer Science 291. European Joint Conferences on Theory and Practice of Software. Apr. 2, 2019, pp. 60–79. DOI: 10.4204/EPTCS.291.6. arXiv: 1904.01290v1 [cs.PL] (cit. on p. 60).

[PP19b] Klaas Pruiksma and Frank Pfenning. "Back to Futures". Oct. 25, 2019. URL: https://www.cs.cmu.edu/~fp/papers/futures19.pdf (visited on 11/06/2019) (cit. on p. 285).

[PP21] Klaas Pruiksma and Frank Pfenning. "A Message-Passing Interpretation of Adjoint Logic". In: *Journal of Logical and Algebraic Methods in Programming* 120, 100637 (Apr. 2021). ISSN: 2352-2208. DOI: 10.1016/j.jlamp.2020.100637 (cit. on pp. 133, 285).

[Pru+18] Klaas Pruiksma et al. "Adjoint Logic". Apr. 24, 2018. URL: https://www.cs.cmu.edu/~fp/papers/adjoint18b.pdf (visited on 11/11/2019) (cit. on p. 285).

[PS92] Prakash Panangaden and Vasant Shanbhogue. "The Expressive Power of Indeterminate Dataflow Primitives". In: *Information and Computation* 98.1 (May 1992), pp. 99–131. ISSN: 0890-5401. DOI: 10.1016/0890-5401(92)90043-F (cit. on p. 243).

[PT00] Benjamin C. Pierce and David N. Turner. "Local Type Inference". In: *ACM Transactions on Programming Languages and Systems* 22.1 (Jan. 2000), pp. 1–44. ISSN: 1558-4593. DOI: 10.1145/345099.345100 (cit. on p. 38).

[Rey98] John C. Reynolds. *Theories of Programming Languages*. New York, New York: Cambridge University Press, 1998. xii+500 pp. ISBN: 978-0-521-10697-9 (cit. on pp. 166, 179).

[RFC793] Information Sciences Institute, University of Southern California. *Transmission Control Protocol*. DARPA Internet Program Protocol Specification. RFC 793. Internet Engineering Task Force, Sept. 1981. DOI: 10.17487/RFC0793 (cit. on p. 285).

[Rie16] Emily Riehl. *Category Theory in Context*. Mineola, New York: Dover Publications, Inc, 2016. ISBN: 978-0-486-80903-8 (cit. on pp. 9–11, 13, 26, 71, 76, 208, 229).

[San12] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge, United Kingdom: Cambridge University Press, Aug. 2012. xii+247 pp. ISBN: 9780511777110. DOI: 10.1017/CBO9780511777110 (cit. on pp. 20, 33–35, 139, 141, 144, 146).

[San92] Davide Sangiorgi. "Expressing Mobility in Process Algebras. First-Order and Higher-Order Paradigms". PhD thesis. University of Edinburgh, 1992. xii+206 pp. HDL: 1842/6569 (cit. on pp. 1, 159, 160).

[Sch72] Horst Schubert. *Categories*. Trans. from the German by Eva Gray. Springer-Verlag Berlin Heidelberg, 1972. xi+385 pp. ISBN: 978-3-642-65364-3. DOI: 10.1007/978-3-642-65364-3 (cit. on p. 9).

[Sco72] Dana Scott. "Continuous Lattices". In: *Toposes, Algebraic Geometry and Logic*. Connections Between Category Theory and Algebraic Geometry & Intuitionistic Logic (Dalhousie University, Halifax, Nova Scotia, Jan. 16–19, 1971). Ed. by F. W. Lawvere. Lecture Notes in Mathematics 274. Springer Berlin Heidelberg, 1972, pp. 97–136. ISBN: 978-3-540-37609-5. DOI: 10.1007/BFb0073967 (cit. on p. 87).

[SD78] Carl A. Sunshine and Yogen K. Datal. "Connection Management in Transport Protocols". In: *Computer Networks* 2.6 (Dec. 1978), pp. 454–473. ISSN: 0376-5075. DOI: 10.1016/0376-5075(78)90053-3 (cit. on p. 285).

[Sel11] P. Selinger. "A Survey of Graphical Languages for Monoidal Categories". In: *New Structures for Physics*. Ed. by Bob Coecke. Lecture Notes in Physics 813. Springer-Verlag Berlin Heidelberg, 2011. Chap. 4, pp. 289–355. ISBN: 978-3-642-12821-9. DOI: 10.1007/978-3-642-12821-9_4 (cit. on pp. 14, 29).

[Sim12] Robert J. Simmons. "Substructural Logical Specifications". PhD thesis. Pittsburgh, Pennsylvania: Computer Science Department, Carnegie Mellon University, Nov. 14, 2012. xvi+300 pp. (cit. on pp. 2, 60, 98).

[Sis83] Aravinda Prasad Sistla. "Theoretical Issues in the Design and Verification of Distributed Systems". PhD thesis. Cambridge, Massachusetts: Harvard University, July 1983. v+140 pp. (cit. on p. 61).

[SP00] Alex Simpson and Gordon Plotkin. "Complete Axioms for Categorical Fixed-Point Operators". In: *15th Annual IEEE Symposium on Logic in Computer Science*. 15th Annual IEEE Symposium on Logic in Computer Science. LICS'00 (Santa Barbara, California, June 26–28, 2000). IEEE Computer Society Technical Committee on Mathematical Foundations of Computing. Los Alamitos, California: IEEE Computer Society, 2000, pp. 30–41. ISBN: 0-7695-0725-5. DOI: 10.1109/LICS.2000.855753 (cit. on pp. 28, 85, 88).

[SP77] M. B. Smyth and G. D. Plotkin. "The Category-Theoretic Solution of Recursive Domain Equations". Extended Abstract. In: *18th Annual Symposium on Foundations of Computer Science*. Formerly called the Annual Symposium on Switching and Automata Theory. 18th Annual Symposium on Foundations of Computer Science. FOCS'77 (Providence, Rhode Island, Oct. 31–Nov. 2, 1977). IEEE 77 CH1278-1 C. IEEE Computer Society's Technical Committee on Mathematical Foundations of Computing. Long Beach, California: Institute of Electrical and Electronics Engineers, 1977, pp. 13–17. DOI: 10.1109/SFCS.1977.30 (cit. on p. 88).

[SP82] M. B. Smyth and G. D. Plotkin. "The Category-Theoretic Solution of Recursive Domain Equations". In: *SIAM Journal on Computing* 11.4 (1982), pp. 761–783. DOI: 10.1137/0211062 (cit. on pp. 27, 64, 74, 76, 86, 88, 243).

[Sta87] Eugene W. Stark. "Concurrent Transition System Semantics of Process Networks". In: *POPL'87*. 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. POPL'87 (Munich, West Germany, Jan. 1987). ACM SIGPLAN. New York, New York: Association for Computing Machinery, 1987, pp. 199–210. ISBN: 978-0-89791-215-0. DOI: 10.1145/41625.41643 (cit. on p. 243).

[Sta90] Eugene W. Stark. "A Simple Generalization of Kahn's Principle To Indeterminate Dataflow Networks". Extended Abstract. In: *Semantics for Concurrency*. International BCS-FACS Workshop (Leicester, United Kingdom, July 23–25, 1990). Ed. by Marta Zofia Kwiatkowska, Michael William Shields, and Richard Monro Thomas. Workshops in Computing. Logic for IT (S.E.R.C.) Springer-Verlag Berlin Heidelberg, 1990, pp. 157–174. ISBN: 978-1-4471-3860-0. DOI: 10.1007/978-1-4471-3860-0_10 (cit. on p. 243).

[Sto77] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press Series in Computer Science. Cambridge, Massachusetts: The MIT Press, 1977. 414 pp. ISBN: 0-262-19147-4 (cit. on p. 179).

[Str19] Tom Strickx. *How Verizon and a BGP Optimizer Knocked Large Parts of the Internet Offline Today*. June 24, 2019. URL: https://blog.cloudflare.com/how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-today/ (visited on 08/10/2021) (cit. on p. 1).

[Sza75] M. E. Szabo. "Polycategories". In: *Communications in Algebra* 3.8 (Jan. 1975), pp. 663–689. ISSN: 1532-4125. DOI: 10.1080/00927877508822067 (cit. on p. 15).

[Tar55] Alfred Tarski. "A Lattice-Theoretical Fixpoint Theorem and Its Applications". In: *Pacific Journal of Mathematics* 5.5 (June 1955), pp. 285–309. ISSN: 0030-8730 (cit. on p. 19).

[TCP11] Bernardo Toninho, Luís Caires, and Frank Pfenning. "Dependent Session Types Via Intuitionistic Linear Type Theory". In: *PPDP'11*. 13th International ACM SIGPLAN Symposium on Principles and Practices of Declarative Programming. PPDP'11 (Odense, Denmark, July 20–22, 2011). New York, New York: Association for Computing Machinery, Inc., 2011, pp. 161–172. ISBN: 978-1-4503-0776-5. DOI: 10.1145/2003476.2003499 (cit. on p. 285).

[TCP13] Bernardo Toninho, Luis Caires, and Frank Pfenning. "Higher-Order Processes, Functions, and Sessions: A Monadic Integration". In: *Programming Languages and Systems*. 22nd European Symposium on Programming. ESOP 2013 (Rome, Italy, Mar. 16–24, 2013). Ed. by Matthias Felleisen and Philippa Gardner. Lecture Notes in Computer Science 7792. Springer-Verlag Berlin Heidelberg, 2013, pp. 350–369. ISBN: 978-3-642-37036-6. DOI: 10.1007/978-3-642-37036-6_20 (cit. on pp. iii, 2, 60, 97, 102, 106, 133).

[Ten95] R. D. Tennent. "Denotational Semantics". In: *Handbook of Logic in Computer Science*. Vol. 3: *Semantic Structures*. Ed. by S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum. 5 vols. New York: Oxford University Press Inc., June 15, 1995, pp. 169–322. ISBN: 0-19-853762-X (cit. on pp. 18, 179, 231).

[THK94] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. "An Interaction-Based Language and Its Typing System". In: *PARLE'94. Parallel Architectures and Languages Europe*. 6th International PARLE Conference (Athens, Greece, July 4–8, 1994). Ed. by Costas Halatsis et al. Lecture Notes in Computer Science 10201. Berlin: Springer-Verlag Berlin Heidelberg, 1994, pp. 398–413. ISBN: 978-3-540-48477-6. DOI: 10.1007/3-540-58184-7_118 (cit. on pp. 97, 133).

[Tom75] Raymond S. Tomlinson. "Selecting Sequence Numbers". In: *ACM SIGOPS Operating Systems Review* 9.3 (July 1975), pp. 11–23. ISSN: 0163-5980. DOI: 10.1145/563905.810894 (cit. on p. 285).

[Ton15] Bernardo Parente Coutinho Fernandes Toninho. "A Logical Foundation for Session-based Concurrent Computation". English and Portuguese. PhD thesis. Universidade Nova de Lisboa, May 2015. xviii+178 pp. (cit. on pp. iii, 160).

[TV19] Peter Thiemann and Vasco T. Vasconcelos. "Label-Dependent Session Types". In: *Proceedings of the ACM on Programming Languages* 4.POPL, 67 (Dec. 2019). ISSN: 2475-1421. DOI: 10.1145/3371135 (cit. on p. 285).

[TY18] Bernardo Toninho and Nobuko Yoshida. "Depending on Session-Typed Processes". In: *Foundations of Software Science and Computation Structures*. 21st International Conference, FOSSACS 2018 (Thessaloniki, Greece, Apr. 14–20, 2018). Ed. by Christel Baier and Ugo Dal Lago. Lecture Notes in Computer Science 10803. European Joint Conferences on Theory and Practice of Software. Cham, Switzerland: SpringerOpen, 2018, pp. 128–145. ISBN: 978-3-319-89366-2. DOI: 10.1007/978-3-319-89366-2_7 (cit. on p. 285).

[Wad14] Philip Wadler. "Propositions As Sessions". In: *Journal of Functional Programming* 24.2-3 (Jan. 31, 2014), pp. 384–418. ISSN: 1469-7653. DOI: 10.1017/s095679681400001x (cit. on pp. 133, 154).

[Wad15] Philip Wadler. "Propositions As Types". In: *Communications of the ACM* 58.12 (Dec. 2015), pp. 75–84. ISSN: 0001-0782. DOI: 10.1145/2699407 (cit. on p. 3).

[Wal05] David Walker. "Substructural Type Systems". In: *Advanced Topics in Types and Programming Languages*. Ed. by Benjamin C. Pierce. Cambridge, Massachusetts: The MIT Press, 2005, pp. 3–43. ISBN: 0-262-16228-8 (cit. on p. 36).

[Wan77] Mitchell Wand. *Fixed-Point Constructions In Order-Enriched Categories*. Tech. rep. 23. Bloomington, Indiana: Computer Science Department, Indiana University, Oct. 1977. 32 pp. (cit. on p. 88).

[YHB07] Nobuko Yoshida, Kohei Honda, and Martin Berger. "Linearity and Bisimulation". In: *The Journal of Logic and Algebraic Programming* 72.2 (July 2007), pp. 207–238. ISSN: 1567-8326. DOI: 10.1016/j.jlap.2007.02.011 (cit. on p. 160).

[Zha91] Guo-Qiang Zhang. *Logic of Domains*. Progress in Theoretical Computer Science 4. Springer Science+Business Media New York, 1991. x+260 pp. ISBN: 9781461204459. DOI: 10.1007/978-1-4612-0445-9 (cit. on pp. 23, 24).

[Zha92] Guo-Qiang Zhang. "dI-Domains As Prime Information Systems". In: *Information and Computation* 100.2 (Oct. 1992), pp. 151–177. ISSN: 0890-5401. DOI: 10.1016/0890-5401(92)90011-4 (cit. on p. 26).

Symbols

Category Theory

Categories

Order Theory

General Judgments

Miscellanea

Multiset Rewriting Systems

Polarized SILL

Interpretations

Judgments

Relations

Types

Index

Page references for definitions and results are given in **bold sans-serif font**.