# Towards a mixed inductive and coinductive logical framework

Zhibo Chen

CMU-CS-21-144

December 2021

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

**Thesis Committee:**
Frank Pfenning, Chair
Karl Crary

*Submitted in partial fulfillment of the requirements for the degree of Master of Science.*

**Keywords:** Logical Frameworks, Mixed Induction and Coinduction

## Abstract

The invention of logical frameworks enables computers to check formal proofs. The main methodology of the logical framework LF and various extensions of LF is to reduce proofs to terms and proof checking to type checking. Formally, we define an encoding relation between the objects of the informal proof system and the objects of the logical framework, and we prove that the encoding is adequate in the sense that the encoding relation is a compositional bijection. Then, to check whether the proof is valid, we just need to check whether the encoded proof term is of the correct judgment type.

Despite the success of the logical framework LF and its extensions, there is one class of proofs that present difficulties when one tries to encode them. Infinitary proof systems, and circular proof systems in particular, provide natural ways to carry out induction and coinduction. However, since terms of LF and its extensions are all finite, there cannot be a direct natural encoding of infinitary proofs into those systems.

In order to solve this problem, we designed CoLF$^1$, the first order fragment of a novel mixed inductive and coinductive logical framework. We develop two type theories in succession, one with non-dependent types and the other with simple-dependent types. Each type theory consists of three components, a semantic theory of infinite terms, a semantic theory of rational terms in which type checking is decidable, and a syntactic theory, in which type checking is implementable. We then present two case studies, one on a subtyping system for recursive types, and the other on a calculus for circular proofs with inductive and coinductive definitions, and we explore to which extent that these systems have natural representations in CoLF$^1$.

## Acknowledgments

My deep thanks go to my research advisor Frank Pfenning. His meticulous guidance and insightful suggestions on every big and little aspects made the research an enjoyable experience during my master's program at Carnegie Mellon University. I would like to thank my thesis committee member Karl Crary for reading through early drafts of the thesis document and providing feedbacks. Besides my thesis committee members, Robert Harper, Stephen Brookes, and Andre Platzer taught me abundant programming language techniques that are used in this work. I am grateful for their instructions. I would like to thank my program advisor Dave Eckhardt, for his support and advice in guiding me through the MSCS program. Finally, I would like to thank various other people, my families, and my friends, for supporting me during my stay here at Pittsburgh.

# Contents

# Chapter 1

# Background and Introduction

## 1.1 Background

### 1.1.1 Logical Frameworks

Logical frameworks provide convenient ways for proof checking of various logics. The first logical framework, LF, was proposed by Harper et al. [HHP93]. To encode a logic in LF, one encodes the judgment forms as types and one encodes derivations as terms. In this way, a proof of a given judgment in the object logic can be represented by a term of the corresponding type in the framework. The encoding must be adequate in the sense that there must be a bijection between proofs of that judgment and the canonical terms of the corresponding type. Furthermore, the encoding must commute with substitution, such that substitutions of proofs in an object logic can be represented by substitutions of terms of the logical framework. Ultimately, the LF logical framework automates proof checking by reducing the problem to type checking. A proof for a judgment is valid if and only if the term corresponding to the proof has the correct type. Furthermore, LF is carefully designed so that the type-checking is decidable.

### 1.1.2 Extensions of the Logical Framework LF

Since the LF logical framework, other logical frameworks are proposed. On one line of work, the logical framework LF is given an interpretation in the style of proof search as a logical program [Pfe94]. The resulting logical framework is called Elf. In Elf, one can check whether a judgment is provable by asking the framework to search for a proof term of the encoded judgment.

Built on the top of Elf, Twelf is a meta-logical framework in which meta-theorems of the underlying logic can be specified and verified [PS98, PS99, Pfe01, HL07]. In Twelf, one can verify whether the judgment with an operational interpretation as defined in Elf is total. This effectively allows one to establish forall-exists statements about the deductive systems encoded in LF.

On another line of research, since LF's derivability consequence relation is structural, ultimately limiting its representability to structural proof systems, LF has been extended with capability to encode various substructural logics. The linear logic framework LLF [CP96] is the logical framework LF extended with linear implications and linear connectives. The concurrent logical framework CLF [Wat+02, Cer+02], and implementation of those [Sch11] enable direct encoding of linear and affine logics and the encoding of concurrent systems.

Research on logical frameworks has generated deep theoretical results and techniques for implementing proof systems. For instance, Harper and Pfenning proposed a normalization algorithm based on the structure of types [HP05]. The technique of canonizing substitutions, the observation that there is a way of avoiding redexes when substituting one $\lambda$-term into another, is discovered as part of the research on CLF [Wat+02, Cer+02, HL07].

### 1.1.3 Infinitary Proof Systems

Infinitary proof systems provide an alternative method for standard inductive reasoning. For purely the coinductive fragment, the validity of functions defined on infinite structures requires the output to be guarded [Coq93]. When mixing coinduction with induction, an additional validity condition must be in place as neither the pure guardedness condition for the coinduction nor the pure subterm check for induction nor a combination of them can guarantee validity. The usual condition involves assigning each inductive and coinductive constructors a natural number priority and checking the validity condition.

Charatonik et al. proposed a priority assignment for the $\mu$-calculus [Cha+98]. Brotherston and Simpson presented an infinitary proof system for inductive proofs [Bro05, BS11]. Fortier and Santocanale designed a system with mixed inductive and coinductive definitions and proposed a similar validity condition that guarantees cut elimination [FS13]. Furthermore, when adapting a similar proof system for proving the liveness property of session-typed processes, Derakhshan and Pfenning proved that similar validity conditions ensure that the system would not admit processes that keep running without communi-

cating [DP19, DP20].

## 1.2 The Research Question

The long term goal is to apply the methodology of logical frameworks to enable representation and checking of infinitary proofs. In this thesis, we address the first order fragment of the framework. Given any circular proof that has a representation in the first order fragment, we expect the framework to be able to check whether the proof has the correct form and whether the proof satisfies the validity condition.

## 1.3 Contributions

The contribution of this thesis will be CoLF$^1$, the first order fragment of a mixed inductive and coinductive logical framework. We present the type theories of CoLF$^1$ with simple terms and no $\lambda$-abstractions. We exemplify the use of the framework by showing that some coinductive proofs and mixed inductive and coinductive proofs can be directly encoded, and their validity can be effectively decided by the type checking algorithm of CoLF$^1$.

This thesis is structured as follows:

- In Chapter 2, we introduce mathematical preliminaries including the method of "depth indexing".
- In Chapter 3, we introduce **SemTT**, a semantic infinitary type theory for simple terms with no dependent types.
- In Chapter 4, we introduce $\mathbf{SemTT_R}$, a restriction of the **SemTT** to rational proof terms.
- In Chapter 5, we introduce **SynTT**, a practical implementable infinitary type theory restricted to rational terms.
- In Chapter 6, we extend the previous three type theories with dependent types. The resulting type theories are called $\mathbf{SemTT^{DT}}$, $\mathbf{SemTT_R^{DT}}$, and $\mathbf{SynTT^{DT}}$.
- In Chapter 7, we carry out a case study on formalizing a subtyping algorithm for iso-recursive types [LBN17]. We demonstrate that the framework leads to a direct and natural theoretical framework of the subtyping algorithm.
- In Chapter 8, we carry out a case study on encoding of various sub-fragments of a circular proof system [FS13]. We demonstrate that in most of the cases, the framework

will result in a natural encoding of infinitary circular proofs.

In Chapter 9, we provide a concluding review of this thesis and mention some of the future work that could be carried out.

# Chapter 2

# Mathematical Preliminaries

We will introduce the fundamental building blocks of our proposed type theories in this chapter.

## 2.1 Depth Indexing

Originally introduced by Appel and McAllester [AM01], step indexing is a technique that can be applied to reason about infinite and non-terminating computations [Ahm04]. Using step indexing, one can prove a property of an infinite sequence of reductions $e_0 \to e_1 \to e_2 \to \ldots$, by induction on the number of reduction steps that we're allowed to observe on the given computation. Definitions by step indexing usually postulate that the given property will always hold on computations of zero depth, i.e. computations that we are not allowed to observe further. Then, we define the condition for the property to hold on $k+1$ depth in terms of one step of the reduction step and whether the property hold on computations of depth $k$. The definition needs to satisfy an additional property called "downward closure" such that the set of terms characterized monotonically decreases as the observation depth increases. In this way, we may use induction to prove a property of a computation with an infinite sequence of reductions.

Because we don't consider $\lambda$-abstractions in the entirety of this thesis, there are no reductions involved as in the typical settings where the step indexing technique is applied. Computations in the usual sense will correspond to the observations of infinite terms in the proposed type theories. We use the term "depth indexing" as the fundamental technique in proving properties of infinite terms. That is, we stratify and define the property by induction on the *observation depth* of the term. In order to prove a property of an infinite term, we prove it by induction on all finite observation depths of the term. By the principle

of depth indexing, the given property will hold on the infinite term.

### 2.1.1 Characterizing Possibly Infinite Terms

We demonstrate the use of depth indexing to characterize possibly infinite terms of CoLF[1].

Informally, a possibly infinite terms characterize the greatest set of terms consistent with the following grammar:

$$\text{Terms (Infinite)} \quad M \quad ::= \quad c\, M_1 \ \ldots \ M_n$$

Formally, we can give the following characterization of infinite terms:

(1) any term is a possibly infinite term of observation depth zero.

(2) a possibly infinite term of observation depth $k+1$ is a constant applied to a finite list of possibly infinite terms of observation depth $k$.

$\boxed{M\ \mathsf{infterm}_k}$ ($M$ is a possibly infinite term of depth $k$)

$$\frac{}{M\ \mathsf{infterm}_0}\ \mathsf{minf0} \qquad\qquad \frac{\forall 1 \leq i \leq n.\ M_i\ \mathsf{infterm}_k}{c\, M_1 \ \ldots \ M_n\ \mathsf{infterm}_{k+1}}\ \mathsf{minf1}$$

Then, when speaking of a possibly infinite term, we mean a possibly infinite term of any observation depth.

$$\text{Terms (Infinite)} \quad M \quad = \quad \{M \mid \forall k. M\ \mathsf{infterm}_k\}$$

We write the judgment $M_{(k)}$ to mean $M\ \mathsf{infterm_k}$. It will usually be the case any property holds of a term of observational depth zero, so $M_{(0)}$ for any possible object $M$. In this way, we further abbreviate the syntax, similar to the BNF format, as follows

$$\text{Terms} \quad M_{(k+1)} \quad ::= \quad c\, M_{1(k)} \ \ldots \ M_{n(k)}$$

That is, any metavariable $M$ which satisfies $M_{(k+1)}$ is of the form $c\, M_{1(k)} \ \ldots \ M_{n(k)}$ where each $M_i$ satisfies the predicate $M_{i(k)}$.

We may use the notation $M_{(\infty)}$ to mean $M_{(k)}$ for all $k$.

Furthermore, if we insist that the step index will go down by one whenever the same syntactic category reappears in the body of the definition, e.g. in this case the infinite term $M$ appears in its definition, then we can omit the step index and claim the definition to be infinite.

$$\text{Terms (Infinite)} \quad M \quad ::= \quad c\, M_1 \ \ldots \ M_n$$

This shall be the methodology we use throughout this thesis.

We also note that a possibly infinite term is finite when all the "leaf" nodes are just a head constant $c$ with no arguments. That is, only the rule $\mathsf{minf1}$ is used to prove $M\ \mathsf{infterm}_k$

for sufficiently large $k$. A possibly infinite term is infinite if the term is not finite. From now on, we may use the phrases 'a term', or 'an infinite term' to refer to a possibly infinite term.

We verify the downward-closure property of the definition of (possibly) infinite terms. Downward closure is a required property of definitions by depth indexing.

**Theorem 2.1.1** (Downward Closure of Infinite Terms)**.** *If $M$ is a (possibly) infinite term of observation depth $k+1$, then $M$ is a (possibly) infinite term of observation depth $k$.*

*Proof.* Directly by induction on $k$.

When $k = 0$, the conclusion holds by clause (1).

When $k > 0$, by definition, $M = c\, M_1 \ldots M_n$, where each $M_i$ is an infinite term of observation depth $k$. By IH, each $M_i$ is of observation depth $k-1$. By rule $M$ is of observation depth $k$. □

## 2.2 Tarski's Fixed Point Theorem

We demonstrate the use of Tarski's fixed point theorem by considering the infinite number $\infty$ (or $\omega$) an example of an infinite term.

### 2.2.1 Example: Infinity as an Infinite Term

Informally, we write infinity as an infinite stack of successors:

$$\infty = succ\,(succ\,(succ \ldots))$$

Informally, we could also write $\infty = succ\,\infty$.

Formally, the term $\infty$ may be characterized by induction on observation depth:

$$\infty_{(k+1)} = succ\,\infty_{(k)}$$

Spelled out, the above notation defines a sequence of predicates $P_k$ for each natural number $k$ such that:

(1) $P_0$ may characterize any term at observation depth zero.

(2) $P_{k+1}$ characterizes a term $succ\,M$ at observation depth $k+1$ if $P_k$ at observation depth $k$ characterize the term $M$ at observation depth $k$.

We verify that if a term $M$ is characterized by $P_k$ for all $k$, then the term is a possibly infinitary term, i.e. $M$ $\mathsf{infterm}_k$ for all $k$. The fact may be proved by proving that $P_k(M)$ implies that $M$ $\mathsf{infterm}_k$ by induction.

We may prove that the informal characterization, $\infty = succ\,\infty$ satisfies $P_k$ by induction on $k$. However, it is unclear that $\infty = succ\,\infty$ is the unique term that satisfies $P_k$ for all $k$.

The uniqueness follows from the fact that the definition $P_k$ makes an observation on the underlying object being defined and postulates its head constant and is thus guarded. The reader may consult the details of the formulation in [AC93].

And thus, we say that $P_k$ is an inductive characterization of the term $\infty = succ\,\infty$.

### 2.2.2 Equality of Infinite Terms

Equality may be defined by using the same methodology of depth indexing.

$\boxed{M =_k M'}$ ($M$ and $M'$ are equal up to depth $k$, defined by induction on $k$)

$$\frac{}{M =_0 M'} \qquad \frac{\forall 1 \leq i \leq n.\ M_i =_k M_i'}{c\,M_1 \cdots M_n =_{k+1} c\,M_1' \cdots M_n'}$$

Note that as in the previous example, the definition of equality presupposes the terms under comparison have the required observation depth. That is, in order to derive $M_i =_k M_j$, it has to be the case that $k \leq max(i, j)$. In other words, is the observation depth of the equality must be less than the observation depth of the term. For instance, the judgment $M_{(3)} =_5 N_{(4)}$ is not derivable because when we get to compare $M'_{(0)} =_2 N'_{(1)}$, we're not allowed to observe the head of the left-hand side as the term has zero observation depth.

**Theorem 2.2.1** (Downward Closure of Equality)**.** *If $M =_{k+1} M'$, then $M =_k M'$.*

*Proof.* Straightforwardly by induction on $k$. □

**Theorem 2.2.2.** *$=_k$ is an equality, i.e. reflexive, symmetric, transitive, and congruent.*

*Proof.* Straightforwardly by induction on $k$. □

We say two infinite terms $M$ and $M'$ are equal, denoted $M = M'$ if $M =_k M'$ for all $k$.

## 2.3 The Priority Assignment

When some terms are required to be finite while others may be infinite, it has been a long observed phenomenon that we need to assign priorities to the term constructors [Cha+98, Sim06, FS13, DP19, DP20]. The motivation is that semantic ambiguities will arise if a term involves both inductive and coinductive constructors. Without a priority assignment, it is unclear whether some given terms are valid or not. Consider the term:

$$E = f\,(g\,E)))$$

The above equation stands for the infinite term, $E = f\,(g\,(f\,(g\,(f\,\ldots))))$, where $f$ is a constructor for some inductive datatype and $g$ is a constructor for some coinductive datatype. Intuitively, we would not say $F = f\,F = f\,(f\,(f\,\ldots))$ is a valid inductive term because inductive terms need to be "smaller" when observing the arguments of the constructor, and the decrease in the size of the term justifies the inductive principle. On the other hand, we are fine with saying that $G = g\,(g\,(g\,\ldots))$ is a valid coinductive term.

However, it is unclear whether $E$ is a valid term or not. As we could have either one of the following mental pictures about what $E$ actually is:

(1) Valid:

$$E = \mathrm{f}\,(\mathrm{g}\,(\mathrm{f}\,(\mathrm{g}\,(\mathrm{f}\,(\mathrm{g}\,\ldots)))))$$

(2) Not Valid:

$$E = \mathrm{f}\,(g\,(\mathrm{f}\,(g\,(\mathrm{f}\,(g\,\ldots)))))$$

That is, in the first case, we view the term $E$ as an infinite stack of $g$'s where the $f$'s are secondary. The term is valid because it is viewed essentially as an infinite stack of coinductive constructors. In the second case, we view the term $E$ as an infinite stack of $f$'s where the $g$'s are secondary. The term is invalid because it is viewed essentially as a stack of inductive constructors.

To distinguish between the above two cases, we need a priority assignment. Following the intuition, assigning a higher priority to $g$ than $f$ will result in a valid term just as in (1), and assigning a higher priority to $f$ than to $g$ will result in an invalid term as in (2).

## 2.4 Proving Universal and Existential Quantifications

We would like the ability to use quantification in our statement of the type theories rules.

A universal statement is to be understood in the following form:

For finite domains:

$$\frac{\mathcal{J}(1) \qquad \mathcal{J}(2) \qquad \ldots \qquad \mathcal{J}(n)}{\forall 1 \leq k \leq n.\mathcal{J}(k)}$$

For infinite domains:

$$\frac{\mathcal{J}(0) \qquad \mathcal{J}(1) \qquad \dots \qquad \mathcal{J}(n) \qquad \dots}{\forall k.\mathcal{J}(k)}$$

An existential statement is to be understood in the following form:

For finite and infinite domains:

$$\frac{\mathcal{J}(n)}{\exists k.\mathcal{J}(k)}(\text{where } n \text{ is a natural number})$$

In this way, the derivation tree of any judgment would be finitely deep but may be infinitely wide.

# Chapter 3

# Semantic Type Theory with Simple Types

In this chapter, we introduce semantic type theory for non-dependent types.

## 3.1 The Extended Signature with Priorities

We extend the kind declaration of LF with an additional constant $\mathsf{cotype}$, which denotes coinductive types. The kind declaration now involves a priority assigned to every type family declaration. A natural design choice of CoLF[1] is to assign priorities to type families, such that all constructor of that type family will have the same priority. The priority is a natural number marked in the top right corner of the kind declaration, i.e. either $\mathsf{type}$ or $\mathsf{cotype}$ declaration. Furthermore, following the convention set forth in [Cha+98] and [FS13], we require $\mathsf{type}$ to have odd priority and $\mathsf{cotype}$ to have even priority. To simplify the proof of adequacy, we require that no two type families share the same priority.

$$\text{Kind} \quad K \quad ::= \quad \mathsf{type}^{\mathbb{N}} \mid \mathsf{cotype}^{\mathbb{N}} \mid P \to K$$

The whole **SemTT** syntax now looks like follows:

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Kind | $K$ | ::= | $\mathsf{type}^{\mathbb{N}} \mid \mathsf{cotype}^{\mathbb{N}} \mid P \to K$ |
| Canonical Type Families | $A$ | ::= | $P \mid P \to A$ |
| Atomic Type Families | $P$ | ::= | $a \mid P\, M$ |
| Terms | $M_{(k+1)}$ | ::= | $c\, M_{1(k)} \ldots M_{n(k)}$ |

We use the priority function $p$ to map from a constructor, a type or a kind into their priority, defined inductively as follows, with respect to signature $\Sigma$:

$\boxed{p(K)}$

$p(\mathsf{type}^k) = k$

$p(\mathsf{cotype}^k) = k$

$p(P \to K) = p(K)$

$\boxed{p(A)}$ and $\boxed{p(P)}$

$p(P \to A) = p(A)$

$p(P\,M) = p(P)$

$p(a) = p(K)$ where $a : K \in \Sigma$

$\boxed{p(c)}$

$p(c) = p(A)$ where $c : A \in \Sigma$

We write $maxP(\Sigma)$ to mean the maximum priority (of any type/cotype) in the signature $\Sigma$. It may be inductively defined as follows:

$\boxed{maxP(\Sigma)}$

1. $p(\cdot) = 0$
2. $p(\Sigma, a : K) = max(maxP(\Sigma), p(K))$
3. $p(\Sigma, c : A) = maxP(\Sigma)$

We use the predicate $pOccurs(\Sigma)$ to denote the set of all priorities occurring in the signature $\Sigma$. It may be inductively defined as follows:

$\boxed{pOccurs(\Sigma)}$

1. $pOccurs(\cdot) = \varnothing$
2. $pOccurs(\Sigma, a : K) = pOccurs(\Sigma) \cup \{p(K)\}$
3. $pOccurs(\Sigma, c : A) = pOccurs(\Sigma)$

We note that $maxP(\Sigma)$ will be the unique greatest element of the set $pOccurs(\Sigma)$ with the usual ordering on natural numbers.

### 3.1.1 Higher and Lower Priorities

We say that $A$ has a higher priority than $B$ if $p(A) < p(B)$. For example, a constructor with priority 2 has a higher priority than a constructor with priority 4. A constructor with priority 0 has the highest priority. We note that the ordering on priority numbers is exactly the opposite of the ordering on priorities, i.e., $A$ has a higher priority than $B$ iff $p(A) < p(B)$.

## 3.2 The Depth Indexing Context $D$

During proof construction, we want to track the current depths of all occurring priorities. The tracking is done through a mathematical structure called "the depth indexing context".

Intuitively the depth indexing context tracks the current observation depth for all types (and cotypes) occurring in $\Sigma$, and admits a lexicographical ordering. Formally, a depth context $D$ is a partial function from natural number to natural numbers with finite domains.

$$D : \mathbb{N} \rightharpoonup \mathbb{N}$$

We let $\mathcal{D}$ be the set of all partial functions from $\mathbb{N} \rightharpoonup \mathbb{N}$ with finite domains, then $D \in \mathcal{D}$.

The set $\mathcal{D}$ admits a lexicographic order $<$ defined as follows:

$D < D'$ for $D, D' \in \mathcal{D}$ iff

(1) $dom(D) = dom(D')$,

(2) there exists an index (natural number), called the *principal index*, $n \in dom(D)$, such that $D(n) < D'(n)$ and for all $0 \leq k < n$, $D(k) \simeq D'(k)$.

Intuitively, the principal index is the index at which $D < D'$ lexicographically.

We say that $D < D'$ on principal index $n$ if $n$ is the index satisfying the condition (2) above.

The notation '$\simeq$' expresses the Kleene equality, which means that either both sides are undefined or both sides are defined and equal.

For each size of the domain, the ordering $<$ on $D$ is well-founded and is the same as the lexicographic ordering on $\mathbb{N}^{|dom(D)|}$.

### 3.2.1 Representing D using the list index

We write $D$ down using the following notation:

$$[D(0), D(1), D(2), \ldots, D(n)]$$

$n$ is the $D(k)$ is undefined for $k > n$. We write $-$ if $D(i)$ is undefined.

We write the empty depth context as $\cdot$, [] or $[-]$.

As an example, $[2, -, 3, 4]$ represents a depth context $D$ that maps 0 to 2, 2 to 3 and 3 to 4. An example of ordering would be $[2, -, 3, 4] < [2, -, 4, 2]$ on principal index 2.

### 3.2.2 Extension Operations on $D$

We define the extension of $D$ with $x$ mapped to $N$, written as $D[x \mapsto N]$, to be the depth indexing context that maps $x$ to $N$ and maps other values $i$ to $D(i)$. When writing $D[x \mapsto N]$, we implicitly require that $x \notin dom(D)$. In this way, $[2, -, 3, 4]$ may be written as

$$[2, -, 3, 4] = [-][0 \mapsto 2][2 \mapsto 3][3 \mapsto 4]$$

Because we required that when writing $D[x \mapsto N]$, $x \notin dom(D)$, any $D$ may be represented as an empty context with an unordered list of extension operations. And thus, we also have

$$[2, -, 3, 4] = [-][0 \mapsto 2][3 \mapsto 4][2 \mapsto 3]$$

$$[2, -, 3, 4] = [-][3 \mapsto 4][2 \mapsto 3][0 \mapsto 2]$$

## 3.3 Type Checking rules

The type checking rules are listed as follows for **SemTT**.

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Kind | $K$ | ::= | $\mathsf{type}^{\mathbb{N}} \mid \mathsf{cotype}^{\mathbb{N}} \mid P \to K$ |
| Canonical Type Families | $A$ | ::= | $P \mid P \to A$ |
| Atomic Type Families | $P$ | ::= | $a \mid P\, M$ |
| Terms | $M_{(k+1)}$ | ::= | $c\, M_{1(k)} \ldots M_{n(k)}$ |

We simultaneously define the following judgments

| | |
|---|---|
| $\Sigma\ \mathsf{ok}$ | Signature $\Sigma$ is type correct |
| $\vdash_\Sigma K \Leftarrow \mathsf{kind}$ | Kind $K$ is a valid kind |
| $\vdash_\Sigma A \Leftarrow \mathsf{type}$ | Type $A$ is a canonical type family |
| $\vdash_\Sigma P \Rightarrow K$ | Atomic type family $P$ synthesizes kind $K$ |
| $\vdash_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ |
| $\vdash^D_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ in the depth context $D$ |

$\boxed{\Sigma\ \mathsf{ok}}$ : $\Sigma$ is a valid signature.

$$\frac{}{\cdot\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Sigma, a : K\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\Sigma, c : A\ \mathsf{ok}}$$

$\boxed{\vdash_\Sigma K \Leftarrow \mathsf{kind}}$ : $K$ is a valid kind, assuming that $\Sigma$ ok.

$$\frac{n \notin pOccurs(\Sigma)}{\vdash_\Sigma \mathsf{type}^n \Leftarrow \mathsf{kind}} (n \text{ is an odd natural number})$$

$$\frac{n \notin pOccurs(\Sigma)}{\vdash_\Sigma \mathsf{cotype}^n \Leftarrow \mathsf{kind}} (n \text{ is an even natural number})$$

$$\frac{\vdash_\Sigma P \Rightarrow K' \qquad K' = \mathsf{type} \,/\, \mathsf{cotype} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\vdash_\Sigma P \to K \Leftarrow \mathsf{kind}}$$

$\boxed{\vdash_\Sigma A \Leftarrow \mathsf{type}}$ : $A$ is a good type, assuming that $\Sigma$ ok.

$$\frac{\vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\vdash_\Sigma P \to A \Leftarrow \mathsf{type}}$$

$$\frac{\vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n}{\vdash_\Sigma P \Leftarrow \mathsf{type}}$$

$\boxed{\vdash_\Sigma P \Rightarrow K}$ : $P$ has kind $K$, assuming that $\Sigma$ ok.

$$\frac{a : K \in \Sigma}{\vdash_\Sigma a \Rightarrow K} \qquad\qquad \frac{\vdash_\Sigma P \Rightarrow P' \to K \qquad \vdash_\Sigma M \Leftarrow P'}{\vdash_\Sigma P\, M \Rightarrow K}$$

$\boxed{\vdash_\Sigma M \Leftarrow P}$ is defined to be:

$$\forall k_0.\forall k_2.\ldots.\exists j_1.\exists j_3.\ldots.\vdash_\Sigma^{[k_0,j_1,k_2,j_3,\ldots]} M \Leftarrow P$$

The length of the depth context is constrained by the largest priority number occurring in $\Sigma$.

To be more specific, $\boxed{\vdash_\Sigma M \Leftarrow P}$ is defined to be:

When $maxP(\Sigma)$ is even :

$$\forall k_0.\forall k_2.\forall k_4\ldots\forall k_{maxP(\Sigma)}.\exists j_1.\exists j_3.\exists j_5.\ldots.\exists j_{maxP(\Sigma)-1}.\vdash_\Sigma^{[k_0,j_1,k_2,j_3,\ldots,k_{maxP(\Sigma)}]} M \Leftarrow P$$

When $maxP(\Sigma)$ is odd

$$\forall k_0.\forall k_2.\forall k_4\ldots\forall k_{maxP(\Sigma)-1}.\exists j_1.\exists j_3.\exists j_5.\ldots.\exists j_{maxP(\Sigma)}.\vdash_\Sigma^{[k_0,j_1,k_2,j_3,\ldots,j_{maxP(\Sigma)}]} M \Leftarrow P$$

$\boxed{\vdash^D_\Sigma M \Leftarrow P}$ : $M$ checks against type $P$ in the depth indexing context $D$, assuming that $\Sigma$ ok.

$$\frac{\text{(Require: } p(P) \text{ is even and } D(p(P)) = 0)}{\vdash^D_\Sigma M \Leftarrow P} cM0$$

$$\frac{\begin{array}{c}\text{(cM1 Requires: (1) } D' < D \text{ on principal index } p(P), \text{ (2) } D'(p(P)) + 1 = D(p(P)), \text{ and (3)} \\ \begin{cases} D'(k) = D(k) \text{ for all } k > p(P) \text{ and } k \text{ is an even number} & \text{if } p(P) \text{ is even} \\ D'(k) = D(k) \text{ for all } k > p(P) & \text{if } p(P) \text{ is odd} \end{cases}\text{)} \\ c : P_1 \to P_2 \to \cdots \to P_n \to P' \in \Sigma \qquad P = P' \qquad \forall 1 \le i \le n.\, \vdash^{D'}_\Sigma M_i \Leftarrow P_i \end{array}}{\vdash^D_\Sigma c\, M_1 \ldots M_n \Leftarrow P} cM1$$

Note that to simplify the presentation, we require all objects that enter type checking to be possibly infinitary terms, i.e. $M$ $\mathsf{infterm}_k$ for any $k$.

The judgment $\vdash^D_\Sigma M \Leftarrow A$ is well-defined by induction on $D$. The axioms of the system are (1) $cM0$ and (2) $cM1$ where we're checking constants with no arguments. For the soundness theorems, we have to impose constraints on the values of $D'(k)$ for $k > p$ on the premise of the rule $cM1$.

As we show in the next section (Theorem 3.4.6), eagerly instantiating all for-all quantifiers before existential quantifiers guarantees the soundness and completeness of priority correctness.

## 3.4 Validity of terms

The type checking rule ensures two properties of the term. One is "form-correctness" and the other is "priority-correctness".

### 3.4.1 Form-Correctness

Form-correctness concerns whether inductive and coinductive constructors construct a correct term. All terms that are typeable on **SemTT** are form-correct but not all form-correct terms are typeable in **SemTT**. Informally, every form correct term corresponds to some pre-proof in an infinitary object logic if this logic admits infinitary proofs.

Formally, form-correctness is defined as:

**Definition 3.4.1** (Form-Correctness)**.** *(Infinite) A term $c\,M_1\,\ldots\,M_n$ is form-correct of type $P$ if $c : P_1 \to \cdots \to P_n \to P$ and for all $1 \leq i \leq n$, $M_i$ is form-correct of type $P_i$.*

Note that the above definition could be rewritten as a definition by depth indexing, and we expect the reader to perform such translation when a definition is marked (infinite). We repeat the translation here:

1. Any term is form-correct of type $P$ at depth 0.
2. A term $c\,M_1\,\ldots\,M_n$ is form-correct at depth $k+1$ of type $P$ if $c : P_1 \to \cdots \to P_n \to P$ and for all $1 \leq i \leq n$, $M_i$ is form-correct at depth $k$ of type $P_i$.

**Theorem 3.4.2.** *If $\vdash_\Sigma M \Leftarrow P$, then $M$ is form-correct of type $P$.*

*Proof.* After quantifier instantiations, we have a set $S$ of derivations of the judgment $\vdash^D_\Sigma M \Leftarrow P$, one for some instantiation of universal quantifiers and existential quantifiers. We add to this set all sub-derivations of every derivation having the form $\vdash^{D'}_\Sigma M' \Leftarrow P'$.

To show $M$ is form-correct of type $P$, we proceed by induction on the observation depth of form-correctness of $M$.

$M$ is form-correct of type $P$ at zero depth.

When depth $k$ is greater than 0, $M$ is of the form $c\,M_1\,\ldots\,M_n$. There must exist derivation $\vdash^D_\Sigma M \Leftarrow P$ in set $S$ ends with an application of the rule $cM1$. The premises of the rule include the fact $c : P_1 \to \cdots \to P_n \to P$, and derivations $\vdash^{D'}_\Sigma M \Leftarrow P_i$. By induction, all $M_i$ are form-correct of type $P$ at depth $k-1$. Then by definition, $M$ is form-correct of type $P$ at depth $k$.

The reason that there must exist derivation $\vdash^D_\Sigma M \Leftarrow P$ in set $S$ ends with an application of the rule $cM1$ for all subcomponents [Coq93] $M' = c\,M_1\,\ldots\,M_n$ of the term $M$ is as follows: if $c$ is an inductive constructor, then the corresponding derivation cannot use $cM0$. If $c$ is a coinductive constructor, and if $cM0$ is used, then the depth context at the use of $cM0$ rule must map $p(c)$ to 0. However, since the mapped to value $p(c)$ is quantified with a universal quantifier, we can pick the same derivation that is exactly the same as this one but instantiates $p(c)$ with a higher value. Then, the rule $cM1$ will be used at the spot where $cM0$ is originally used.

□

### 3.4.2 Priority Correctness

Informally, A term $M$ is priority correct iff the highest priority constructor that occurs infinitely often along any trace is not inductive. The validity is only defined on a term of

infinite observation depth, $M_{(\infty)}$. To make the statement mathematically rigorous, there are a few definitions leading up to the validity of a **SemTT** term.

A pretrace $T$ is an infinite word over $\mathbb{N}$. A prefix $p$ of a pretrace $T$ is a finite word over $\mathbb{N}$, such that $T$ begins with $p$, e.g. $T$ can be written as $p\,T'$ where $T'$ is another pretrace. For instance, 121 is a prefix of the pretrace $12134211139023420123\ldots$. An empty word is denoted $\epsilon$, and is the prefix of any pretrace.

The denotation of any prefix of a pretrace in $M$ is a unique constructor or is undefined. We denote this denotation *det*, defined as follows (by induction on the length of the prefix)

$det(\epsilon, c\,M_1\ \ldots M_n) = c$

$$det(mT, c\,M_1\ \ldots M_n) = \begin{cases} det(T, M_{m+1}) & \text{if } 0 \leq m < n \\ undefined & \text{if } m \geq n \end{cases}$$

**Definition 3.4.3** (Trace)**.** *A trace $T$ in $M$ is a pretrace $T$ in $M$ where all prefixes of $T$ determine a constructor, i.e., $det(p, M) \neq undefined$ for all possible prefixes of $T$, $p$.*

A symbol $c$ occurs infinitely often in $M$ along a trace $T$, iff there are infinitely many prefixes of $T$, $p$, such that $det(p, M) = c$.

We note that an object $M$ is finite, iff for any pretrace $T$, there exists a natural number $l$, called the depth of $M$, such that for all prefixes of $T$, $pT$, if the length of $pT$ is greater than $l$, then $det(pT, M)$ is undefined.

**Definition 3.4.4** (Priority Correctness)**.** *An object $M$ is priority correct iff either*

*(a) $M$ is finite,*

*or (b) for all traces $T$ in $M$, if $c$ occurs infinitely often in $M$ along $T$, then either (1) there exists a coinductive constructor $c'$ of higher priority than that of $c$, ($p(c') > p(c)$) and $c'$ occurs infinitely often in $M$ along $T$, or (2) $c$ is coinductive.*

**Theorem 3.4.5** (Soundness: Typeability Implies Priority Correctness)**.** *If $\vdash_\Sigma M \Leftarrow P$, then $M$ is priority correct.*

*Proof.* We first note that the derivation of $\vdash_\Sigma M \Leftarrow P$ consists of a sequence of quantifier instantiations and derivations of the form $\vdash_\Sigma^D M \Leftarrow P$, where $D = [n_0, n_1, n_2, \ldots, n_m]$.

Next, we show $M$ is priority correct. Either $M$ is finite, or $M$ is infinite. If $M$ is finite, then it is priority correct. Consider the case where $M$ is infinite.

For a trace $T$, since there are finitely many constructors in $\Sigma$, let $S$ be the set of all constructors that occurred along that trace.

For any constructor $c \in S$, either $p(c)$ is even or odd (i.e., either $c$ is an inductive constructor or a coinductive constructor). If $p(c)$ is even (i.e., $c$ is coinductive), then the

priority correctness condition (b)(2) is satisfied for $c$. If $p(c)$ is odd (i.e., $c$ is inductive), we show that there exists a constructor $c'$ with a higher priority.

Assume towards a contradiction that $c$ is the highest priority constructor that occurs infinitely often along that trace. Then the derivation of $\vdash^{D}_{\Sigma} M \Leftarrow P$ consists of an infinite number of rule $cM0$ where the premises have depth contexts that are smaller on principal index $p(c)$.

Consider the term where any finite occurrences of all constructors in $S$ with higher priorities than $c$ have been observed, the depth indexing context will map $p(c)$ to a finite natural number $N$ that is decremented by one each time the constructor $c$ is encountered along that path. Upon $N+1$ occurrences of $c$, the derivation will have the form $\vdash^{D[p(c)\mapsto 0]}_{\Sigma}$ $c\, M_1 \ldots M_n \Leftarrow P'$, and there is no rule in the system that has this form of conclusion. This is a contradiction.

Therefore, there must exist some constructor that has higher priority than $c$ that occurs infinitely often.

□

**Theorem 3.4.6** (Completeness: Priority Correctness Implies Typeability)**.** *For some signature $\Sigma$, if a term $M$ is form-correct of type $P$, and $M$ is priority correct, then $\vdash_{\Sigma} M \Leftarrow P$.*

*Proof.* To show $\vdash_{\Sigma} M \Leftarrow P$, we need to construct derivations with conclusions of the form

$$\vdash_{\Sigma} [k_0, j_1, k_2, j_3, \ldots, (k/j)_{maxP(\Sigma)}] M \Leftarrow P$$

where $(k/j)_{maxP(\Sigma)}$ denotes $k_{maxP(\Sigma)}$ if $maxP(\Sigma)$ is even and $j_{maxP(\Sigma)}$ otherwise.

Due to the nature of universal and existential quantifiers, the values for $k_i$'s are given, and we are free to pick any value for $j_i$'s.

Since $M$ is form-correct, we have a skeleton proof of the conclusion

$$\vdash^{D}_{\Sigma} M \Leftarrow P$$

where $D$ is a not-yet-filled depth context of the form $[k_0, j_1, k_2, j_3, \ldots, (k/j)_{maxP(\Sigma)}]$. It remains to figure out how to pick the priorities for $j$'s and how to handle known values of $k$'s. We note that the ability to instantiate existentially quantified variables after knowing the instantiations of all universally quantified variables is crucial for this proof.

We pick $j$ by assigning each of them a temporary variable, and we instantiate those variables after the construction is complete. Each temporary variable $j_i$ has an associated "minimum value" that is zero when first created.

($\star$) For an arbitrary $M$ with the yet-to-construct derivation with the conclusion,

$$\vdash_{\Sigma} [k_0, j_1 - m_1, k_2, j_3 - m_3, \ldots, kj] M \Leftarrow P$$

where $kj$ denotes $k_{maxP(\Sigma)}$ if $maxP(\Sigma)$ is even and $j_{maxP(\Sigma)} - m_{maxP(\Sigma)}$ otherwise.

An observation of $M$ reveals that $M$ is of the form $c\, M_1 \ldots M_n$. Either $p(c)$ is even or odd.

(a) If $p(c)$ is even, then check whether $k_{p(c)}$ is zero or not. If $k_{p(c)}$ is zero, we cut off the remaining derivation and close the derivation with the rule $cM0$. Otherwise, we apply the rule $cM1$ decrement this index, set all odd indexes with lower priority to map to a fresh unknown variable with the associated minimum value, and proceed to construct derivation for the terms $M_i$ for each $i$, by following the same procedure (starting at ($\star$)).

(b) On the other hand, if $p(c)$ is odd, we know that $j_{p(c)} - m_{p(c)}$ contains an unknown variable. We add one to the minimum value $m_{p(c)}$ associated with the unknown variable $j_{p(c)}$. We update the context by setting $p(c)$ to map to the updated value of $j_{p(c)} - m_{p(c)}$. Then, we proceed to construct a derivation in the updated context for the terms $M_i$ for each $i$, by following the same procedure (starting at ($\star$)).

Next, we prove the termination of the above construction.

First, we show that we are unable to keep adding one to the minimum value associated with an unknown variable. For an arbitrary unknown variable, the priority correctness of $M$ ensures we cannot come back through branch (b) forever and keep incrementing the minimum value associated the same variable. We prove this fact by proving the contrapositive. Suppose we kept adding one to the minimum value of some unknown variable $j_i$ mapped to by index $i$ forever, since we instantiate a new variable at index $i$ when encountering a coinductive constructor of higher priority, there must be no coinductive constructors of higher priority along the specific trace where we keep adding one to $j_i$. However, since $i$ is an odd index, the term is not priority correct.

Second, we show that only finitely many fresh variables will be created in the above construction. Since a fresh variable is created when encountering some coinductive constructor of a higher priority, it is sufficient to show that only finitely many constructors of any given priority will be encountered in the above construction. We show this by induction on the priority $p$ that only finitely many constructors of priority $p$ will be encountered in the above algorithm.

Case $p$ is even, the values corresponding to the even indexes in the depth indexing context are "given" by the universal quantifier and are all finite. Since the corresponding value in the depth context can be decremented only finitely many times until it reaches zero, we only encounter finitely many constructors of priority $p$.

Case $p$ is odd, by the induction hypothesis, only finitely many constructors will be encountered for all priorities $< p$, and thus only finitely many fresh variables will be created at $p$. Whenever we encounter a constructor at priority $p$, we increment the minimum value associated with a specific unknown variable. As proved previously, the number of increments is finite, and since there are only finitely many unknown variables, we will encounter only finitely many constructors at priority $p$.

Since the maximum priority is bounded, we will encounter finitely many constructors in the above construction, and therefore the construction will terminate.

Once the construction terminates, we instantiate all unknown variables with their associated minimum values. Thus, we have constructed a proof of

$$\vdash_{\Sigma}{}^{[k_0, j_1, k_2, j_3, \ldots, (k/j)_{max P(\Sigma)}]} M \Leftarrow P$$

for all given values of $k$ and have computed the suitable values of $j$. From this proof, we can apply the quantifier rules to get a proof of $\vdash_{\Sigma} M \Leftarrow P$.

□

## 3.5 Metatheorems

Sometimes, terms of a given type we are interested in do not involve constructors of some other type. For instance, terms of type *nat* do not contain constructors of type *natlist*. In other words, the type *nat* does not depend on the type *natlist* and therefore the depth of type *natlist* is not involved in the type checking of terms of the type *nat*. Then, when writing out the adequacy theorems for *nat*, we may omit the depth index corresponding to the type *natlist*. We make this intuition precise through the following definitions and theorems.

**Definition 3.5.1** (Type Dependency)**.** *For a fixed signature $\Sigma$, we say that a type $P$* directly depends on *a type $Q$ if there exists a constructor $c : \cdots \to P \to \cdots \to Q$ in $\Sigma$.*

*We take "*depends on*" relation to be the reflective transitive closure of "*directly depends on*". That is, we say that a type $P$* depends on *a type $Q$ if there exists a chain of dependencies $R_1, R_2, \ldots, R_n$ such that $R_i$* directly depends on *$R_{i+1}$, $R_1 = P$, and $R_n = Q$.*

**Theorem 3.5.2** (Depth Weakening)**.** *When $M$ does not involve constructors of priority $n$, or when $P$ does not depend on a type with priority $n$, if $\vdash^{D}_{\Sigma} M \Leftarrow P$, then for all $k$, $\vdash^{D[n \mapsto k]}_{\Sigma} M \Leftarrow P$.*

*Proof.* Directly by induction on the given derivation. □

**Theorem 3.5.3** (Depth Strengthening)**.** *When $M$ does not involve constructors of priority $n$, or when $P$ does not depend on a type with priority $n$, if for some $k$, $\vdash_\Sigma^{D[n\mapsto k]} M \Leftarrow P$, then $\vdash_\Sigma^D M \Leftarrow P$.*

*Proof.* Directly by induction on the given derivation. □

**Theorem 3.5.4** (Depth Shift)**.** *If $\vdash_\Sigma^D M \Leftarrow A$, then $\vdash_\Sigma^{D'} M \Leftarrow A$, where $D'(n+2k) \simeq D(n)$ for all natural numbers $n$, and some integer $k$, such that $n + 2k \geq 0$.*

*Proof.* Directly by induction on the given derivation. □

## 3.6 Examples

In this section, we present a few examples that illustrate the use of **SemTT** to encode some infinite structures.

### 3.6.1 Conatural Numbers

Conats can be easily defined in a signature $\Sigma =$

$\mathit{conat} \;:\; \mathit{cotype}^0.$

$\mathit{cozero} \;:\; \mathit{conat}.$

$\mathit{cosucc} \;:\; \mathit{conat} \to \mathit{conat}.$

The infinity $\omega = \mathit{cosucc}\,(\mathit{cosucc}\,\dots)$ maybe checked against *conat* as follows:

$$\dfrac{\dfrac{}{\vdash_\Sigma^0 \omega \Leftarrow \mathit{conat}} \quad \{\dfrac{\mathit{cosucc} : \mathit{conat} \to \mathit{conat} \qquad \overset{\dots\text{until we reach } 0}{\vdash_\Sigma^{[l-1]} \omega \Leftarrow \mathit{conat}}}{\vdash_\Sigma^{[l]} \omega = \mathit{cosucc}\,\omega \Leftarrow \mathit{conat}}\}_{l\in\mathbb{N}_{\geq 1}}}{\forall k.\, \vdash_\Sigma^{[k]} \omega \Leftarrow \mathit{conat}}}{\vdash_\Sigma \omega \Leftarrow \mathit{conat}}$$

The canonical terms a type $A$ are terms $M$ where $\vdash_\Sigma M \Leftarrow A$.

**Lemma 3.6.1** (Canonical Forms of Conats)**.** *If $k > 0$ and $\vdash_\Sigma^{[k]} M \Leftarrow \mathit{conat}$, then either $M = \mathit{cozero}$ or $M = \mathit{cosucc}\,M'$ and $\vdash_\Sigma^{[k-1]} M' \Leftarrow \mathit{conat}$.*

*Proof.* $M$ must be of the form $c\,M_1\,\dots\,M_n$, where $c : P_1 \to \cdots \to P_n \to \mathit{conat} \in \Sigma$. We only have two choices for $c$, $c = \mathit{cozero}$ and $n = 0$ or $c = \mathit{cosucc}$ and $n = 1$.

In the first case we have $M = \mathit{cozero}$, and in the second case we have $M = \mathit{cosucc}M'$ and $\vdash_\Sigma^{[k-1]} M' \Leftarrow \mathit{conat}$.

□

**Theorem 3.6.2.** *There is a compositional bijection between conat and the canonical forms of type conat.*

To facilitate the proof of this theorem, we first need to state it precisely.

$\Gamma \vdash_\Sigma M \Leftarrow \mathit{conat}$ iff $\forall k.\Gamma \vdash_\Sigma^{[k]} M \Leftarrow \mathit{conat}$.

Informally we say

- anything is a conat of depth 0

- $Z$ is a conat of depth $k+1$

- $S(X)$ is a conat of depth $k+1$ if $X$ is a conat of depth $k$.

We need to define the association from conat to $LF$ term by the following predicate $(\gg_k)$:

- $X \gg_0 M$ always

- $Z \gg_{k+1} \mathit{cozero}$ always

- $S(X) \gg_{k+1} \mathit{cosucc}(Y)$ iff $X \gg_k Y$

We say that $X \gg M$ iff $\forall k.X \gg_k M$.

**Theorem 3.6.3.** *(1) If $X \gg M$, then $X$ is a conat and $\vdash_\Sigma M \Leftarrow \mathit{conat}$*

*(2) If $\vdash_\Sigma M \Leftarrow \mathit{conat}$ there exists unique $X$ such that and $X \gg M$*

*(3) If $X$ is a conat, there exists unique $M$ such that and $X \gg M$.*

*Proof.* The theorem follows from the next one, where the depth index is made explicit. □

**Theorem 3.6.4.** *For all $k$,*

*(1) If $X \gg_k M$, then $X$ is a conat up to depth $k$ and $\vdash_\Sigma^{[k]} M \Leftarrow \mathit{conat}$.*

*(2) If $\vdash_\Sigma^{[k]} M \Leftarrow \mathit{conat}$, there exists unique conat $X$ up to depth $k$ such that and $X \gg_k M$*

*(3) If $X$ is a conat up to depth $k$, there exists unique term $M$ up to depth $k$ such that and $X \gg_k M$.*

*Proof.* By induction on $k$,

Case $k = 0$,

(1) any $X$ is a conat up to depth 0 and any $M$ satisfies $\vdash_\Sigma^{[0]} M \Leftarrow \mathit{conat}$.

(2) given $M$, any $X$ satisfies $X \gg_0 M$, and any two conats are equal conats up to depth 0.

(3) given $X$, any $M$ satisfies $X \gg_0 M$, and any two terms are equal terms up to depth 0.

Case $k > 0$,

(1) Suppose $X \gg_k M$,

we have either (a) $X = Z$ and $M = \mathit{cozero}$ or (b) $X = S(X')$ and $M = \mathit{cosucc}(M')$, where $X' \gg_{k-1} M'$.

If it is former (a), then $Z$ is a conat and $\vdash_\Sigma^{[k]} \mathit{cozero} \Leftarrow \mathit{conat}$.

If it is latter (b), then by IH $X'$ is a conat of depth $k-1$ and $\vdash_\Sigma^{[k-1]} M' \Leftarrow \mathit{conat}$.

Then by definition $S(X')$ is a conat of depth $k$ and by rule $\vdash_\Sigma^{[k]} \mathit{cosucc}(M') \Leftarrow \mathit{conat}$.

(2) By the canonical forms lemma (3.6.1), either $M = \mathit{cozero}$ or $M = \mathit{cosucc}\, M'$ where $\vdash_\Sigma^{[k-1]} M' \Leftarrow \mathit{conat}$.

Case (a) $M = \mathit{cozero}$, let $X$ be $Z$, then we have $Z \gg_k \mathit{cozero}$, and if $X' \gg_k \mathit{cozero}$, we have $X' = Z$ by cases.

Case (b) $M = \mathit{cosucc}\, M'$, and $\vdash_\Sigma^{[k-1]} M' \Leftarrow \mathit{conat}$. By IH there exists $X' \gg_{k-1} M'$ such that $X'$ is the unique conat up to depth $k-1$. Let $X$ be $S(X')$. $X$ is a conat of depth $k$.

Suppose $Y \gg_k M$ is also a conat up to depth $k$, then by case analysis $Y = S(Y')$, where $Y' \gg_{k-1} M'$, since $X'$ is unique $X' = Y'$ and $Y = X$.

(3) Given $X$ up to depth $k$, either $X = Z$ or $X = S(X')$ and $X'$ is a conat up to depth $k-1$. In the former case, $M = \mathit{cozero}$ is the unique term. In the latter case, by IH, there exists unique $M'$ such that $X' \gg_{k-1} M'$. Let $M = \mathit{cosucc}(M')$, and we verify that $M$ is the unique term such that $X \gg_k M$. □

### 3.6.2 Infinite Streams of Natural Numbers with Finite Paddings

A stream of natural numbers with finite paddings in between can be defined in a signature $\Sigma =$

| | | |
|---|---|---|
| $\mathit{pstream}$ | : | $\mathit{cotype}^0$. |
| $\mathit{padding}$ | : | $\mathit{type}^1$. |
| $\mathit{nat}$ | : | $\mathit{type}^3$. |
| $\mathit{zero}$ | : | $\mathit{nat}$. |
| $\mathit{succ}$ | : | $\mathit{nat} \to \mathit{nat}$. |
| $\mathit{stream\_cons}$ | : | $\mathit{nat} \to \mathit{padding} \to \mathit{pstream}$. |
| $\mathit{pad}$ | : | $\mathit{padding} \to \mathit{padding}$. |
| $\mathit{next}$ | : | $\mathit{pstream} \to \mathit{padding}$. |

To satisfy the validity condition, the priority of $\mathit{pstream}$ should be higher than the priority of $\mathit{padding}$. Lower priority numbers indicate higher priority.

We write the natural number 0 to mean $\mathit{zero}$, 1 to mean $\mathit{succ}\, \mathit{zero}$ and 2 to mean $\mathit{succ}\, (\mathit{succ}\, \mathit{zero})$. We present an example of a stream of 2's with two paddings in between. The stream $S$ satisfying

$$S = \mathit{stream\_cons}\, 2\, (\mathit{pad}\, (\mathit{pad}\, (\mathit{next}\, S)))$$

is an element of $\mathit{pstream}$.

**Lemma 3.6.5** (Canonical Forms of *pstream* and *padding*)**.** *For all $j$, for $k > 0$, for some $l$,*

*(1) if $\vdash_\Sigma {}^{[k,j,-,l]} M \Leftarrow pstream$, then $M = stream\_cons\, M_1\, M_2$ where $\vdash_\Sigma {}^{[k-1,j',-,l']} M_1 \Leftarrow nat$, and $\vdash_\Sigma {}^{[k-1,j',-,l']} M_2 \Leftarrow padding$, for some $j'$, $l'$.*

*(2) if $\vdash_\Sigma {}^{[j,k,-,l]} M \Leftarrow padding$, then either (a) $M = pad\, M_1$ where $\vdash_\Sigma {}^{[j,k-1,-,l]} M_1 \Leftarrow padding$, or (b) $M = next\, M_1$ where $\vdash_\Sigma {}^{[j,k-1,-,l]} M_1 \Leftarrow pstream$.*

*Proof.* Directly by induction over given derivation. □

**Theorem 3.6.6** (Adequacy of Representation)**.** *There is a bijection between stream of natural numbers with finite padding in between and canonical forms of pstream.*

*Proof.* Informally, we abbreviate a stream of natural numbers using the following notation: $N_1, P_{11}, P_{12}, \ldots, P_{1p_1}, N_2, P_{21}, P_{22}, \ldots, P_{2p_2}, N_3, \ldots,$ where $N_i$ represents natural numbers for all natural numbers $i$, and $P_{ij}$ represents paddings, with $p_i$ being a natural number denote the length of the padding.

And this padding corresponds to the canonical term:
$stream\_cons\ulcorner N_1 \urcorner\, (pad^{p_1}\, (next\, (stream\_cons\ulcorner N_2 \urcorner\, (pad^{p_2}\, (next\, (stream\_cons\ulcorner N_3 \urcorner\, \ldots)))))))$
Formally, streams are defined as follows,

1. A stream $S$ of depth 0 is anything.
2. A stream $S$ of depth $k+1$ is of the form $N, P_1, P_2, \ldots, P_n, S'$ where $S'$ is a stream of depth $k$, and $P_i$'s are paddings.

With the following encoding relation: (as usual, $\gg_0$ holds always)

- stream $N, P_1, \ldots, P_n, S' \gg_{k+1} stream\_cons\ulcorner N \urcorner\, (pad^n\, (next\, M_{S'}))$ where $S' \gg_k M_{S'}$

Then we need to prove the following theorem:

(1) For all $j$, for some $l$, for all $k$, if $S$ a stream of depth $k$, then there exists unique term $M$ up to depth $k$ such that $\vdash_\Sigma {}^{[k,j,-,l]} M \Leftarrow pstream$ and $S \gg_k M$.

(2) For all $k$, for all $j$, for some $l$, if $\vdash_\Sigma {}^{[k,j,-,l]} M \Leftarrow pstream$, then there exists unique stream $S$ up to depth $k$ such that and $S \gg_k M$.

We prove all claims simultaneously by induction on $k$.

Base case $k = 0$, the conclusions are trivially satisfied.

Case $k > 0$,

(1) Suppose $S$ is a stream of depth $k$, $S = N, P_1, \ldots, P_n, S'$ where $S'$ is a stream of depth $k-1$ and $N$ is a natural number. By IH, there exists unique $M_{S'}$ up to depth $k-1$ such that $S' \gg_{k-1} M_{S'}$, and $\vdash_\Sigma {}^{[k-1,j,-,l']} M_{S'} \Leftarrow padding$. Let $M =$

$stream\_cons \ulcorner N \urcorner (pad^n\,(next\, M_{S'}))$ where $\ulcorner N \urcorner$ is the adequate encoding of $N$, and by the rules (shown below) $\vdash_\Sigma^{[k,j,-,l]} M \Leftarrow stream$, and by definition $S \gg_k M$. Uniqueness follows by cases on $S \gg_k M$.

1. $\vdash_\Sigma^{[k-1,j,-,l']} M_{S'} \Leftarrow padding$ IH (Assumption)
2. $\vdash_\Sigma^{[k-1,j+1,-,l']} next\, M_{S'} \Leftarrow padding$ by rule cM1
3. $\vdash_\Sigma^{[k-1,j+1+n,-,l']} pad^n\,(next\, M_{S'}) \Leftarrow padding$ by the rule cM1 applied $n$ times
4. $\vdash_\Sigma^{[k-1,j,-,l']} \ulcorner N \urcorner \Leftarrow nat$ adequacy for $nat$ and weakening
5. $\vdash_\Sigma^{[k,j,-,l]} stream\_cons \ulcorner N \urcorner (pad^n\,(next\, M_{S'})) \Leftarrow padding$ by rule cM1

(2) If $\vdash_\Sigma {}^{[k,j,-,l]} M \Leftarrow pstream$, by the canonical form lemma (3.6.5), $M = stream\_cons\, M_1\, M_2$, where $\vdash_\Sigma {}^{[k-1,j',-,l']} M_1 \Leftarrow nat$ and $\vdash_\Sigma {}^{[k-1,j',-,l']} M_2 \Leftarrow padding$ for some $j'$, $l$. By adequacy for nat, we have a natural number $N$ such that $\ulcorner N \urcorner = M_1$.

For $\vdash_\Sigma^{[k-1,j',-,l']} M_2 \Leftarrow padding$, we do an induction on $j'$ to show that for some $n$, $M_2 = pad^n\,(next\, M')$ where $\vdash_\Sigma^{[k-1,j'-1-n,-,l']} M' \Leftarrow pstream$.

Case $j' = 0$. This case is not possible because the type $padding$ is an inductive type.

Case $j' > 0$. By canonical forms lemma 3.6.5, either (a) $M_2 = next\, M'$ where $\vdash_\Sigma^{[k-1,j'-1,-,l']} M' \Leftarrow pstream$, or (b) $M_2 = pad\, M'$ where $\vdash_\Sigma^{[k-1,j'-1,-,l']} M' \Leftarrow padding$

If it is former (a), then we have $n = 0$ and $M_2 = next\, M'$ where $\vdash_\Sigma^{[k-1,j'-1,-,l']} M' \Leftarrow pstream$.

If it is latter (b), then by IH we have for some $n'$, $M' = pad^{n'}\,(next\, M')$ where $\vdash_\Sigma^{[k-1,(j'-1)-1-n',-,l']} M' \Leftarrow pstream$, we let $n = n' + 1$, and we have $M_2 = pad\,(M') = pad^n\,(next\, M')$, and $\vdash_\Sigma^{[k-1,j'-1-n,-,l']} M' \Leftarrow pstream$.

Then by IH, there exists a unique stream $S'$ up to depth $k-1$ such that $S' \gg_{k-1} M'$, then by rule, we let $S = N, P_1, \ldots, P_n, S'$, and we verify that $S \gg_k M$, where $M$ is $stream\_cons \ulcorner N \urcorner (pad^n\,(next\, M_2))$.

The uniqueness follows by cases on $S \gg_k M$.

□

# Chapter 4

# Semantic Type Theory with Simple Types and Rational Terms

In this chapter, we present $\mathbf{SemTT_R}$, the system where the terms of **SemTT** are restricted to rational terms. The system $\mathbf{SemTT_R}$ mimics circular proof systems in the literature [BS11].

## 4.1 Preliminaries

A term is rational if the set of its subterms is finite [Cou83]. Formally, we may represent a set of subterms using a system of equation. In the following sections, I shall informally write $T = M(T)$ for a rational term $T = M\,(M\,(M\,\ldots))$. For instance, I shall write $\infty = \mathit{cosucc}\,\infty$ for $\infty = \mathit{cosucc}\,(\mathit{cosucc}\,(\mathit{cosucc}\,\ldots))$.

We now allow parameters to appear in the context. For example, given symbols $k$ and $j$, $[k, j]$ is a depth context. The ordering on the depth context naturally extends to depth contexts with symbols. For instance, we have $[k-3, j] < [k-1, j]$ on principal index 0.

## 4.2 Type Checking Rules

The signature and judgment forms of $\mathbf{SemTT_R}$ are the same as that of **SemTT**, except that terms are required to be rational, and that depth contexts range over symbols besides natural numbers.

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | $::=$ | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Kind | $K$ | $::=$ | $\mathsf{type}^{\mathbb{N}} \mid \mathsf{cotype}^{\mathbb{N}} \mid P \to K$ |
| Canonical Type Families | $A$ | $::=$ | $P \mid P \to A$ |
| Atomic Type Families | $P$ | $::=$ | $a \mid P\, M$ |
| Terms (Required to be rational) | $M_{(k+1)}$ | $::=$ | $c\, M_{1(k)} \ldots M_{n(k)}$ |

We simultaneously define the following judgments

| | |
|---|---|
| $\Sigma\ \mathsf{ok}$ | Signature $\Sigma$ is type correct |
| $\vdash_\Sigma K \Leftarrow \mathsf{kind}$ | Kind $K$ is a valid kind |
| $\vdash_\Sigma A \Leftarrow \mathsf{type}$ | Type $A$ is a canonical type family |
| $\vdash_\Sigma P \Rightarrow K$ | Atomic type family $P$ synthesizes kind $K$ |
| $\vdash_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ |
| $\vdash^D_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ in the depth context $D$ |

$\boxed{\Sigma\ \mathsf{ok}}$ : $\Sigma$ is a valid signature.

$$\frac{}{\cdot\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Sigma, a : K\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\Sigma, c : A\ \mathsf{ok}}$$

$\boxed{\vdash_\Sigma K \Leftarrow \mathsf{kind}}$ : $K$ is a valid kind, assuming that $\Sigma\ \mathsf{ok}$.

$$\frac{n \notin pOccurs(\Sigma)}{\vdash_\Sigma \mathsf{type}^n \Leftarrow \mathsf{kind}} (n \text{ is an odd natural number})$$

$$\frac{n \notin pOccurs(\Sigma)}{\vdash_\Sigma \mathsf{cotype}^n \Leftarrow \mathsf{kind}} (n \text{ is an even natural number})$$

$$\frac{\vdash_\Sigma P \Rightarrow K' \qquad K' = \mathsf{type} \,/\, \mathsf{cotype} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\vdash_\Sigma P \to K \Leftarrow \mathsf{kind}}$$

$\boxed{\vdash_\Sigma A \Leftarrow \mathsf{type}}$ : $A$ is a good type, assuming that $\Sigma\ \mathsf{ok}$.

$$\frac{\vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\vdash_\Sigma P \to A \Leftarrow \mathsf{type}}$$

$$\frac{\vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n}{\vdash_\Sigma P \Leftarrow \mathsf{type}}$$

$\boxed{\vdash_\Sigma P \Rightarrow K}$ : $P$ has kind $K$, assuming that $\Sigma$ ok.

$$\frac{a : K \in \Sigma}{\vdash_\Sigma a \Rightarrow K} \qquad\qquad \frac{\vdash_\Sigma P \Rightarrow P' \to K \qquad \vdash_\Sigma M \Leftarrow P'}{\vdash_\Sigma P\, M \Rightarrow K}$$

$\boxed{\vdash_\Sigma M \Leftarrow P}$ is defined to be:

$$\vdash_\Sigma^{[k_0,k_1,k_2,\ldots,k_n]} M \Leftarrow P$$

where

(1) $k_i$ is a fresh symbol $k_i$ if $i$ is even

(2) $k_i$ is a concrete number if $i$ is odd

(3) $n = maxP(\Sigma)$

$\boxed{\vdash_\Sigma^D M \Leftarrow P}$ : $M$ checks against type $P$ in the depth indexing context $D$, assuming that $\Sigma$ ok.

$$\frac{c : P_1 \to P_2 \to \cdots \to P_n \to P' \in \Sigma \qquad P = P' \qquad \forall 1 \le i \le n.\, \vdash_\Sigma^{D[p(P)\mapsto k]} M_i \Leftarrow P_i}{\vdash_\Sigma^{D[p(P)\mapsto k+1]} c\, M_1\, \ldots\, M_n \Leftarrow P}\, cM$$

$$\frac{}{\vdash_\Sigma^D M \Leftarrow P}\, bud$$

The derivation $\vdash_\Sigma^D M \Leftarrow P$ may be circular by the bud rule, which points to a prior point in the derivation tree.

When writing down an application of the bud rule, we must explicitly specify which judgment prior in the derivation that the bud rule "points to". We say that the application of the rule *bud* on judgment $\vdash_\Sigma^D M \Leftarrow A$ is valid if the "pointed to" judgment prior in the derivation tree is of the form $\vdash_\Sigma^{D'} M \Leftarrow A$, and $D < D'$ on an even principal index. Due to the structure of the proof system, it must be the case that $D[p(P)] = k_i - n$, $D'[p(P)] = k_i - n'$ and $n > n'$, where $n, n'$ are concrete natural numbers. For example, when checking that $\omega = cosucc\,(cosucc\,(\ldots))$ has type *conat* in the following signature,

$conat$ : $cotype^0$.

$cozero$ : $conat$.

$cosucc$ : $conat \to conat$.

The following proof demonstrates a valid application of the bud rule.

$$\frac{cosucc : conat \to conat \qquad \dfrac{(*)}{\vdash_\Sigma^{[l-1]} \omega \Leftarrow conat}\, bud}{(*) \vdash_\Sigma^{[l]} \omega = cosucc\, \omega \Leftarrow conat}$$

The *bud* rule "points" to the judgment at $(*)$, which is a judgment prior in the derivation. The application of the *bud* rule is valid because $[l-1] < [l]$ on principal index 0, which is an even number.

## 4.3 Metatheorems

**Theorem 4.3.1** (Soundness)**.** *If* $\vdash_\Sigma M \Leftarrow A$ *in* $\mathbf{SemTT_R}$*, then* $\vdash_\Sigma M \Leftarrow A$ *in* $\mathbf{SemTT}$.

*Proof.* To show the conclusion, first we perform quantifier instantiations.

For every instantiation of the universally quantifications, we construct a symbol substitution where the symbol $k_i$ for each even $i$ is mapped to a concrete value (given by the universal quantifier). The existential quantifiers at the topmost level can be directly provided the corresponding value of $k_i$ for odd $i$.

Then we proceed to map the proof of $\vdash^D_\Sigma M \Leftarrow P$ in $\mathbf{SemTT_R}$ to a proof of $\vdash^D_\Sigma M \Leftarrow P$ in $\mathbf{SemTT}$ by induction on the proof.

Case rule $cM$. Either $p(P)$ is even or odd.

When $p(P)$ is odd, we translate the rule to $cM1$ and proceed to translate the premises.

When $p(P)$ is even, we check whether the symbol substitution yields 0 for $p(P)$. If so, we apply the $cM0$ rule. If not, we apply the $cM1$ rule and instantiate a new variable for all odd index positions that have lower priority. The values of those variables will be instantiated either at the translation of the bud rule, or can be safely set to zero.

Case rule *bud*. We "jump" to the derivation (before the translation) where *bud* points to and instantiate all unknown variables (created during the translation of the rule $cM$) by unifying the expressions at each index.

We check that the translation is productive in the sense that each application of the bud must be preceded with some application of the $cM$ rule because the bud rule must point a depth context that is strictly larger.

□

## 4.4 Examples

We illustrate how the system $\mathbf{SemTT_R}$ provides a natural way of writing down type checking proofs of rational objects in $\mathbf{SemTT}$. Furthermore, we illustrate how priorities affect the interpretation of the underlying signature.

### 4.4.1 A Stream of Natural Numbers

When the stream has a higher priority than natural numbers:

$\Sigma =$

$$\begin{array}{lcl} \mathit{stream} & : & \mathit{cotype}^0. \\ \mathit{nat} & : & \mathit{type}^1. \\ \mathit{zero} & : & \mathit{nat}. \\ \mathit{succ} & : & \mathit{nat} \to \mathit{nat}. \\ \mathit{s\_cons} & : & \mathit{nat} \to \mathit{stream} \to \mathit{stream}. \end{array}$$

To prove that the infinite term $S$ satisfying the equation $S = \mathit{s\_cons}\,2\,(\mathit{s\_cons}\,3\,S)$ has type $\mathit{stream}$,

$$\dfrac{\dfrac{\dfrac{}{\vdash^{[k_0-1,2]} 0 \Leftarrow \mathit{nat}}}{\vdash^{[k_0-1,3]} 1 \Leftarrow \mathit{nat}}}{\vdash^{[k_0-1,4]} 2 \Leftarrow \mathit{nat}} \quad \dfrac{\dfrac{\dfrac{}{\vdash^{[k_0-2,1]} 0 \Leftarrow \mathit{nat}}}{\vdash^{[k_0-2,2]} 1 \Leftarrow \mathit{nat}}}{\vdash^{[k_0-2,3]} 2 \Leftarrow \mathit{nat}} \quad \dfrac{(*)}{\vdash^{[k_0-2,4]} S \Leftarrow \mathit{stream}}}{\dfrac{\vdash^{[k_0-2,4]} 3 \Leftarrow \mathit{stream} \qquad \vdash^{[k_0-1,4]} \mathit{s\_cons}\,3\,S \Leftarrow \mathit{stream}}{(*) \vdash^{[k_0,4]} S = \mathit{s\_cons}\,2\,(\mathit{s\_cons}\,3\,S) \Leftarrow \mathit{stream}}}\ \mathit{bud}$$

It could also be the case that streams have lower priorities than natural numbers.

$\Sigma =$

$$\begin{array}{lcl} \mathit{stream} & : & \mathit{cotype}^2. \\ \mathit{nat} & : & \mathit{type}^1. \\ \mathit{zero} & : & \mathit{nat}. \\ \mathit{succ} & : & \mathit{nat} \to \mathit{nat}. \\ \mathit{s\_cons} & : & \mathit{nat} \to \mathit{stream} \to \mathit{stream}. \end{array}$$

$$\dfrac{\dfrac{\dfrac{}{\vdash^{[-,2,k_2-1]} 0 \Leftarrow \mathit{nat}}}{\vdash^{[-,3,k_2-1]} 1 \Leftarrow \mathit{nat}}}{\vdash^{[-,4,k_2-1]} 2 \Leftarrow \mathit{nat}} \quad \dfrac{\dfrac{\dfrac{}{\vdash^{[-,1,k_2-2]} 0 \Leftarrow \mathit{nat}}}{\vdash^{[-,2,k_2-2]} 1 \Leftarrow \mathit{nat}}}{\vdash^{[-,3,k_2-2]} 2 \Leftarrow \mathit{nat}} \quad \dfrac{(*)}{\vdash^{[-,4,k_2-2]} S \Leftarrow \mathit{stream}}}{\dfrac{\vdash^{[-,4,k_2-2]} 3 \Leftarrow \mathit{stream} \qquad \vdash^{[-,4,k_2-1]} \mathit{s\_cons}\,3\,S \Leftarrow \mathit{stream}}{(*) \vdash^{[-,4,k_2]} S = \mathit{s\_cons}\,2\,(\mathit{s\_cons}\,3\,S) \Leftarrow \mathit{stream}}}\ \mathit{bud}$$

### 4.4.2 Four Kinds of Streams

Assume the auxiliary definition:

$P$ : $type^5$.
$p$ : $P$.
$nat$ : $type^3$.
$zero$ : $nat$.
$succ$ : $nat \to nat$.

1. Left-fair streams, written in the fixed point notation of

$$\nu X.\mu Y.\mathbb{N} \otimes X \oplus P \otimes Y$$

Left-fair streams could be defined in the following signature ($X$ stands for stream and $Y$ stands for padding)

$pstream$ : $cotype^0$.
$padding$ : $type^1$.
$next$ : $padding \to pstream$.
$num$ : $nat \to pstream \to padding$.
$pad$ : $P \to padding \to padding$.

Let term $S$ represent a stream of 2's with one padding in between, i.e., $S$ satisfies the equation $S = next\,(pad\,p\,(num\,2\,S))$. We can show that $S$ has type $pstream$. Here's the derivation.

$$\dfrac{\dfrac{\vdash^{[k_0-1,1,k_2,1,k_4,3]} p \Leftarrow P \qquad \dfrac{\dfrac{\dfrac{}{\vdash^{[k_0-1,0,k_2,1,k_4,1]} 0 \Leftarrow nat}}{\vdash^{[k_0-1,0,k_2,1,k_4,2]} 1 \Leftarrow nat}}{\vdash^{[k_0-1,0,k_2,1,k_4,3]} 2 \Leftarrow nat} \quad \dfrac{(*)}{\vdash^{[k_0-1,0,k_2,1,k_4,3]} S \Leftarrow pstream}\,bud}{\vdash^{[k_0-1,1,k_2,1,k_4,3]} num\,2\,S \Leftarrow padding}}{\vdash^{[k_0-1,2,k_2,1,k_4,3]} pad\,p\,(num\,2\,S) \Leftarrow padding}}{(*) \vdash^{[k_0,2,k_2,1,k_4,3]} S = next\,(pad\,p\,(num\,2\,S)) \Leftarrow pstream}$$

2. Empty type.

In the previous example, the priority of induction and coinduction may not be switched. If the stream has a lower type than that of the paddings, the resulting type would be empty.

$pstream$ : $cotype^2$.
$padding$ : $type^1$.
$next$ : $padding \to pstream$.
$num$ : $nat \to pstream \to padding$.
$pad$ : $P \to padding \to padding$.

A stream of 2's with one padding in between, i.e., $S$ satisfies the equation $S = next\,(pad\,p\,(num\,2\,S))$ can no longer be typed:

$$\dfrac{\dfrac{}{\vdash^{[k_0,1,k_2-1,1,k_4,3]} p \Leftarrow P} \qquad \dfrac{\dfrac{\dfrac{}{\vdash^{[k_0,0,k_2-1,1,k_4,1]} 0 \Leftarrow nat}}{\vdash^{[k_0,0,k_2-1,1,k_4,2]} 1 \Leftarrow nat}}{\vdash^{[k_0,0,k_2-1,1,k_4,3]} 2 \Leftarrow nat} \quad (*)\text{Cannot apply the bud rule!} \quad \vdash^{[k_0,0,k_2-1,1,k_4,3]} S \Leftarrow padding}{\vdash^{[k_0,1,k_2-1,1,k_4,3]} num\,2\,S \Leftarrow padding}}{\vdash^{[k_0,2,k_2-1,1,k_4,3]} pad\,p\,(num\,2\,S) \Leftarrow padding}}{(*) \vdash^{[k_0,2,k_2,1,k_4,3]} S = next\,(pad\,p\,(num\,2\,S)) \Leftarrow pstream}$$

We cannot apply the bud rule at $(*)$ because the contexts $[k_0, 0, k_2, 1, k_4, 3] < [k_0, 2, k_2 - 1, 1, k_4, 3]$ on principal index 1, which is an odd index.

3. Stream of freely mixed numbers and paddings

$$\mu X.\nu Y.\mathbb{N} \otimes X \oplus P \otimes Y$$

When the induction has the priority over coinduction, the above expression corresponds to the following signature ($X$ stands for pstream and $Y$ stands for padding)

$$\begin{array}{lcl} pstream & : & type^1. \\ padding & : & cotype^0. \\ next & : & padding \to pstream. \\ num & : & nat \to pstream \to padding. \\ pad & : & P \to padding \to padding. \end{array}$$

The term presenting a stream of 2's with one padding in between, i.e., $S$ satisfying $S = next\,(pad\,p\,(num\,2\,S))$, can be type-checked:

$$\dfrac{\dfrac{}{\vdash^{[k_0-1,1,k_2,1,k_4,3]} p \Leftarrow P} \qquad \dfrac{\dfrac{\dfrac{}{\vdash^{[k_0-2,1,k_2,1,k_4,1]} 0 \Leftarrow nat}}{\vdash^{[k_0-2,1,k_2,1,k_4,2]} 1 \Leftarrow nat}}{\vdash^{[k_0-2,1,k_2,1,k_4,3]} 2 \Leftarrow nat} \quad \dfrac{(*)}{\vdash^{[k_0-2,1,k_2,1,k_4,3]} S \Leftarrow padding}\,bud}{\vdash^{[k_0-1,1,k_2,1,k_4,3]} num\,2\,S \Leftarrow padding}}{\vdash^{[k_0,1,k_2,1,k_4,3]} pad\,p\,(num\,2\,S) \Leftarrow padding}}{(*) \vdash^{[k_0,2,k_2,1,k_4,3]} S = next\,(pad\,p\,(num\,2\,S)) \Leftarrow pstream}$$

4. With the same expression as the previous case,

$$\mu X.\nu Y.\mathbb{N} \otimes X \oplus P \otimes Y$$

When priorities are switched such that induction has a higher priority than coinduction, the expression corresponds to streams with finitely many numbers and then only paddings. Therefore, the previous stream is no longer well-typed.

$$
\begin{array}{lcl}
\mathit{pstream} & : & \mathit{type}^1. \\
\mathit{padding} & : & \mathit{cotype}^2. \\
\mathit{next} & : & \mathit{padding} \to \mathit{pstream}. \\
\mathit{num} & : & \mathit{nat} \to \mathit{pstream} \to \mathit{padding}. \\
\mathit{pad} & : & P \to \mathit{padding} \to \mathit{padding}.
\end{array}
$$

$$
\dfrac{\dfrac{}{\vdash^{[k_0,1,k_2-1,1,k_4,1]} p \Leftarrow P} \qquad \dfrac{\dfrac{\dfrac{}{\vdash^{[k_0,1,k_2-2,1,k_4,1]} 0 \Leftarrow \mathit{nat}}}{\vdash^{[k_0,1,k_2-2,1,k_4,2]} 1 \Leftarrow \mathit{nat}} \quad (*)\text{Cannot apply the bud rule!}}{\vdash^{[k_0,1,k_2-2,1,k_4,3]} 2 \Leftarrow \mathit{nat}} \quad \vdash^{[k_0,1,k_2-2,1,k_4,3]} S \Leftarrow \mathit{padding}}{\vdash^{[k_0,1,k_2-1,1,k_4,3]} \mathit{num}\,2\,S \Leftarrow \mathit{padding}}}{\vdash^{[k_0,1,k_2,1,k_4,3]} \mathit{pad}\,p\,(\mathit{num}\,2\,S) \Leftarrow \mathit{padding}}}{(*) \vdash^{[k_0,2,k_2,1,k_4,3]} S = \mathit{next}\,(\mathit{pad}\,p\,(\mathit{num}\,2\,S)) \Leftarrow \mathit{pstream}}
$$

We cannot close the derivation by applying the bud rule because the depth context $[k_0, 1, k_2 - 2, 1, k_4, 3] < [k_0, 2, k_2, 1, k_4, 3]$ on principal index 1, an odd principal index. However, streams of finitely many numbers followed by only paddings are well typed. An example would be the term representing a stream where pads occur infinitely after an occurrence of 2, i.e., $\mathit{next}\,(\mathit{num}\,2\,(\mathit{next}\,M)))$, where $M$ is a term satisfying the equation $M = \mathit{pad}\,p\,M$.

$$
\dfrac{\dfrac{\dfrac{}{\vdash^{[k_0,1,k_2-1,1,k_4,1]} 0 \Leftarrow \mathit{nat}}}{\vdash^{[k_0,1,k_2-1,1,k_4,2]} 1 \Leftarrow \mathit{nat}} \qquad \dfrac{\dfrac{}{\vdash^{[k_0,0,k_2-2,1,k_4,3]} p \Leftarrow P} \qquad \dfrac{(*)}{\vdash^{[k_0,0,k_2-2,1,k_4,3]} M \Leftarrow \mathit{padding}}}{(*) \vdash^{[k_0,0,k_2-1,1,k_4,3]} \mathit{pad}\,p\,M \Leftarrow \mathit{padding}}}{\vdash^{[k_0,1,k_2-1,1,k_4,3]} \mathit{next}\,(\mathit{pad}\,p\,M) \Leftarrow \mathit{pstream}}}{\vdash^{[k_0,1,k_2,1,k_4,3]} \mathit{num}\,2\,(\mathit{next}\,M)) \Leftarrow \mathit{padding}}}{\vdash^{[k_0,2,k_2,1,k_4,3]} \mathit{next}\,(\mathit{num}\,2\,(\mathit{next}\,M)) \Leftarrow \mathit{pstream}}\ \mathit{bud}
$$

# Chapter 5

# Syntactic Type Theory with Simple Types

In this chapter, we introduce **SynTT**, an easily implementable type theory for $\mathbf{SemTT_R}$. Type checking is decidable in this system. We further prove the soundness and completeness of this system with respect to $\mathbf{SemTT_R}$.

## 5.1 Function Definitions by Depth Indexing

Functions could be defined using the same methodology of depth indexing. We may interpret a function as a binary relation satisfying the function condition: every unique term on the left corresponds to a unique term on the right, and the productivity condition: the right-hand side must be productive. For instance, equality, as a binary relation, is also the identity function.

We could use clauses of the following form when defining a function (one clause for each form of $M$),

$$f_{k+1}(M) =_{k+1} E(f_k(M'))$$

The above expression can be understood to define an inductive binary relation on the set of terms, such that

- any two terms are related at level 0
- $M$ is related to $E(X)$ at level $k+1$ when $M'$ is related to $X$ at level $k$.

That is, the above expression defines a relation $\mathcal{R}$ such that $M\,\mathcal{R}\,M'$ iff $\forall k. M\,\mathcal{R}_k\,M'$, where

$$\frac{}{N\,\mathcal{R}_0\,N'} \qquad\qquad \frac{M'\,\mathcal{R}_k\,X}{M\,\mathcal{R}_{k+1}\,E(X)}$$

When $E$ is guarded (aka productive), the above binary relation satisfies the productivity condition (since $E$ is guarded), and the function condition (since if $M$ relates to $M'$ at every level, then such $M'$ will be unique). Because we intend the relation to be interpreted as a function, we use the functional notation $f_{k+1}(M) =_{k+1} E(f_k(M'))$ when writing out the relation. We say that the function $f$ sends an object $M$ to the unique object $M'$ if $M\,\mathcal{R}\,M'$. An example of this definition would be substitution operations of infinite terms defined in the next section.

**Theorem 5.1.1** (Downward Closure of Functions)**.** $f_{k+1}(M) =_k f_k(M)$

*Proof.* By induction on $k$.

Case $k = 0$, the equality is trivial. When $k > 0$, By definition $f_k(M) =_{k-1} E(f_{k-1}(M'))$, and $f_{k+1}(M) =_{k+1} E(f_k(M'))$.

By the induction hypothesis, $f_k(M') =_{k-1} f_{k-1}(M')$.

Because equality is a congruence and $E$ is guarded, $f_{k+1}(M) =_{k+1} E(f_k(M'))$.

□

## 5.2 Extending Terms with Variables

The terms so far do not have the notion of variables and substitution. In order to develop a practical theory, it is important to consider terms with variables for not-yet-known terms. And therefore, we develop the theory of adding variables to the syntax of the terms.

$$\text{Terms} \quad M_{(k+1)} \quad ::= \quad c\,M_{1(k)}\,\ldots\,M_{n(k)} \mid x$$

With the new extended equality rules:

$\boxed{M =_k M'}$ ($M$ and $M'$ are equal up to depth $k$, defined by induction on $k$)

$$\frac{}{M =_0 M'} \qquad \frac{}{x =_{k+1} x} \qquad \frac{\forall i.\ M_i =_k M_i'}{c\,M_1 \cdots M_n =_{k+1} c\,M_1' \cdots M_n'}$$

Variables must be depth correct when presented as part of a term, and they also enjoy downward closure properties. Substitutions must respect the observation depths of variables. For instance, if $M_{(k)} = c\,x$ then $x$ will have depth $k-1$, and when writing down any substitution instance $[M_{2(j)}/x]M_{(k)}$ we require that $j \geq k-1$. Otherwise, the

substituted object will not have enough observation depth. We denote the maximum depth of variable $x$ in $M$, $max(x \in M)$. For instance, if $M_{(k)} = c\,x\,(d\,x)\,(c\,y)$, $max(x \in M_{(k)}) = k-1$ and $max(y \in M_{(k)}) = k-2$.

We also note that due to the syntax of the terms, variables cannot be applied to arguments, $x\ M'$ is not a well-formed term.

We define $[M'/x]_k M$ first by induction on $k$, then by cases on $M$. That is, we are defining a function of the form $[M'/x](-)$. In order for the substitution to be well-defined, we require the observation depth of $M'$ to be $\geq max(x \in M)$. Moreover, the substituted term $[M'/x]_k M$ will have the same observation depth as $M$, and that depth is equal to $k$. We recall that $[M'/x]_0 M =_0 M''$ for any term $M''$.

$[M'/x]_{k+1} x =_{k+1} M'$

$[M'/x]_{k+1} y =_{k+1} y\ (y \neq x)$

$[M'/x]_{k+1}(c\,M_1\ \ldots\ M_n) =_{k+1} c\,[M'/x]_k M_1\ \ldots\ [M'/x]_k M_n\ (y \neq x)$

**Lemma 5.2.1** (Downward Closure of Substitution)**.** *If $max(x \in M_{(k)}) \leq j \leq k$, then $[N_{(k)}/x]_k M_{(k)} =_k [N_{(j)}/x]_k M_{(k)}$.*

*Proof.* By downward closure, if $N_{(k)}$, then $N_{(j)}$, then by downward closure of the equality, $N_{(k)} =_l N_{(j)}$ for all $l \leq j$. By reflexivity, $M_k =_k M_k$, call this equality proof $\mathcal{D}$. Then we replace all the proofs in $\mathcal{D}$ of the form $x_l =_l x_l$ by appropriate $N_{(k)} =_l N_{(j)}$, and then we get a proof of $[N_{(k)}/x]_k M_{(k)} =_k [N_{(j)}/x]_k M_{(k)}$. □

**Lemma 5.2.2** (Maximum Depth of Variables in an Object)**.** *If $M_{(k)} = c\ \ldots$ and $x \in free(M_{(k)})$, then $max(x \in M_{(k)}) \leq k-1$.*

*Proof.* Observation will decrease observation depth.

Since the observation depth of $M$ is $k$, the observation depth of all arguments is $k-1$, and $x$ is either one of the arguments, in which case it has depth $k-1$, or it occurs as a subterm of one of the arguments, in which case it has depth $\leq k-1$. □

**Lemma 5.2.3.** *If $M_{(k)}$, then $max(x \in M_{(k)}) \leq k$.*

*Proof.* Directly by the definition of observation depth. □

**Lemma 5.2.4** (Equality of Substitution)**.** *If $M_1 =_j M_2$ and $max(x \in M) \leq j$, then $[M_1/x]_k M =_k [M_2/x]_k M$.*

*Proof.* We have $M =_k M$ by reflexivity, whose derivation includes derivation of the form $\dfrac{}{x = x}$. We substitute those derivations with $M_1 =_j M_2$, and the resulting derivation is a derivation of $[M_1/x]_k M =_k [M_2/x]_k M$ □

**Theorem 5.2.5** (Equality Commutes with Substitution)**.** *If* $M_1 =_j M_2$ *and* $M_3 =_k M_4$*, and* $max(max(x \in M_3), max(x \in M_4)) \leq j$*, then* $[M_1/x]_k M_3 =_k [M_2/x]_k M_4$*.*

*Proof.* Given a derivation of $M_3 =_k M_4$, we have subderivations of the form $\dfrac{}{x = x}$. Substitute those rules with the derivation of the $M_1 =_j M_2$. After performing necessary cut down of derivation at depth 0, we get a derivation of $[M_1/x]_k M_3 =_k [M_2/x]_k M_4$.

□

## 5.3 Representation of Terms and Fixed Points

The terms in this language are rational terms given by the following grammar. Again, we disallow application of variables to arguments.

$$\text{Terms} \quad M \quad ::= \quad x \mid c\, M_1 \ \ldots \ M_n \mid \mathsf{fix}\, x.M$$

We implicitly require $M$ to be well-formed by preventing fixed points that aren't productive.

$\boxed{M \ \mathsf{wellformed}}$

$$\frac{}{x \ \mathsf{wellformed}} \qquad \frac{\forall 1 \leq i \leq n. M_i \ \mathsf{wellformed}}{c\, M_1 \ \ldots \ M_n \ \mathsf{wellformed}} \qquad \frac{M \ \mathsf{guarded} \qquad M \ \mathsf{wellformed}}{\mathsf{fix}\, x.M \ \mathsf{wellformed}}$$

$\boxed{M \ \mathsf{guarded}}$

$$\text{(variables are not guarded)} \qquad \frac{}{c\, M_1 \ \ldots \ M_n \ \mathsf{guarded}} \qquad \frac{M \ \mathsf{guarded}}{\mathsf{fix}\, x.M \ \mathsf{guarded}}$$

Now, we define the fixed point expansion from well-formed terms of **SynTT** to terms of **SemTT** (with variables), $exp_k : M^{\mathsf{wellformed}}_{(\mathbf{SynTT})} \to M_{(k)(\mathbf{SemTT})}$

$exp_{k+1}(x) =_{k+1} x_{k+1}$

$exp_{k+1}(c\, M_1 \ \ldots M_n) =_{k+1} c\, exp_k(M_1) \ \ldots \ exp_k(M_n)$

$exp_{k+1}(\mathsf{fix}\, x.M) =_{k+1} [exp_k(\mathsf{fix}\, x.M)/x]_{k+1} exp_{k+1}(M)$

We did not choose to define the last clause as $exp_{k+1}(\mathsf{fix}\, x.M) =_{k+1} exp_k([\mathsf{fix}\, x.M/x]M)$ because we want the resulting object to have observation depth at least $k+1$. We check that the right-hand side of $exp$ is productive because of guardedness. Let $exp$ denotes the function that maps $M$ to $M'$ such that $exp_k(M) =_k M'$ for every $k$.

Through a series of theorems, we prove that substitution commutes with $exp$.

**Lemma 5.3.1** (Downward Closure of *exp*)**.** *If $j \geq k$, then $exp_j(M) =_k exp_k(M)$.*

*Proof.* We show this by showing $exp_{k+1}(M) =_k exp_k(M)$, then the result follows by applying the proof $k - j$ times.

To show $exp_{k+1}(M) =_k exp_k(M)$,

by induction on $k$, $k = 0$ is trivial, $k > 0$, then by cases on $M$,

- $M = x$, then we have $x =_k x$ directly by rule

- $M = c\, M_1 \ldots M_n$, then $exp_{k+1}(M) =_{k+1} c\, exp_k(M_1) \ldots exp_k(M_n)$, by IH on, for all $1 \leq i \leq n$, $exp_k(M_i) =_{k-1} exp_{k-1}(M_i)$, then by the equality rules, $c\, exp_k(M_1) \ldots exp_k(M_n) =_k c\, exp_{k-1}(M_1) \ldots exp_{k-1}(M_n)$, since $c\, exp_{k-1}(M_1) \ldots exp_{k-1}(M_n) =_k exp_k(M)$, and $exp_{k+1}(M) =_k c\, exp_k(M_1) \ldots exp_k(M_n)$ by downward closure of $=_{k+1}$, we have $exp_{k+1}(M) =_k exp_k(M)$.

- $M = \mathsf{fix}\, x.M'$, we have

1. $exp_{k+1}(M) =_{k+1} [exp_k(M)/x]_{k+1} exp_{k+1}(M')$ — by definition
2. $exp_{k+1}(M) =_k [exp_k(M)/x]_k exp_{k+1}(M')$ — by downward closure of $=_{k+1}$, functions on 1
3. $exp_k(M) =_k [exp_{k-1}(M)/x]_k exp_k(M')$ — by definition
4. $exp_k(M) =_{k-1} exp_{k-1}(M)$ — by IH on $k$
5. $exp_{k+1}(M') =_k exp_k(M')$ — by IH on $M'$
6. $max(x \in exp_k(M')) \leq k - 1$ — by guardedness of $M$ and 5.2.2
7. $exp_{k+1}(M) =_k exp_k(M)$ — by 2 3 and 5.2.5

□

**Theorem 5.3.2.** *Properties of exp:*

1. *If $exp(M) = M'$, then $exp_k(M) =_k M'$*
2. *$exp(x) = x$*
3. *$exp(c\, M_1 \ldots M_n) = c\, exp(M_1) \ldots exp(M_n)$*
4. *$exp_k(\mathsf{fix}\, x.M) =_k [exp_k(\mathsf{fix}\, x.M)/x] exp_k(M)$*
5. *$exp(\mathsf{fix}\, x.M) = [exp(\mathsf{fix}\, x.M)/x] exp(M)$*

*Proof.* 1. Directly by definition

2.

1. $exp(x)_0 =_0 x$ — by definition of $=_0$
2. $exp(x)_{k+1} =_{k+1} x$ — by definition of $exp_{k+1}$
3. $\forall k. exp(x)_k =_k x$ — from previous two steps
4. $exp(x) = x$ — by definition

3. For any $k$:

1. $\forall 1 \leq i \leq n. \forall k. exp(M_i) =_k exp_k(M_i)$ by definition of $exp$
2. $\forall 1 \leq i \leq n. exp(M_i) =_{k-1} exp_{k-1}(M_i)$ instantiate $\forall k.$ with $k-1$
3. $c\, exp_{k-1}(M_1) \ldots exp_{k-1}(M_n) =_k c\, exp(M_1) \ldots exp(M_n)$ by definition
4. $exp_k(c\, M_1 \ldots M_n) =_k c\, exp(M_1) \ldots exp(M_n)$ by definition and transitivity
5. $exp(c\, M_1 \ldots M_n) = c\, exp(M_1) \ldots exp(M_n)$ by definition

4. To show $exp_k(\mathsf{fix}\, x.M) =_k [exp_k(\mathsf{fix}\, x.M)/x]exp_k(M)$

   By induction on $k$, the case for $k = 0$ is trivial, for $k > 0$,
   1. $exp_k(\mathsf{fix}\, x.M) =_k [exp_{k-1}(\mathsf{fix}\, x.M)/x]_k exp_k(M)$ definition
   2. $exp_k(\mathsf{fix}\, x.M) =_{k-1} exp_{k-1}(\mathsf{fix}\, x.M)$ downward closure of $exp_k$
   3. $max(x \in exp_k(M)) \leq k-1$ guardedness of $M$
   4. $[exp_{k-1}(\mathsf{fix}\, x.M)/x]exp_k(\mathsf{fix}\, x.M) =_k [exp_k(\mathsf{fix}\, x.M)/x]_k exp_k(M)$ substitution lemma
   5. $exp_k(\mathsf{fix}\, x.M) =_k [exp_k(\mathsf{fix}\, x.M)/x]_k exp_k(M)$ transitivity, 1, 4

5. Directly follows from 4. since $k$ is generic in 4.

□

**Theorem 5.3.3** ($exp$ commutes with substitution)**.** *$exp_k([M_2/x]M_1) =_k [exp_k(M_2)/x]_k exp_k(M_1)$, for all $k$. Consequently, $exp([M_2/x]M_1) = [exp(M_2)/x]exp(M_1)$.*

*Proof.* To show $exp_k([M_2/x]M_1) =_k [exp_k(M_2)/x]_k exp_k(M_1)$, for all $k$,

by induction on $k$, then on the structure of $M_1$.

The case for $k = 0$ is trivial. Case $k > 0$, by cases on $M_1$

1. $M_1 = x$, then
   $exp_k([M_2/x]x) =_k exp_k(M_2) = [exp_k(M_2)/x]_k exp_k(x)$.
2. $M_1 = y \neq x$, then
   $exp_k([M_2/x]y) =_k exp_k(y) = [exp_k(M_2)/x]_k exp_k(y)$.
3. $M_1 = c\, M_1' \ldots M_n'$
   $exp_k([M_2/x](c\, M_1' \ldots M_n'))$
   $=_k exp_k(c\, [M_2/x]M_1' \ldots [M_2/x]M_n')$
   $=_k c\, exp_{k-1}([M_2/x]M_1') \ldots exp_{k-1}([M_2/x]M_n')$
   $=_k c\, [exp_{k-1}(M_2)/x]_{k-1} exp_{k-1}(M_1') \ldots [exp_{k-1}(M_2)/x]_{k-1} exp_{k-1}(M_n')$ by IH
   $=_k [exp_{k-1}(M_2)/x]_k (c\, exp_{k-1}(M_1') \ldots exp_{k-1}(M_n'))$
   $=_k [exp_{k-1}(M_2)/x]_k exp_k(c\, M_1' \ldots M_n')$
   $=_k [exp_k(M_2)/x]_k exp_k(c\, M_1' \ldots M_n')$ by the substitution lemma and guardedness, since $exp_{k-1}(M_2) =_{k-1} exp_k(M_2)$.

4. $M_1 = \mathsf{fix}\, y.M'$

   LHS: $exp_k([M_2/x]\,\mathsf{fix}\, y.M')$

   $=_k exp_k(\mathsf{fix}\, y.[M_2/x]M)$

   $=_k [exp_{k-1}(\mathsf{fix}\, y.[M_2/x]M)/y]_k exp_k([M_2/x]M)$

   $=_k [exp_{k-1}([M_2/x]\,\mathsf{fix}\, y.M)/y]_k exp_k([M_2/x]M)$

   $=_k [exp_{k-1}([M_2/x]\,\mathsf{fix}\, y.M)/y]_k [exp_k(M_2)/x]_k exp_k(M)$ by IH

   $=_k [[exp_{k-1}(M_2)/x]exp_{k-1}(\mathsf{fix}\, y.M)/y]_k [exp_k(M_2)/x]_k exp_k(M)$ by IH

   RHS: $[exp_k(M_2)/x]_k exp_k(\mathsf{fix}\, y.M)$

   $=_k [exp_k(M_2)/x]_k [exp_{k-1}(\mathsf{fix}\, y.M)/y]_k exp_k(M)$

   $=_k [[exp_k(M_2)/x]_k exp_{k-1}(\mathsf{fix}\, y.M)/y]_k [exp_k(M_2)/x]_k exp_k(M)$

   By Downward closure, $exp_{k-1}(M_2) =_{k-1} (M_2)$, since $max(x \in exp_{k-1}(\mathsf{fix}\, y.M)) \leq k-1$, by substitution lemma,$[exp_{k-1}(M_2)/x]exp_{k-1}(\mathsf{fix}\, y.M) =_{k-1} [exp_k(M_2)/x]_k exp_{k-1}(\mathsf{fix}\, y.M)$. Since $y$ is guarded in $[exp_k(M_2)/x]_k exp_k(M)$, by substitution lemma again we get the equality.

□

## 5.4 Type Checking Rules

The syntax of **SynTT** signature is as follows:

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Context | $\Gamma$ | ::= | $\cdot \mid \Gamma, x : P$ |
| Kind | $K$ | ::= | $\mathsf{type} \mid \mathsf{cotype} \mid P \to K$ |
| Canonical Type Families | $A$ | ::= | $P \mid P \to A$ |
| Atomic Type Families | $P$ | ::= | $a \mid P\, M$ |
| Terms | $M$ | ::= | $x \mid c\, M_1 \ldots M_n \mid \mathsf{fix}\, x.M$ |

As an optimization, we remove concrete priority assignment annotations on type and cotype, and instead rely on the ordering of declarations of family constants in $\Sigma$ to determine their ordering. Specifically, type level constants that are declared earlier in the signature have lower priorities (higher priority numbers) than those declared later in the signature.

The assignment procedure could be described as:

1. Initialize a variable $kP$ for current priority and set it to 0.
2. Iterate through the list of type level constants *in reverse order*, i.e., $a : K \in \Sigma$:
3. If $K$ ends with cotype and $kP$ is even, or if $K$ ends with type and $kP$ is odd, set the priority of $a$ to $kP$, increment $kP$ and proceed.

4. If $K$ ends with $\mathsf{cotype}$ and $kP$ is odd, or if $K$ ends with $\mathsf{type}$ and $kP$ is even, set the priority of $a$ to $kP + 1$, set $kP = kP + 1$ and proceed.

We may now use our old definition of $p$ that assigns the priorities to every constructor after performing the above procedure.

Because the typing rules enjoy the depth shift property, type checking is sound as long as the relative priority between types are fixed during introductions of new types and term constructors.

We simultaneously define the following judgments.

| | |
|---|---|
| $\Sigma\ \mathsf{ok}$ | Signature $\Sigma$ is type correct |
| $\vdash_\Sigma K \Leftarrow \mathsf{kind}$ | Kind $K$ is a valid kind |
| $\vdash_\Sigma A \Leftarrow \mathsf{type}$ | Type $A$ is a canonical type family |
| $\vdash_\Sigma P \Rightarrow K$ | Atomic type family $P$ synthesizes kind $K$ |
| $\Gamma \vdash_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ in context $\Gamma$ |
| $P_1 = P_2$ | $P_1$ and $P_2$ are equal atomic types. |
| $S \vdash M_1 = M_2$ | $M_1$ and $M_2$ are equal terms under constraint $S$ |
| $M\ \mathsf{valid}^\Delta$ | $M$ is priority correct with respect to $\Delta$ |
| $Cs\ \mathsf{good}$ | A list of constructors $Cs$ are priority correct |

$\boxed{\Sigma\ \mathsf{ok}}$ : $\Sigma$ is a valid signature.

$$\frac{}{\cdot\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Sigma, a : K\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\Sigma, c : A\ \mathsf{ok}}$$

$\boxed{\vdash_\Sigma K \Leftarrow \mathsf{kind}}$ : $K$ is a valid kind, assuming $\Sigma\ \mathsf{ok}$

$$\frac{}{\vdash_\Sigma \mathsf{type} \Leftarrow \mathsf{kind}} \qquad \frac{}{\vdash_\Sigma \mathsf{cotype} \Leftarrow \mathsf{kind}}$$

$$\frac{\vdash_\Sigma P \Rightarrow K' \qquad K' = \mathsf{type} \,/\, \mathsf{cotype} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\vdash_\Sigma P \to K \Leftarrow \mathsf{kind}}$$

$\boxed{\vdash_\Sigma A \Leftarrow \mathsf{type}}$ : $A$ is a good type, assuming $\Sigma\ \mathsf{ok}$

$$\frac{\vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type} \,/\, \mathsf{cotype} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\vdash_\Sigma P \to A \Leftarrow \mathsf{type}} \qquad \frac{\vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type} \,/\, \mathsf{cotype}}{\vdash_\Sigma P \Leftarrow \mathsf{type}}$$

$\boxed{\vdash_\Sigma P \Rightarrow K}$ : $A$ has kind $K$, assuming $\Sigma\ \mathsf{ok}$, $A$ constant or application

$$\frac{a : K \in \Sigma}{\vdash_\Sigma a \Rightarrow K} \qquad \frac{\vdash_\Sigma P \Rightarrow P' \to K \qquad M\ \mathsf{wellformed} \qquad \cdot \vdash_\Sigma M \Leftarrow P' \qquad M\ \mathsf{valid}^\cdot}{\vdash_\Sigma P\, M \Rightarrow K}$$

$\boxed{\Gamma \vdash_\Sigma M \Leftarrow P}$ : $M$ checks against type $P$, assuming $M$ wellformed, $M$ valid

$$\frac{P' = P}{\Gamma, x : P' \vdash_\Sigma x \Leftarrow P}\,\mathsf{refl} \qquad \frac{\Gamma, x : P \vdash_\Sigma M \Leftarrow P}{\Gamma \vdash_\Sigma \mathsf{fix}\, x.M \Leftarrow P}\,\mathsf{fix}$$

$$\frac{c : P_1 \to \cdots \to P_n \to P' \qquad P' = P \qquad \forall 1 \leq i \leq n. \Gamma \vdash_\Sigma M_i \Leftarrow P_i}{\Gamma \vdash_\Sigma c\, M_1 \ldots M_n \Leftarrow P}\,\mathsf{cM}$$

$\boxed{P_1 = P_2}$ : $P_1$ and $P_2$ are equal atomic types

$$\frac{}{a = a} \qquad \frac{P = P' \qquad \cdot \vdash M = M'}{P\, M = P'\, M'}$$

$\boxed{S \vdash M_1 = M_2}$ : $M_1$ and $M_2$ are equal canonical terms, modulo equalities in $S$ (unordered)

$$\frac{}{S \vdash_\Sigma x = x} \qquad \frac{}{S, M = M' \vdash_\Sigma M = M'} \qquad \frac{}{S, M' = M \vdash_\Sigma M = M'}$$

$$\frac{S, \mathsf{fix}\, x.M = M' \vdash_\Sigma [\mathsf{fix}\, x.M/x]M = M'}{S \vdash_\Sigma \mathsf{fix}\, x.M = M'} \qquad \frac{S, M = \mathsf{fix}\, x.M' \vdash_\Sigma M = [\mathsf{fix}\, x.M'/x]M'}{S \vdash_\Sigma M = \mathsf{fix}\, x.M'}$$

$$\frac{\forall 1 \leq i \leq n. S \vdash M_i = M_i'}{S \vdash c\, M_1 \ldots M_n = c\, M_1 \ldots M_n}$$

To define validity, we need to check each trace modeled by the recurrence of fixed-point bound variables. Thus, we let $Cs$ be a list of constructors and let $\Delta ::= \cdot \mid \Delta, x \hookrightarrow Cs$ denote a mapping from variables to a set of constructors. Implicit $\alpha$-renaming of fixpoint-bound variables are used to avoid potential name conflicts.

Define the operation of appending a constructor to a context by appending the constructor to every list that is in the context.

$\boxed{\Delta + c = \Delta'}$

$$\frac{}{\cdot + c = \cdot} \qquad \frac{\Delta + c = \Delta'}{(\Delta, x \hookrightarrow Cs) + c = \Delta', x \hookrightarrow Cs \cup \{c\}}$$

Then define

$\boxed{M\ \mathsf{valid}^{\Delta}}$ : $M$ is valid with respect to $\Delta$

$$\frac{Cs\ \mathsf{good}}{x\ \mathsf{valid}^{\Delta,\mathsf{x}\hookrightarrow\mathsf{Cs}}} \qquad \frac{M\ \mathsf{valid}^{\Delta,\mathsf{x}\hookrightarrow\varnothing}}{\mathsf{fix}\,x.M\ \mathsf{valid}^{\Delta}} \qquad \frac{M_i\ \mathsf{valid}^{\Delta+\mathsf{c}}}{c\,M_1\ \ldots\ M_n\ \mathsf{valid}^{\Delta}}$$

Recall that $p(c)$ denotes the priority of the constructor $c$. Define

$\boxed{Cs\ \mathsf{good}}$ iff $\forall c \in Cs$. either (a) $\exists c' \in Cs$ , $p(c')$ is even and $p(c') > p(c)$ or (b) $c$ is coinductive.

## 5.5 Metatheorems

In this section, we prove the decidability of type checking, soundness and completeness of **SynTT** with respect to $\mathbf{SemTT_R}$.

### 5.5.1 Decidability of Type Checking

**Theorem 5.5.1** (Decidability of Term Equality)**.** *Given any $M$ and $M'$ in* **SynTT***, it is decidable whether $\cdot \vdash_\Sigma M = M'$.*

*Proof.* By analyzing the type checking rules of $S \vdash_\Sigma M_1 = M_2$, it either (a) checking each argument subterms or (b) it adds one clause to the set of constraints. Due to the identity rules ($\dfrac{}{S, M = M' \vdash_\Sigma M = M'}$, $\dfrac{}{S, M' = M \vdash_\Sigma M = M'}$), no two equalities in $S$ are the same, because if so, we could immediately close the derivation by one of the identity rules.

It is known that for any rational term, the set of its subterms when viewed as rational trees are finite. Since $S$ contains only subterms of the initial term, $S$ will only contain finitely many equality constraints, and thus canonical term checking is decidable. □

**Corollary 5.5.2.** *Given any two atomic type families $P_1$ and $P_2$, it is decidable whether $\vdash_\Sigma P_1 = P_2$.*

*Proof.* Directly follows from Theorem 5.5.1. □

**Theorem 5.5.3** (Decidability of Type Checking)**.** *For any term $M$ type $A$, it is decidable whether $\Gamma \vdash_\Sigma M \Leftarrow A$.*

*Proof.* The rules are syntax directed and the premise terms are always smaller than the conclusion term. Family level equalities are decidable by Theorem 5.5.2

□

**Corollary 5.5.4** (Decidability of Signature Checking)**.** *For any signature* $\Sigma$*, it is decidable whether* $\Sigma$ $\mathsf{ok}$*.*

*Proof.* Directly follows from previous theorems.

□

### 5.5.2 Soundness

**Theorem 5.5.5** (Soundness of validity checking)**.** *If* $M$ $\mathsf{valid}$*, then* $exp(M)$ *is priority correct.*

*Proof.* Either $exp(M)$ is infinite or finite.

If $exp(M)$ is finite, then it is valid.

If $exp(M)$ is infinite, we may annotate the derivation of $M$ $\mathsf{valid}^{\Delta}$ with trace information.

The trace annotation is complete in the sense that if $T$ is an infinite trace in $exp(M)$, then it must be introduced by one of the fixed point expressions, which is the only way to introduce non-finiteness.

Since $exp(\mathsf{fix}\, x.M) = [exp(\mathsf{fix}\, x.M)/x]exp(M)$, $exp(M)$ could be thought of as a context for iteratively constructing the infinite object, and the constructors that occur infinitely often on the path will just involve the constructors in between $exp(M)$ and $x$.

And thus, the conditions on $Cs$ mirror and ensure the desired validity on traces of $exp(M)$.

□

To present the soundness of the type checking of **SynTT**, we need to extend the derivations of **SemTT** with variables, that is, hypothetical derivations to be replaced by actual derivations. We use the notation $\mathcal{V}_{\Delta}(R)$, where $R$ is the "name" of the variable and $\Delta$ is the custom data associated with the variable that may be accessed later when actual substitution occurs. When a variable carries no custom data, we write $\mathcal{V}(R)$.

We use the translation $\mathsf{EXP}$ to map **SynTT** derivations to **SemTT** derivations with variables. $\mathsf{EXP}$ is also defined by step indexing as **SemTT** derivations are usually infinite.

**Theorem 5.5.6** (Soundness of Object Equality)**.** *If* $\cdot \vdash M = M'$*, then* $exp(M) = exp(M')$*.*

*Proof.* $\mathsf{EXP}$ maps derivations of the form $S \vdash M = M'$ to derivations of $exp(M) = exp(M')$. Here, we choose to build an infinite proof of $exp(M) = exp(M')$ instead of a circular proof.

- $\mathsf{EXP}_{j+1}(\dfrac{}{S \vdash x = x}) = \dfrac{}{x = x}$

- $\mathsf{EXP}_{j+1}(\dfrac{\begin{array}{c}\mathcal{D}_i\\ \forall 1 \leq i \leq n. S \vdash M_i = M_i'\end{array}}{S \vdash c\, M_1 \cdots M_n = c\, M_1' \cdots M_n'})$

$$= \frac{\begin{array}{c}\mathcal{D}_i\\ \mathsf{EXP}_j(S \vdash M_i = M_i')\\ \forall 1 \leq i \leq n. exp_{k-1}(M_i) = exp_{k-1}(M_i')\end{array}}{c\, exp_{k-1}(M_1) \cdots exp_{k-1}(M_n) = c\, exp_{k-1}(M_1') \cdots exp_{k-1}(M_n')}$$

- $\mathsf{EXP}_{j+1}(\dfrac{}{S, M = M' \vdash M = M'}) = \mathcal{V}(M = M' \vdash M = M')$
- $\mathsf{EXP}_{j+1}(\dfrac{}{S, M = M' \vdash M' = M}) = \mathcal{V}(M = M' \vdash M' = M)$
- $\mathsf{EXP}_{j+1}(\dfrac{\begin{array}{c}\mathcal{D}\\ S, \mathsf{fix}\, x.M = M' \vdash [\mathsf{fix}\, x.M/x]M = M'\end{array}}{S \vdash \mathsf{fix}\, x.M = M'})$

$$[\mathsf{SYM}(\mathsf{EXP}_j(\frac{\begin{array}{c}\mathcal{D}\\ S, \mathsf{fix}\, x.M = M' \vdash [\mathsf{fix}\, x.M/x]M = M'\end{array}}{S \vdash \mathsf{fix}\, x.M = M'}))/\mathcal{V}(\frac{}{\mathsf{fix}\, x.M = M' \vdash M' = \mathsf{fix}\, x.M}$$

$$][\mathsf{EXP}_j(\frac{\begin{array}{c}\mathcal{D}\\ S, \mathsf{fix}\, x.M = M' \vdash [\mathsf{fix}\, x.M/x]M = M'\end{array}}{S \vdash \mathsf{fix}\, x.M = M'})/\mathcal{V}(\frac{}{\mathsf{fix}\, x.M = M' \vdash \mathsf{fix}\, x.M = M'})$$

$$] \mathsf{EXP}_{j+1}(\mathcal{D})$$

$$= \quad exp([\mathsf{fix}\, x.M/x]M) = exp(M')$$

We show that $exp(\mathsf{fix}\, x.M) = [exp(\mathsf{fix}\, x.M)/x]exp(M) = exp([\mathsf{fix}\, x.M/x]M)$ by Theorem 5.3.3 and Theorem 5.3.3.

$\mathsf{SYM}$ denotes the symmetric transformation of the equality proof of **SemTT** terms (Theorem 2.2.2).

- Similarly, we define

$$\mathsf{EXP}_{j+1}(\frac{\begin{array}{c}\mathcal{D}\\ S, M = \mathsf{fix}\, x.M' \vdash M = [\mathsf{fix}\, x.M'/x]M'\end{array}}{S \vdash M = \mathsf{fix}\, x.M'})$$

$$[\mathsf{SYM}(\mathsf{EXP}_j(\frac{\begin{array}{c}\mathcal{D}\\ S, M = \mathsf{fix}\, x.M' \vdash M = [\mathsf{fix}\, x.M'/x]M'\end{array}}{S \vdash M = \mathsf{fix}\, x.M'}))/\mathcal{V}(\frac{}{M = \mathsf{fix}\, x.M' \vdash \mathsf{fix}\, x.M' = M}$$

$$][\mathsf{EXP}_j(\frac{\begin{array}{c}\mathcal{D}\\ S, M = \mathsf{fix}\, x.M' \vdash M = [\mathsf{fix}\, x.M'/x]M'\end{array}}{S \vdash M = \mathsf{fix}\, x.M'})/\mathcal{V}(\frac{}{M = \mathsf{fix}\, x.M' \vdash M = \mathsf{fix}\, x.M'})$$

$$] \mathsf{EXP}_{j+1}(\mathcal{D})$$

$$= \quad exp(M) = exp([\mathsf{fix}\, x.M'/x]M')$$

Again, we have $exp(\mathsf{fix}\, x.M') = [exp(\mathsf{fix}\, x.M')/x]exp(M') = exp([\mathsf{fix}\, x.M'/x]M')$ by The-

orem 5.3.3 and Theorem 5.3.3.

Since $M$ is guarded, the defined translation $\mathsf{EXP}$ is productive because of the guardedness of $M$ in $\mathsf{fix}\, x.M$. Thus, $\mathsf{EXP}$ denotes a unique **SemTT** object equality proof given a **SynTT** equality proof.

□

The function $exp$ can be naturally extended to expand type families, kinds and signature.

$exp(P \to A) = exp(P) \to exp(A)$

$exp(P\, M) = exp(P)\, exp(M)$

$exp(a) = a$

Similarly, $\mathsf{EXP}$ naturally extends to type family equalities.

**Theorem 5.5.7** (Soundness of Typing)**.** *If* $\cdot \vdash_\Sigma M \Leftarrow P$ *in* **SynTT**, *then* $\vdash_{exp(\Sigma)} exp(M) \Leftarrow exp(P)$ *in* $\mathbf{SemTT_R}$

*Proof.* To show $\vdash_{exp(\Sigma)} exp(M) \Leftarrow exp(P)$ in $\mathbf{SemTT_R}$, it is sufficient to show $\vdash^{[k_0,k_1,\ldots,k_n]}_{exp(\Sigma)} exp(M) \Leftarrow exp(A)$ in $\mathbf{SemTT_R}$, where $k_n$'s are fresh symbols when $n$ is even and $k_n$'s are concrete numbers when $n$ is odd. Following a similar method as the proof or Theorem 3.4.6, we let $k_i$'s (where $i$ is odd) be fresh unknown variables $j_i$ along with their associated minimum values $m_i$.

Then we translate the proof by induction on the derivation of $\Gamma \vdash_\Sigma M \Leftarrow P$. When encountering the rule $\mathsf{cM}$, if $p(P)$ is even, we decrement the value mapped by $p(P)$ in the depth indexing context, and proceed to translate the premises. If $p(P)$ is odd, we decrement the unknown variable mapped by $p(P)$ int depth context, add one to the minimum value associated with that variable, and proceed to analyze the premises.

When encountering the rule $\mathsf{fix}$, we just directly proceed to analyze the premise. When encountering the rule $\mathsf{refl}$, we may close the derivation with *bud*, and make it point to the derivation $\Gamma \vdash_\Sigma \mathsf{fix}\, x.M \Leftarrow P$ where the fixed-point bound variable $x$ is first introduced in the context. We set the term $M$ of the bud rule to be $exp(\mathsf{fix}\, x.M)$. The bud rule can always satisfy the closure condition. The validity of the terms ensures that the highest priority predicate along the path is coinductive, and guardedness ensures that some constructor must have been encountered. Thus, the depth context must be smaller on an even principal index.

Since the above procedure is defined by the structure induction on the derivation, it will terminate. We then instantiate all unknown variables $j_i$ (where $i$ is odd) and evaluate

all corresponding arithmetic expressions associated with $j_i$. We then would have a proof of $\vdash_{exp(\Sigma)} exp(M) \Leftarrow exp(P)$ in $\mathbf{SemTT_R}$.

Formally, the translation procedure maybe realized using the $\mathsf{EXP}$ function defined by induction on the argument derivation as follows.

We extend $\mathsf{EXP}$ to act on derivations of $\Gamma \vdash M \Leftarrow A$, and now $\mathsf{EXP}$ carries the current depth indexing context in the top left corner, written as ${}^{D}\mathsf{EXP}$.

- ${}^{D}\mathsf{EXP}(\dfrac{P' = P}{\Gamma, x : P' \vdash_\Sigma x \Leftarrow P}) = \mathcal{V}_D(\dfrac{}{x \vdash x})$

- $${}^{D[p(A)\mapsto k+1]}\mathsf{EXP}(\frac{c : P_1 \to \cdots \to P_n \in P' \in \Sigma \qquad \overset{\mathcal{E}}{P = P'} \qquad \forall 1 \le i \le n. \Gamma \vdash_\Sigma \overset{\mathcal{D}_i}{M_i} \Leftarrow P_i}{\Gamma \vdash_\Sigma c\, M_1 \cdots M_n \Leftarrow P})$$

$$= \frac{\begin{array}{c} c : exp(P_1) \to \cdots \to exp(P_n) \to exp(P') \in exp(\Sigma) \\ \mathsf{EXP}(\overset{\mathcal{E}}{P = P'}) \qquad\qquad {}^{D[p(A)\mapsto k]}\mathsf{EXP}(\Gamma \vdash_\Sigma \overset{\mathcal{D}_i}{M_i} \Leftarrow P_i) \\ exp(P) = exp(P') \qquad \forall 1 \le i \le n. \vdash^{D[p(P)\mapsto k]}_{exp(\Sigma)} exp(M_i) \Leftarrow exp(P_i) \end{array}}{\vdash^{D[p(A)\mapsto k+1]}_{exp(\Sigma)} c\, exp(M_1)\ \ldots\ exp(M_n) \Leftarrow exp(P)}$$

- ${}^{D}\mathsf{EXP}(\dfrac{\overset{\mathcal{D}}{\Gamma, x : P \vdash_\Sigma M \Leftarrow P}}{\Gamma \vdash_\Sigma \mathsf{fix}\, x.M \Leftarrow P})$

$$[\frac{}{\vdash^{D'}_{exp(\Sigma)} exp(M) \Leftarrow exp(P)} bud/\mathcal{V}_{D'}(\frac{}{x \vdash x})]\, \mathsf{EXP}(\Gamma, x : P \vdash_\Sigma \overset{\mathcal{D}}{M} \Leftarrow P)$$

$$= \qquad \vdash^{D}_{exp(\Sigma)} [exp(\mathsf{fix}\, x.M)/x]exp(M) \Leftarrow exp(P)$$

We have $[exp(\mathsf{fix}\, x.M)/x]exp(M) = exp(\mathsf{fix}\, x.M)$ by Theorem 5.3.3.5. Note that the recursive call's data argument $D'$ is retrieved from the variable declaration.

□

### 5.5.3 Completeness

We note that every rational term can be written as a fixed point expression. For instance, Amadio and Cardelli outlined a detailed procedure of translating a rational term to a fixed point expression [AC93]. Let $exp^{-1}$ denote this translation.

**Theorem 5.5.8** (Completeness of Typing)**.** *If* $\cdot \vdash_\Sigma M \Leftarrow P$ *in* $\mathbf{SemTT_R}$*,*

*then* $\vdash_{exp^{-1}(\Sigma)} exp^{-1}(M) \Leftarrow exp^{-1}(P)$ *in* $\mathbf{SynTT}$*.*

*Proof.* The proof of $\cdot \vdash_\Sigma M \Leftarrow P$ consists of a single derivation with the conclusion $\cdot \vdash^{[k_0,k_1,k_2,\ldots,k_n]}_{\Sigma} M \Leftarrow P$.

We construct the derivation of $\vdash_{exp^{-1}(\Sigma)} exp^{-1}(M) \Leftarrow exp^{-1}(P)$ in **SynTT** iteratively as follows:

1. Erase all the depth contexts and replace all $cM$ of $\mathbf{SemTT_R}$ rule by $\mathsf{cM}$ rule of **SynTT**.

2. For all leaf rules of *bud*, we replace the rule with one of the identity rules and insert (if we have not done so already) the $\mathsf{fix}$ rule at the position where the *bud* rule points to. It is always possible to write the expression as a fixed point because the *bud* rule requires two terms to be identical.

□

# Chapter 6

# Type Dependencies on Simple Terms

Dependent types provide mechanisms of formalizing judgments about infinite terms. They can encode "rule schemes" that rely on the notion of metavariables. In this chapter, we explore $\mathbf{SemTT^{DT}}$, $\mathbf{SemTT_R^{DT}}$, $\mathbf{SynTT^{DT}}$, the extension of $\mathbf{SemTT}$, $\mathbf{SemTT_R}$, $\mathbf{SynTT}$ with dependent $\Pi$-types.

## 6.1 Type Checking Rules

### 6.1.1 Semantic Type Theory with Dependent Types

The system $\mathbf{SemTT^{DT}}$ is the extension of $\mathbf{SemTT}$ with dependent types.

Syntax:

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Context | $\Gamma$ | ::= | $\cdot \mid \Gamma, x : P$ |
| Kind | $K$ | ::= | $\mathsf{type}^{\mathbb{N}} \mid \mathsf{cotype}^{\mathbb{N}} \mid \Pi x : P.\ K$ |
| Canonical Type Families | $A$ | ::= | $P \mid \Pi x : P.\ A$ |
| Atomic Type Families | $P$ | ::= | $a \mid P\, M$ |
| Terms | $M_{k+1}$ | ::= | $x \mid c\, M_{1(k)} \ \ldots\ M_{n(k)}$ |

The priority assignment function $p$ just ignores type dependencies and remains as before. We still maintain the requirement that the priorities for each type family level constant are different. We replicate the definitions here.

$\boxed{p(K)}$

$p(\mathsf{type}^k) = k$

$p(\mathsf{cotype}^k) = k$

$p(\Pi x : P.K) = p(K)$

$\boxed{p(A)}$ and $\boxed{p(P)}$

$p(\Pi x : P.A) = p(A)$

$p(P\,M) = p(P)$

$p(a) = p(K)$ where $a : K \in \Sigma$

$\boxed{p(c)}$

$p(c) = p(A)$ where $c : A \in \Sigma$

We simultaneously define the following judgments

| | |
|---|---|
| $\Sigma\ \mathsf{ok}$ | Signature $\Sigma$ is type correct |
| $\vdash_\Sigma \Gamma\ \mathsf{ok}$ | Context $\Gamma$ is well-formed |
| $\Gamma \vdash_\Sigma K \Leftarrow \mathsf{kind}$ | Kind $K$ is a valid kind |
| $\Gamma \vdash_\Sigma A \Leftarrow \mathsf{type}$ | Type $A$ is a canonical type family |
| $\Gamma \vdash_\Sigma P \Rightarrow K$ | Atomic type family $P$ synthesizes kind $K$ |
| $\Gamma \vdash_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ |
| $\Gamma \vdash^D_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ in the depth context $D$ |

$\boxed{\Sigma\ \mathsf{ok}}$ : $\Sigma$ is a valid signature.

$$\frac{}{\vdash \cdot\ \mathsf{ok}} \qquad \frac{\vdash \Sigma\ \mathsf{ok} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\vdash \Sigma, a : K\ \mathsf{ok}} \qquad \frac{\vdash \Sigma\ \mathsf{ok} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\vdash \Sigma, c : A\ \mathsf{ok}}$$

$\boxed{\vdash_\Sigma \Gamma\ \mathsf{ok}}$ : $\Gamma$ is a valid context, assuming that $\Sigma\ \mathsf{ok}$.

$$\frac{}{\vdash_\Sigma \cdot\ \mathsf{ok}} \qquad \frac{\vdash_\Sigma \Gamma\ \mathsf{ok} \qquad \Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n}{\vdash_\Sigma \Gamma, x : P\ \mathsf{ok}}$$

$\boxed{\Gamma \vdash_\Sigma K \Leftarrow \mathsf{kind}}$ : $K$ is a valid kind, assuming that $\Sigma\ \mathsf{ok}$, $\vdash_\Sigma \Gamma\ \mathsf{ok}$.

$$\frac{n \notin pOccurs(\Sigma)}{\Gamma \vdash_\Sigma \mathsf{type}^n \Leftarrow \mathsf{kind}} (n \text{ is an odd natural number})$$

$$\frac{n \notin pOccurs(\Sigma)}{\Gamma \vdash_\Sigma \mathsf{cotype}^n \Leftarrow \mathsf{kind}} (n \text{ is an even natural number})$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K' \qquad K' = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \Gamma, x : P \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Gamma \vdash_\Sigma \Pi x : P.\ K \Leftarrow \mathsf{kind}}$$

$\boxed{\Gamma \vdash_\Sigma A \Leftarrow \mathsf{type}}$ : $A$ is a good type, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \Gamma, x : P \vdash_\Sigma B \Leftarrow \mathsf{type}}{\Gamma \vdash_\Sigma \Pi x : P.\ B \Leftarrow \mathsf{type}}$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n\, e}{\Gamma \vdash_\Sigma P \Leftarrow \mathsf{type}}$$

$\boxed{\Gamma \vdash_\Sigma P \Rightarrow K}$ : $P$ has kind $K$, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{a : K \in \Sigma}{\vdash_\Sigma a \Rightarrow K} \qquad\qquad \frac{\Gamma \vdash_\Sigma P \Rightarrow \Pi x : P'.\ K \qquad \Gamma \vdash_\Sigma M \Leftarrow P'}{\Gamma \vdash_\Sigma P\, M \Rightarrow [M/x]K}$$

$\boxed{\Gamma \vdash_\Sigma M \Leftarrow P}$ is defined to be:
When $maxP(\Sigma)$ is even :

$$\forall k_0.\forall k_2.\forall k_4 \ldots \forall k_{maxP(\Sigma)}.\exists j_1.\exists j_3.\exists j_5.\ldots.\exists j_{maxP(\Sigma)-1}.\Gamma \vdash_\Sigma^{[k_0,j_1,k_2,j_3,\ldots,k_{maxP(\Sigma)}]} M \Leftarrow P$$

When $maxP(\Sigma)$ is odd

$$\forall k_0.\forall k_2.\forall k_4 \ldots \forall k_{maxP(\Sigma)-1}.\exists j_1.\exists j_3.\exists j_5.\ldots.\exists j_{maxP(\Sigma)}.\Gamma \vdash_\Sigma^{[k_0,j_1,k_2,j_3,\ldots,j_{maxP(\Sigma)}]} M \Leftarrow P$$

$\boxed{\Gamma \vdash_\Sigma^D M \Leftarrow P}$ : $M$ checks against type $P$ in the depth indexing context $D$, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{(\text{Require: } p(P) \text{ is even and } D(p(P)) = 0)}{\Gamma \vdash_\Sigma^D M \Leftarrow P} cM0$$

(cM1 Requires: (1) $D' < D$ on principal index $p(P)$, (2) $D'(p(P)) + 1 = D(p(P))$, and (3)
$\begin{cases} D'(k) = D(k) \text{ for all } k > p(P) \text{ and } k \text{ is an even number} & \text{if } p(P) \text{ is even} \\ D'(k) = D(k) \text{ for all } k > p(P) & \text{if } p(P) \text{ is odd} \end{cases}$ )

$$\frac{\begin{array}{c} c : \Pi x_1 : P_1.\ \Pi x_2 : P_2.\ \ldots\ \Pi x_n : P_n.\ P' \in \Sigma \qquad [M_1, \ldots, M_n/x_1, \ldots, x_n]P' = P \\ \forall 1 \le i \le n. \Gamma \vdash_\Sigma^{D'} M_i \Leftarrow [M_1, \ldots, M_{i-1}/x_1, \ldots, x_{i-1}]P_i \end{array}}{\vdash_\Sigma^D c\, M_1\ \ldots\ M_n \Leftarrow P} cM1$$

### 6.1.2 Semantic Type Theory with Rational Terms and Dependent Types

Syntax:

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Context | $\Gamma$ | ::= | $\cdot \mid \Gamma, x : P$ |
| Kind | $K$ | ::= | $\mathsf{type}^{\mathbb{N}} \mid \mathsf{cotype}^{\mathbb{N}} \mid \Pi x : P.\ K$ |
| Canonical Type Families | $A$ | ::= | $P \mid \Pi x : P.\ A$ |
| Atomic Type Families | $P$ | ::= | $a \mid P\, M$ |
| Terms | $M_{k+1}$ | ::= | $x \mid c\, M_{1(k)} \ \ldots\ M_{n(k)}$ |

We simultaneously define the following judgments

| | |
|---|---|
| $\Sigma\ \mathsf{ok}$ | Signature $\Sigma$ is type correct |
| $\vdash_\Sigma \Gamma\ \mathsf{ok}$ | Context $\Gamma$ is well-formed |
| $\Gamma \vdash_\Sigma K \Leftarrow \mathsf{kind}$ | Kind $K$ is a valid kind |
| $\Gamma \vdash_\Sigma A \Leftarrow \mathsf{type}$ | Type $A$ is a canonical type family |
| $\Gamma \vdash_\Sigma P \Rightarrow K$ | Atomic type family $P$ synthesizes kind $K$ |
| $\Gamma \vdash_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ |
| $\Gamma \vdash^D_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ in the depth context $D$ |

The priority assignment function $p$ just ignores type dependencies and remains unchanged.

$\boxed{\Sigma\ \mathsf{ok}}$ : $\Sigma$ is a valid signature.

$$\frac{}{\vdash \cdot\ \mathsf{ok}} \qquad \frac{\vdash \Sigma\ \mathsf{ok} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\vdash \Sigma, a : K\ \mathsf{ok}} \qquad \frac{\vdash \Sigma\ \mathsf{ok} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\vdash \Sigma, c : A\ \mathsf{ok}}$$

$\boxed{\vdash_\Sigma \Gamma\ \mathsf{ok}}$ : $\Gamma$ is a valid context, assuming that $\Sigma\ \mathsf{ok}$.

$$\frac{}{\vdash_\Sigma \cdot\ \mathsf{ok}} \qquad \frac{\vdash_\Sigma \Gamma\ \mathsf{ok} \qquad \Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n}{\vdash_\Sigma \Gamma, x : P\ \mathsf{ok}}$$

$\boxed{\Gamma \vdash_\Sigma K \Leftarrow \mathsf{kind}}$ : $K$ is a valid kind, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{n \notin pOccurs(\Sigma)}{\Gamma \vdash_\Sigma \mathsf{type}^n \Leftarrow \mathsf{kind}} (n \text{ is an odd natural number})$$

$$\frac{n \notin pOccurs(\Sigma)}{\Gamma \vdash_\Sigma \mathsf{cotype}^n \Leftarrow \mathsf{kind}} (n \text{ is an even natural number})$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K' \qquad K' = \mathsf{type} \,/\, \mathsf{cotype} \qquad \Gamma, x : P \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Gamma \vdash_\Sigma \Pi x : P.\ K \Leftarrow \mathsf{kind}}$$

$\boxed{\Gamma \vdash_\Sigma A \Leftarrow \mathsf{type}}$ : $A$ is a good type, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \Gamma, x : P \vdash_\Sigma B \Leftarrow \mathsf{type}}{\Gamma \vdash_\Sigma \Pi x : P.\ B \Leftarrow \mathsf{type}}$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type} \,/\, \mathsf{cotype}}{\Gamma \vdash_\Sigma P \Leftarrow \mathsf{type}}$$

$\boxed{\Gamma \vdash_\Sigma P \Rightarrow K}$ : $P$ has kind $K$, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{a : K \in \Sigma}{\vdash_\Sigma a \Rightarrow K} \qquad\qquad \frac{\Gamma \vdash_\Sigma P \Rightarrow \Pi x : P'.\ K \qquad \Gamma \vdash_\Sigma M \Leftarrow P'}{\Gamma \vdash_\Sigma P\, M \Rightarrow [M/x]K}$$

$\boxed{\Gamma \vdash_\Sigma M \Leftarrow P}$ is defined to be:

$$\Gamma \vdash_\Sigma^{[k_0,k_1,k_2,\ldots,k_n]} M \Leftarrow P$$

where

(1) $k_i$ is a fresh symbol $k_i$ if $i$ is even

(2) $k_i$ is a concrete number if $i$ is odd

(3) $n = maxP(\Sigma)$

$\boxed{\Gamma \vdash_\Sigma^D M \Leftarrow P}$ : $M$ checks against type $P$ in the depth indexing context $D$, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{c : \Pi x_1 : P_1.\ \Pi x_2 : P_2.\ \ldots\ \Pi x_n : P_n.\ P' \in \Sigma \qquad [M_1, \ldots, M_n/x_1, \ldots, x_n]P' = P \qquad \forall 1 \leq i \leq n.\ \vdash_\Sigma^{D[p(P)\mapsto k]} M_i \Leftarrow [M_1, \ldots, M_{i-1}/x_1, \ldots, x_{i-1}]P_i}{\vdash_\Sigma^{D[p(P)\mapsto k+1]} c\, M_1\ \ldots\ M_n \Leftarrow P}\, cM$$

$$\frac{}{\Gamma \vdash_\Sigma^D M \Leftarrow P}\, bud$$

The closure condition of *bud* remains unchanged.

### 6.1.3 Syntactic Type Theory with Dependent Types

Syntax:

| | | | |
|---|---|---|---|
| Signature | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Context | $\Gamma$ | ::= | $\cdot \mid \Gamma, x : P$ |
| Kind | $K$ | ::= | $\mathsf{type} \mid \mathsf{cotype} \mid \Pi x : P.\ K$ |
| Canonical Type Families | $A$ | ::= | $P \mid \Pi x : P.\ A$ |
| Atomic Type Families | $P$ | ::= | $a \mid P\, M$ |
| Terms | $M$ | ::= | $x \mid c\, M_1\ \ldots\ M_n \mid \mathsf{fix}\, x.M$ |

We still use the automatic priority assignment from before. We require terms to be well-formed. Since the syntax of the terms is not changed, the well-formedness definition is the same as before. The equality checking rules for atomic families and terms remain the same as well.

We adapt the type checking rules to include type dependencies. We have the following judgments.

| | |
|---|---|
| $\Sigma\ \mathsf{ok}$ | Signature $\Sigma$ is type correct |
| $\vdash_\Sigma \Gamma\ \mathsf{ok}$ | Context $\Gamma$ is well-formed |
| $\Gamma \vdash_\Sigma K \Leftarrow \mathsf{kind}$ | Kind $K$ is a valid kind |
| $\Gamma \vdash_\Sigma A \Leftarrow \mathsf{type}$ | Type $A$ is a canonical type family |
| $\Gamma \vdash_\Sigma P \Rightarrow K$ | Atomic type family $P$ synthesizes kind $K$ |
| $\Gamma \vdash_\Sigma M \Leftarrow P$ | Term $M$ checks against type $P$ in context $\Gamma$ |
| $M\ \mathsf{valid}^{\Gamma;\Delta}$ | $M$ is priority correct with respect to $\Delta$ and $\Gamma$ |

$\boxed{\Sigma\ \mathsf{ok}}$ : $\Sigma$ is a valid signature.

$$\frac{}{\cdot\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Sigma, a : K\ \mathsf{ok}} \qquad \frac{\Sigma\ \mathsf{ok} \qquad \vdash_\Sigma A \Leftarrow \mathsf{type}}{\Sigma, c : A\ \mathsf{ok}}$$

$\boxed{\vdash_\Sigma \Gamma\ \mathsf{ok}}$ : $\Gamma$ is a valid context, assuming that $\Sigma$ ok.

$$\frac{}{\vdash_\Sigma \cdot\ \mathsf{ok}} \qquad \frac{\vdash_\Sigma \Gamma\ \mathsf{ok} \qquad \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n}{\vdash \Gamma, x : P\ \mathsf{ok}}$$

$\boxed{\Gamma \vdash_\Sigma K \Leftarrow \mathsf{kind}}$ : $K$ is a valid kind, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{}{\Gamma \vdash_\Sigma \mathsf{type} \Leftarrow \mathsf{kind}} \qquad \frac{}{\Gamma \vdash_\Sigma \mathsf{cotype} \Leftarrow \mathsf{kind}}$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K' \qquad K' = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \Gamma, x : P \vdash_\Sigma K \Leftarrow \mathsf{kind}}{\Gamma \vdash_\Sigma \Pi x : P.\ K \Leftarrow \mathsf{kind}}$$

$\boxed{\Gamma \vdash_\Sigma A \Leftarrow \mathsf{type}}$ : $A$ is a good type, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type}^n \,/\, \mathsf{cotype}^n \qquad \Gamma, x : P \vdash_\Sigma B \Leftarrow \mathsf{type}}{\Gamma \vdash_\Sigma \Pi x : P.\ B \Leftarrow \mathsf{type}}$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow K \qquad K = \mathsf{type} \,/\, \mathsf{cotype}}{\Gamma \vdash_\Sigma P \Leftarrow \mathsf{type}}$$

$\boxed{\Gamma \vdash_\Sigma P \Rightarrow K}$ : $P$ has kind $K$, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok.

$$\frac{a : K \in \Sigma}{\Gamma \vdash_\Sigma a \Rightarrow K} \qquad \frac{\Gamma \vdash_\Sigma P \Rightarrow \Pi x : P.K \qquad \cdot \vdash_\Sigma M \Leftarrow P \qquad M\ \mathsf{valid}^{\Gamma;\cdot} \qquad M\ \mathsf{wellformed}}{\Gamma \vdash_\Sigma P\, M \Rightarrow [M/x]K}$$

$\boxed{\Gamma \vdash_\Sigma M \Leftarrow P}$ : $M$ checks against type $P$, assuming that $\Sigma$ ok, $\vdash_\Sigma \Gamma$ ok, $M$ wellformed, $M$ $\mathsf{valid}^{\Gamma;\cdot}$,

$$\frac{P' = P}{\Gamma, x : P' \vdash_\Sigma x \Leftarrow P}\ \mathsf{refl} \qquad \frac{\Gamma, x : P \vdash_\Sigma M \Leftarrow P}{\Gamma \vdash_\Sigma \mathsf{fix}\, x.M \Leftarrow P}\ \mathsf{fix}$$

$$\frac{\begin{array}{c} c : \Pi x_1 : P_1.\ \Pi x_2 : P_2.\ \ldots\ \Pi x_n : P_n.\ P' \in \Sigma \\ [M_1, \ldots, M_n/x_1, \ldots, x_n]P' = P \qquad \forall 1 \le i \le n . \Gamma \vdash_\Sigma M_i \Leftarrow [M_1, \ldots, M_{i-1}/x_1, \ldots, x_{i-1}]P_i \end{array}}{\Gamma \vdash_\Sigma c\, M_1 \ldots\ M_n \Leftarrow P}\ \mathsf{cM}$$

The validity checking is updated to accommodate both fix-bound and $\Pi$-bound variables. The condition for $Cs$ good remains the same.

$\boxed{M\ \mathsf{valid}^{\Gamma;\Delta}}$ : $M$ is valid with respect to $\Delta$ (fix-bound variables) and $\Gamma$ ($\Pi$-bound variables). $\Gamma$ and $\Delta$ should be disjoint.

$$\frac{x \in \Gamma \qquad x \notin \Delta}{x\ \mathsf{valid}^{\Gamma;\Delta}} \qquad \frac{x \notin \Gamma \qquad Cs\ \mathsf{good}}{x\ \mathsf{valid}^{\Gamma;\Delta,x\hookrightarrow Cs}} \qquad \frac{M\ \mathsf{valid}^{\Gamma;\Delta,x\hookrightarrow\varnothing}}{\mathsf{fix}\,x.M\ \mathsf{valid}^{\Gamma;\Delta}} \qquad \frac{M_i\ \mathsf{valid}^{\Gamma;\Delta+c}}{c\,M_1\ \ldots\ M_n\ \mathsf{valid}^{\Gamma;\Delta}}$$

### 6.1.4 Metatheorems

Because the structure of the terms has not changed, we expect all parallel counterparts of the metatheorems for the simply typed case to hold in the dependently typed case.

## 6.2 An Example of Bitstreams

Informally, a bit stream is an infinite word over $0, 1$. For instance, the following is a bitstream:

$$10100101010010111101110100010 11 \ldots\ldots\ldots\ldots$$

Informally, we may define the bitstream by induction on the observation depth:

- Anything is a bit stream at observation depth zero.
- $1S$ and $0S$ are bitstreams of depth $k+1$ if $S$ is a bitstream of depth $k$.

We have the following $\mathbf{SemTT^{DT}}$ signature:

$$\Sigma = \begin{array}{lll} \mathit{bitstream} & : & \mathit{cotype}^2. \\ b0 & : & \mathit{bitstream} \to \mathit{bitstream}. \\ b1 & : & \mathit{bitstream} \to \mathit{bitstream}. \end{array}$$

There is a bijection between canonical forms of the type *bitstream* and bitstreams defined informally evidenced by the following encoding defined by induction on the observation depth:

- $0S \gg_{k+1} b0\,M$ if $S \gg_k M$
- $1S \gg_{k+1} b1\,M$ if $S \gg_k M$
- $S \gg_0 M$ always

**Theorem 6.2.1** (Adequacy of the Definition)**.** *If $S \gg M$, then $S$ is a bitstream and $\cdot \vdash M \Leftarrow \mathit{bitstream}$.*

*Proof.* Direct by induction on the index $k$. □

**Theorem 6.2.2** (Adequacy of Encoding)**.** *1. If $S$ is a bitstream, then there exists a unique $M$ such that $S \gg M$, and $\cdot \vdash_\Sigma M \Leftarrow \mathit{bitstream}$.*

*2. If $\cdot \vdash_\Sigma M \Leftarrow bitstream$, then there exists a unique $S$ such that $S \gg M$, and $S$ is a bitstream.*

*Proof.* In the proof, we shall omit $\Sigma$ under the $\vdash$ where appropriate to reduce visual clutter.

We prove the following theorems for all $k$, then the results follow. 1. If $S$ is a bitstream of observation depth $k$, then there exists a unique $M$ (up to depth $k$) such that $S \gg_k M$, and $\cdot \vdash^{[-,-,k]} M \Leftarrow bitstream$.

2. If $\cdot \vdash^{[-,-,k]} M \Leftarrow bitstream$, then there exists a unique bitstream $S$ (up to depth $k$) such that $S \gg_k M$.

We prove both by induction on $k$.

When $k = 0$, both results hold trivially.

When $k > 0$,

1. Suppose $S$ is a bitstream of observation depth $k$, then $S = 0S'$ or $S = 1S'$ where $S'$ is a bitstream of observation depth $k-1$, by IH, there exists unique $M'$ such that $S' \gg_{k-1} M'$, and $\cdot \vdash^{[-,-,k-1]} M' \Leftarrow bitstream$.

If $S = 1S'$, we let $M = b1\, M'$, we have $S \gg_k M$ by definition, and $\cdot \vdash^{[-,-,l]} b1\, M' \Leftarrow bitstream$ by rule. Uniqueness follows by the induction hypothesis and cases on $\gg_k$.

The case for $S = 0S'$ is similar.

2. when $\cdot \vdash^{[-,-,k]} M \Leftarrow bitstream$, we see that $M$ cannot be variable and has to be $c\, M_1 \ldots M_n$, by typing rules, $c$ must have type $\Pi x_1 : P_1. \ \ldots \ \Pi x_n : P_n.bitstream$ (note that the head of the atomic family doesn't participate in substitution).

Thus, $c$ is either $b0$ and $b1$ and $M$ is either $b0\, M'$ or $b1\, M'$, with $\cdot \vdash^{[-,-,k-1]} M' \Leftarrow bitstream$.

By IH, there exists a unique bitstream up to depth $k-1$ $S'$ such that $S' \gg_{k-1} M'$.

If $M = b1\, M'$, then we let $S = 1S'$, we have $S \gg_k M$ by definition, and moreover $S$ is a bitstream by definition. Uniqueness follows by the induction hypothesis and cases on $\gg_k$.

The case for $M = b0\, M'$ is similar.

□

Operations on bitstreams can be defined using coinduction. For instance, the relation $flip(A, B)$ on bitstream may be defined by coinduction:

$$\frac{flip(S, S')}{flip(0S, 1S')} \qquad \frac{flip(S, S')}{flip(1S, 0S')}$$

As usual, the coinduction definition can be written by induction on the observational depth:

$$\frac{flip_k(S,S')}{flip_{k+1}(0S,1S')} \qquad \frac{flip_k(S,S')}{flip_{k+1}(1S,0S')} \qquad \frac{}{flip_0(S,S')}$$

To encode this in $\mathbf{SemTT^{DT}}$, we first encode the judgment flip indexed by two bitstream objects, then encode the rules which are general in bitstream $S$ and $S'$.

$flip$ : $bitstream \to bitstream \to cotype^0$.

$flip0$ : $\Pi S : bitstream.\ \Pi S' : bitstream.\ flip\, S\, S' \to flip\, (b0\, S)\, (b1\, S')$.

$flip1$ : $\Pi S : bitstream.\ \Pi S' : bitstream.\ flip\, S\, S' \to flip\, (b1\, S)\, (b0\, S')$.

We define the encoding $\gg$ as usual:

- $\dfrac{}{flip_0(S,S')} \gg_0 M$ always
- $\dfrac{\overset{\mathcal{D}}{flip_k(S,S')}}{flip_{k+1}(0S,1S')} \gg_{k+1} flip0\, M_S\, M'_S\, M_D$ if $\overset{\mathcal{D}}{flip_k(S,S')} \gg_k M_D$, $S \gg M_S$, $S' \gg M'_S$
- $\dfrac{\overset{\mathcal{D}}{flip_k(S,S')}}{flip_{k+1}(1S,0S')} \gg_{k+1} flip0\, M_S\, M'_S\, M_D$ if $\overset{\mathcal{D}}{flip_k(S,S')} \gg_k M_D$, $S \gg M_S$, $S' \gg M'_S$

**Theorem 6.2.3** (Adequacy of Encoding)**.** *For any stream $S$ and $S'$, there is a bijection between (infinite) derivations ending in $flip(S,S')$ and canonical forms of the type $flip\, M_S\, M'_S$, where $S \gg M_S$ and $S' \gg M'_S$.*

*Proof.* We show a series of lemmas, that entail the original theorem in the end.

1. $\Rightarrow$: If $\overset{\mathcal{D}}{flip_k(S,S')}$, then there exists unique $M$ such that, $\overset{\mathcal{D}}{flip_k(S,S')} \gg_k M$ and $\cdot \vdash^{[k,-,j]} M \Leftarrow flip\, M_S\, M'_S$ for all $j$, where $S \gg M_S$ and $S' \gg M'_S$.

Given a derivation $\overset{\mathcal{D}}{flip_k(S,S')}$,

we want to show that there exists $M$ such that, $\overset{\mathcal{D}}{flip_k(S,S')} \gg M$ and $\cdot \vdash^{[k,-,j]} M \Leftarrow flip\, M_S\, M'_S$ for all $j$, where $S \gg M_S$ and $S' \gg M'_S$.

By induction on $k$, $k = 0$ is trivial, then suppose $k > 0$,

then by cases, $\overset{\mathcal{D}}{flip_k(S,S')}$ is either $\dfrac{\overset{\mathcal{E}}{flip_{k-1}(S_1,S'_1)}}{flip_k(0S_1,1S'_1)}$ or $\dfrac{\overset{\mathcal{E}}{flip_{k-1}(S_1,S'_1)}}{flip_k(1S_1,0S'_1)}$. Suppose it is former, then by IH on $\overset{\mathcal{E}}{flip_{k-1}(S_1,S'_1)}$ we have $M'$ where $\overset{\mathcal{E}}{flip_{k-1}(S_1,S'_1)} \gg_{k-1} M'$, and $\cdot \vdash^{[k-1,-,j]} M' \Leftarrow flip\, M_{S_1}\, M'_{S'_1}$ for all $j$, where $S_1 \gg M_{S_1}$ and $S'_1 \gg M'_{S'_1}$.

By rule, we have $S = 0S_1 \gg b0\, M_{S_1} = M_S$ and $S' = 1S'_1 \gg b1\, M'_{S'_1} = M'_{S'}$, which follows from the correctness of step indexing.

Thus, let $M = flip\,(b0\, M_{S_1})\,(b1\, M'_{S'_1})\, M'$.

By rule, we have $\overset{\mathcal{D}}{flip_k}(S, S') = \dfrac{\overset{\mathcal{E}}{flip_{k-1}}(S_1, S'_1)}{flip_k(0S_1, 1S'_1)} \gg_k M$, and we check (for all $j$)

(1) $flip0 : \Pi S_1 : bitstream.\ \Pi S'_1 : bitstream.\ flip\, S_1\, S'_1 \to flip\,(b0\, S_1)\,(b1\, S'_1)$.

(2) $[M_{S_1}, M'_{S'_1}/S_1, S'_1] flip\,(b0\, S_1)\,(b1\, S'_1) = flip\, M_S\, M'_{S'}$

(3) $\cdot \vdash^{[k-1,-,j]} M_S \Leftarrow bitstream$ by adequacy for bitstream

(4) $\cdot \vdash^{[k-1,-,j]} M'_{S'} \Leftarrow bitstream$

(5) $\cdot \vdash^{[k-1,-,j]} M' \Leftarrow flip\, M_{S_1}\, M'_{S'_1}$

Then by rule we have $\cdot \vdash^{[k,-,j]} M \Leftarrow flip\, M_S\, M'_{S'}$.

The other case is analogous.

2. $\Leftarrow$: For all $j$, if $\cdot \vdash^{[k,-,j]} M \Leftarrow flip\, M_S\, M'_{S'}$ then there exists unique derivation $\overset{\mathcal{D}}{flip_k}(S, S')$, where $S \gg M_S$ and $S' \gg M'_{S'}$, such that $\overset{\mathcal{D}}{flip_k}(S, S') \gg_k M$.

Given $\cdot \vdash^{[k,-,j]} M \Leftarrow flip\, M_S\, M'_{S'}$,

By induction on $k$. $k = 0$ is trivial. Now suppose $k > 0$.

Then by observation, $M$ must be $c\, M_1\, \ldots\, M_n$, where $c : \Pi x_1 : P_1.\ \ldots \Pi x_n : P_n.P \in \Sigma$, where $[M_1, \ldots, M_n/x_1, \ldots, x_n]P = flip\, M_S\, M'_{S'}$.

Since substitution doesn't change the head of the family, we know that head of the family is $flip$ and $c$ must be either $flip0$ or $flip1$.

If $c$ is flip0, $M = c\, M_1\, M_2\, M_3$

Since

$$flip0 : \Pi S_1 : bitstream.\ \Pi S'_1 : bitstream.\ flip\, S_1\, S'_1 \to flip\,(b0\, S_1)\,(b1\, S'_1).$$

By inversion on the derivation,

we have (1) $[M_1, M_2/S_1, S'_1](flip\,(b0\, S_1)\,(b1\, S'_1)) = flip\, M_S\, M'_{S'}$. Then $M_S = b0\, M_1$ and $M'_{S'} = b1\, M_2$

(2) $\cdot \vdash^{[k-1,-,j]} M_3 \Leftarrow flip\, M_1\, M_2$

By IH, there exists a derivation $\overset{\mathcal{E}}{flip_{k-1}}(S_1, S'_1)$ such that $S_1 \gg M_1$, $S'_1 \gg M_2$ and $\overset{\mathcal{E}}{flip_{k-1}}(S_1, S'_1) \gg_{k-1} M_3$.

Let $S = 0S_1$ and $S' = 0S'_1$, by properties of $\gg$, we have $S \gg M_S$ and $S' \gg M'_{S'}$.

Let $\mathcal{D}$ be $\dfrac{\overset{\mathcal{E}}{flip_{k-1}}(S_1, S'_1)}{flip_k(S, S')}$, i.e. $\dfrac{\overset{\mathcal{E}}{flip_{k-1}}(S_1, S'_1)}{flip(0S_1, 1S'_1)}$,

and we check that indeed $\dfrac{\overset{\mathcal{E}}{flip_{k-1}(S_1, S_1')}}{flip(0S_1, 1S_1')} \gg_k flip0\, M_1\, M_2\, M_3 = M$

The case for $c = flip1$ is analogous.

□

# Chapter 7

# Case Study: Subtyping Algorithms of Isorecursive Types

The type equality and subtyping algorithm has been extensively studied in the literature, [AC93, BH98, DA10, LBN17]. It is recognized to be an example of mixed induction and coinduction.

Here, we present a formalization of Ligatti's subtyping system for isorecursive types [LBN17].

## 7.1 The Iso-Recursive Types and Their Representation

Types are divided into two categories. Open types are types with (possibly) free variables, and they are denoted $\bar{\tau}$. Open types $\bar{\tau}$ are the least set of objects generated using the following grammar.

$$\bar{\tau} ::= nat \mid real \mid \bar{\tau}_1 \to \bar{\tau}_2 \mid \bar{\tau}_1 + \bar{\tau}_2 \mid \bar{\tau}_1 \times \bar{\tau}_2 \mid \mu t.\bar{\tau} \mid t$$

Closed types are a subset of open types without free variables, and they are denoted $\tau$.

Since the subtyping algorithm is only considered on closed types, we only consider the set of closed types $\tau$ for encoding into CoLF[1]'s $\mathbf{SynTT^{DT}}$ (and the sound and complete $\mathbf{SemTT^{DT}_{R}}$). We identify isorecusive types with their appropriate unfolding, in such a way that the "iso" aspect is preserved of the isorecursive types.

We rectify the types to their infinite representation. In particular, we add a constructor $\mu$, $\tau ::= \cdots \mid \mu\tau$, and define the type expansion on closed types by structural induction,

$exp_{k+1}(\mu t.\tau) = \mu([exp_k(\mu t.\tau)/t]exp_{k+1}(\tau))$. For example, a type $\mu t.1 + t$ is viewed as the infinitary rational term $T = \mu(1 + T) = \mu(1 + \mu(1 + \mu(\dots)))$.

We define the following signature:

$$
\begin{array}{lcl}
tp & : & type. \\
tp\mu & : & cotype. \\
nat & : & tp. \\
real & : & tp. \\
arr & : & tp \to tp \to tp. \\
sum & : & tp \to tp \to tp. \\
prod & : & tp \to tp \to tp. \\
from\mu & : & tp\mu \to tp. \\
\mu & : & tp \to tp\mu.
\end{array}
$$

Because we're working in **SynTT**, $tp\mu$ is assumed to have higher priority (lower priority number) than $tp$,

We have the following encoding relation:

$\ulcorner - \urcorner$ sends an object from $\tau$ to a term of type $tp$, defined as follows:

$$
\begin{array}{lcl}
\ulcorner nat \urcorner & = & nat \\
\ulcorner real \urcorner & = & real. \\
\ulcorner \tau_1 \to \tau_2 \urcorner & = & arr \ulcorner \tau_1 \urcorner \ulcorner \tau_2 \urcorner. \\
\ulcorner \tau_1 + \tau_2 \urcorner & = & plus \ulcorner \tau_1 \urcorner \ulcorner \tau_2 \urcorner. \\
\ulcorner \tau_1 \times \tau_2 \urcorner & = & prod \ulcorner \tau_1 \urcorner \ulcorner \tau_2 \urcorner. \\
\ulcorner \mu\tau \urcorner & = & from\mu\,(\mu\,(\ulcorner \tau \urcorner))
\end{array}
$$

We also have the following equality

$$\ulcorner \mu t.\tau \urcorner = \mathsf{fix}\, t.from\mu\,(\mu\,(\ulcorner \tau \urcorner))$$

where $\ulcorner t \urcorner = t$

We note that the constructors $from\mu$ and $\mu$ always go together because of the design of our systems.

**Theorem 7.1.1** (Adequacy of Encoding)**.** $\ulcorner - \urcorner$ *is a bijection between closed types and rational terms of type tp (up to term unfolding).*

*Proof.* Formally this is provable using a depth indexing argument, by establishing a correspondence between the depth of the isorecursive type (the number of $\mu$'s we're allowed to observe under) and the depth of the type $tp\mu$.

That is, we prove that there is a bijection between types $\tau$ with $\mu$-depth (the number of $\mu's$ we're allowed to observe under) $k$ and $M = \ulcorner \tau \urcorner$ such that $\vdash_{\Sigma}^{[k,j]} \ulcorner \tau \urcorner \Leftarrow tp$ (for some

$j$), by induction on $k$.

Then since $\vdash_{\Sigma}^{[k,j]} \ulcorner \tau \urcorner \Leftarrow tp$ implies $\vdash_{\Sigma} \ulcorner \tau \urcorner \Leftarrow tp$, we have the adequacy.

□

## 7.2 Encoding of the Emptiness Proof

The emptiness predicate checks whether a type is empty (that is, when there are no values of that type). Ligatti et al. [LBN17] gave an inductive proof with an explicit context. Such proof can be directly encoded coinductively (without a context). The rules of Ligatti's system are copied below.

$\boxed{U \vdash \mathsf{val}(\tau) = \varnothing}$

$$\frac{U \vdash \mathsf{val}(\tau_1) = \varnothing \qquad U \vdash \mathsf{val}(\tau_2) = \varnothing}{U \vdash \mathsf{val}(\tau_1 + \tau_2) = \varnothing}\text{U-SUM} \qquad \frac{U \vdash \mathsf{val}(\tau_1) = \varnothing}{U \vdash \mathsf{val}(\tau_1 \times \tau_2) = \varnothing}\text{U-PROD-1}$$

$$\frac{U \vdash \mathsf{val}(\tau_2) = \varnothing}{U \vdash \mathsf{val}(\tau_1 \times \tau_2) = \varnothing}\text{U-PROD-2} \qquad \frac{\mu t.\bar{\tau} \in U}{U \vdash \mathsf{val}(\mu t.\bar{\tau}) = \varnothing}\text{U-REC-1}$$

$$\frac{U \cup \{\mu t.\bar{\tau}\} \vdash \mathsf{val}([\mu t.\bar{\tau}/t]\bar{\tau}) = \varnothing}{U \vdash \mathsf{val}(\mu t.\bar{\tau}) = \varnothing}\text{U-REC-2}$$

The emptiness proof can be directly encoded coinductively, and thus removing the need for an explicit context. We append the following to the signature:

$$\begin{array}{lcl}
\mathit{empty} & : & \mathit{tp} \to \mathit{cotype}. \\
\mathit{empty/usum} & : & \Pi\tau_1 : \mathit{tp}.\Pi\tau_2 : \mathit{tp}.\mathit{empty}\,\tau_1 \to \mathit{empty}\,\tau_2 \to \mathit{empty}\,(\mathit{sum}\,\tau_1\,\tau_2) \\
\mathit{empty/uprod1} & : & \Pi\tau_1 : \mathit{tp}.\Pi\tau_2 : \mathit{tp}.\mathit{empty}\,\tau_1 \to (\mathit{prod}\,\tau_1\,\tau_2) \\
\mathit{empty/uprod2} & : & \Pi\tau_1 : \mathit{tp}.\Pi\tau_2 : \mathit{tp}.\mathit{empty}\,\tau_2 \to (\mathit{prod}\,\tau_1\,\tau_2) \\
\mathit{empty/urec} & : & \Pi\tau : \mathit{tp}.\mathit{empty}\,\tau \to (\mathit{from}\mu\,(\mu\,\tau))
\end{array}$$

To state adequacy, we first modify the proof to act on infinitary type expressions. Specifically, we replace U-REC-1 and U-REC-2 with the following rule U-REC, and let the resulting proof system be infinitary. Since the terms are rational and the rules are syntax directed, the resulting system is also a rational system. After the removal of the U-REC-1 rule and the U-REC-2 rule, the context will always be empty.

$$\frac{\cdot \vdash \mathsf{val}(\tau) = \varnothing}{\cdot \vdash \mathsf{val}(\mu(\tau)) = \varnothing}\text{U-REC}$$

Then we establish the following encoding relation (as a coinductive definition).

1. $\ulcorner \dfrac{\cdot \vdash \mathsf{val}(\tau_1) = \varnothing \qquad \cdot \vdash \mathsf{val}(\tau_2) = \varnothing}{\cdot \vdash \mathsf{val}(\tau_1 + \tau_2) = \varnothing}\text{U-SUM}\urcorner$
   $= \mathit{empty/usum}\,(\ulcorner\tau_1\urcorner)\,(\ulcorner\tau_2\urcorner)\,(\ulcorner \cdot \vdash \mathsf{val}(\tau_1) = \varnothing\urcorner)\,(\ulcorner \cdot \vdash \mathsf{val}(\tau_1) = \varnothing\urcorner)$
2. $\ulcorner \dfrac{\cdot \vdash \mathsf{val}(\tau_1) = \varnothing}{\cdot \vdash \mathsf{val}(\tau_1 \times \tau_2) = \varnothing}\text{U-PROD-1}\urcorner = \mathit{empty/uprod1}\,(\ulcorner\tau_1\urcorner)\,(\ulcorner\tau_2\urcorner)\,(\ulcorner \cdot \vdash \mathsf{val}(\tau_1) = \varnothing\urcorner)$
3. $\ulcorner \dfrac{\cdot \vdash \mathsf{val}(\tau_2) = \varnothing}{\cdot \vdash \mathsf{val}(\tau_1 \times \tau_2) = \varnothing}\text{U-PROD-2}\urcorner = \mathit{empty/uprod2}\,(\ulcorner\tau_1\urcorner)\,(\ulcorner\tau_2\urcorner)\,(\ulcorner \cdot \vdash \mathsf{val}(\tau_2) = \varnothing\urcorner)$
4. $\ulcorner \dfrac{\cdot \vdash \mathsf{val}(\tau) = \varnothing}{\cdot \vdash \mathsf{val}(\mu(\tau)) = \varnothing}\text{U-REC}\urcorner = \mathit{empty/urec}\,(\ulcorner\tau\urcorner)\,(\ulcorner \cdot \vdash \mathsf{val}(\tau) = \varnothing\urcorner)$

**Theorem 7.2.1** (Adequacy)**.** *For any isorecursive type $\tau$, there is a bijection between the proof that $\tau$ is empty and the canonical terms of type empty $(\ulcorner\tau\urcorner)$.*

*Proof.* Both proofs and terms are infinitary and rational. We can establish a correspondence between an infinitary proof of depth $k$ and the derivation $\vdash^{[k,-,l,j]} M \Leftarrow \mathit{empty}\,(\ulcorner\tau\urcorner)$ (where $l$ is the depth index of type $tp\mu$, $j$ is the depth index of type $tp$). Then the bijection follows from this correspondence. □

## 7.3 Encoding of the Subtyping Proof

The subtyping rules are copied below.

$\boxed{S \vdash \tau \leq \tau'}$

$$\frac{}{S \vdash \mathit{nat} \leq \mathit{real}}\text{S-BASE} \qquad \frac{}{S \vdash \mathit{nat} \leq \mathit{nat}}\text{S-NAT} \qquad \frac{}{S \vdash \mathit{real} \leq \mathit{real}}\text{S-REAL}$$

$$\frac{\mathsf{val}(\tau) = \varnothing}{S \vdash \tau \leq \tau'}\text{S-}\bot \qquad \frac{\mathsf{val}(\tau_1') = \varnothing}{S \vdash \tau \leq \tau_1' \to \tau_2'}\text{S-}\top \qquad \frac{S \vdash \tau_1' \leq \tau_1 \qquad S \vdash \tau_2 \leq \tau_2'}{S \vdash \tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'}\text{S-FUN}$$

$$\frac{S \vdash \tau_1 \leq \tau_1' \qquad S \vdash \tau_2 \leq \tau_2'}{S \vdash \tau_1 + \tau_2 \leq \tau_1' + \tau_2'}\text{S-SUM} \qquad \frac{S \vdash \tau_1 \leq \tau_1' \qquad S \vdash \tau_2 \leq \tau_2'}{S \vdash \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'}\text{S-PROD}$$

$$\frac{\mu t.\bar{\tau} \leq \mu t'.\bar{\tau}' \in S}{S \vdash \mu t.\bar{\tau} \leq \mu t'.\bar{\tau}'}\text{S-REC-1} \qquad \frac{S \cup \{\mu t.\tau \leq \mu t'.\tau'\} \vdash [\mu t.\bar{\tau}/t]\bar{\tau} \leq [\mu t'.\bar{\tau}'/t']\bar{\tau}'}{S \vdash \mu t.\bar{\tau} \leq \mu t'.\bar{\tau}'}\text{S-REC-2}$$

We can again modify the proof to act on infinitary type expressions by combining the rule S-REC-1 and the rule S-REC-2 into one rule. We also note that the context $S$ will always be empty after the rule S-REC-1 and the rule S-REC-2 are removed.

$$\frac{\cdot \vdash \tau \leq \tau'}{\cdot \vdash \mu(\tau) \leq \mu(\tau')}\text{S-REC}$$

We can encode the above proof rules into the following $\mathbf{SynTT^{DT}}$ signature.

$subtp \quad : \quad tp \to tp \to cotype.$

$sbase \quad : \quad subtp\, nat\, real.$

$snat \quad : \quad subtp\, nat\, nat.$

$sreal \quad : \quad subtp\, real\, real.$

$s\bot \quad : \quad \Pi t : tp.\Pi t' : tp.empty\, t \to subtp\, t\, t'.$

$s\top \quad : \quad \Pi t : tp.\Pi t'_1 : tp.\Pi t'_2 : tp.empty\, t'_1 \to subtp\, t\, (arr\, t'_1\, t'_2).$

$sfun \quad : \quad \Pi t_1 : tp.\Pi t_2 : tp.\Pi t'_1 : tp.\Pi t'_2 : tp.subtp\, t'_1\, t_1 \to subtp\, t_2\, t'_2 \to subtp\, (arr\, t_1\, t_2)\, (arr\, t'_1\, t'_2).$

$ssum \quad : \quad \Pi t_1 : tp.\Pi t_2 : tp.\Pi t'_1 : tp.\Pi t'_2 : tp.subtp\, t_1\, t'_1 \to subtp\, t_2\, t'_2 \to subtp\, (sum\, t_1\, t_2)\, (sum\, t'_1\, t'_2).$

$sprod \quad : \quad \Pi t_1 : tp.\Pi t_2 : tp.\Pi t'_1 : tp.\Pi t'_2 : tp.subtp\, t_1\, t'_1 \to subtp\, t_2\, t'_2 \to subtp\, (prod\, t_1\, t_2)\, (prod\, t'_1\, t'_2).$

$srec \quad : \quad \Pi t : tp.\Pi t' : tp.subtp\, t\, t' \to subtp\, (\mu\, t)\, (\mu\, t')$

We have the following proof encoding relations.

1. $\ulcorner \dfrac{}{\cdot \vdash nat \leq real}\text{S-BASE}\urcorner = sbase$
2. $\ulcorner \dfrac{}{\cdot \vdash nat \leq nat}\text{S-NAT}\urcorner = snat$
3. $\ulcorner \dfrac{}{\cdot \vdash real \leq real}\text{S-REAL}\urcorner = sreal$
4. $\ulcorner \dfrac{\mathsf{val}(\tau) = \varnothing}{\cdot \vdash \tau \leq \tau'}\text{S-}\bot\urcorner = s\bot\, (\ulcorner \tau \urcorner)\, (\ulcorner \tau' \urcorner)\, (\ulcorner \mathsf{val}(\tau) = \varnothing \urcorner)$
5. $\ulcorner \dfrac{\mathsf{val}(\tau'_1) = \varnothing}{\cdot \vdash \tau \leq \tau'_1 \to \tau'_2}\text{S-}\top\urcorner = s\top\, (\ulcorner \tau \urcorner)\, (\ulcorner \tau'_1 \urcorner)\, (\ulcorner \tau'_2 \urcorner)\, (\ulcorner \mathsf{val}(\tau'_1) = \varnothing \urcorner)$
6. $\ulcorner \dfrac{\cdot \vdash \tau'_1 \leq \tau_1 \qquad \cdot \vdash \tau_2 \leq \tau'_2}{\cdot \vdash \tau_1 \to \tau_2 \leq \tau'_1 \to \tau'_2}\text{S-FUN}\urcorner$
   $= sfun\, (\ulcorner \tau_1 \urcorner)\, (\ulcorner \tau_2 \urcorner)\, (\ulcorner \tau'_1 \urcorner)\, (\ulcorner \tau'_2 \urcorner)\, (\ulcorner \cdot \vdash \tau'_1 \leq \tau_1 \urcorner)\, (\ulcorner \cdot \vdash \tau_2 \leq \tau'_2 \urcorner)$
7. $\ulcorner \dfrac{\cdot \vdash \tau_1 \leq \tau'_1 \qquad \cdot \vdash \tau_2 \leq \tau'_2}{\cdot \vdash \tau_1 + \tau_2 \leq \tau'_1 + \tau'_2}\text{S-SUM}\urcorner$
   $= ssum\, (\ulcorner \tau_1 \urcorner)\, (\ulcorner \tau_2 \urcorner)\, (\ulcorner \tau'_1 \urcorner)\, (\ulcorner \tau'_2 \urcorner)\, (\ulcorner \cdot \vdash \tau_1 \leq \tau'_1 \urcorner)\, (\ulcorner \cdot \vdash \tau_2 \leq \tau'_2 \urcorner)$

8. $\ulcorner \dfrac{\cdot \vdash \tau_1 \leq \tau_1' \qquad \cdot \vdash \tau_2 \leq \tau_2'}{\cdot \vdash \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'} \text{S-PROD} \urcorner$
$= \mathit{sprod}\,(\ulcorner \tau_1 \urcorner)\,(\ulcorner \tau_2 \urcorner)\,(\ulcorner \tau_1' \urcorner)\,(\ulcorner \tau_2' \urcorner)\,(\ulcorner \cdot \vdash \tau_1 \leq \tau_1' \urcorner)\,(\ulcorner \cdot \vdash \tau_2 \leq \tau_2' \urcorner)$
9. $\ulcorner \dfrac{\cdot \vdash \tau \leq \tau'}{\cdot \vdash \mu(\tau) \leq \mu(\tau')} \text{S-REC} \urcorner = \mathit{srec}\,(\ulcorner \tau \urcorner)\,(\ulcorner \tau' \urcorner)\,(\ulcorner \cdot \vdash \tau \leq \tau' \urcorner)$

**Theorem 7.3.1** (Adequacy)**.** *For any type $\tau_1$ and $\tau_2$, there is a bijection between the subtyping proof and canonical terms of the type $\mathit{sub}\,(\ulcorner \tau_1 \urcorner)\,(\ulcorner \tau_2 \urcorner)$*

*Proof.* We recognize that both sides to be infinitary and rational. We can directly establish a correspondence between a subtyping proof of depth $k$ and the derivation $\vdash^{[k,-,u,-,l,j]} M \Leftarrow \mathit{subtp}\,(\ulcorner \tau_1 \urcorner)\,(\ulcorner \tau_2 \urcorner)$, where $k$ is the index for type *sub*, $u$ is the index for type *empty*, $l$ is the index for type $tp\mu$ and $j$ is the index for type *tp*. □

# Chapter 8

# Case Study: Circular Proof Systems with Fixed Points

Fortier and Santocanale [FS13] presented a circular proof system with mixed inductive and coinductive fixed points. We explore to what extent the system is encodable in CoLF$^1$. Specifically, we report that the cut-free purely inductive fragment and the purely cut-free coinductive fragment are representable. We encountered some difficulties when trying to encode the cut-free mixed inductive and coinductive fragment. We think that further work is needed in order to encode the full system.

Fortier and Santocanale's system is parametrized by a system of equations, $\{x =_{\rho(\pi(x))} \tau(x)\}_{x \in X}$, where $X$ is a set of variables, $\pi$ is the priority function, $\rho$ maps even priorities to the symbol $\nu$ and odd priorities to the symbol $\mu$, and $\tau$ maps a type variable to a concrete type. Therefore, we also parametrize CoLF$^1$ signature by the same system of equations.

## 8.1 Encoding of the Terms

Terms are given by the following grammar.

$$t ::= x \mid t_1 + t_2 \mid t_1 \times t_2 \mid 0 \mid 1$$

Since the terms in this system is finite, we can encode the terms as follows.

$$
\begin{array}{lcl}
Var & : & type. \\
\{x & : & Var\}_{x \in X} \\
term & : & type. \\
var & : & Var \to term. \\
sum & : & term \to term \to term. \\
prod & : & term \to term \to term. \\
0 & : & term. \\
1 & : & term.
\end{array}
$$

We have the following encoding relation.

$$
\begin{array}{lcl}
\ulcorner x \urcorner & = & Var\, x \\
\ulcorner 0 \urcorner & = & 0 \\
\ulcorner 1 \urcorner & = & 1 \\
\ulcorner t_1 + t_2 \urcorner & = & sum\, (\ulcorner t_1 \urcorner)\, (\ulcorner t_2 \urcorner) \\
\ulcorner t_1 \times t_2 \urcorner & = & prod\, (\ulcorner t_1 \urcorner)\, (\ulcorner t_2 \urcorner)
\end{array}
$$

**Theorem 8.1.1** (Adequacy of Encoding)**.** *There is a compositional bijection between the informal terms and canonical forms of the type term.*

*Proof.* By depth shift and depth weakening, it is sufficient to show

(1) If $t$ is a term, then there exists a unique term $M = \ulcorner t \urcorner$, and $\vdash^{[-,j,-,1]} M \Leftarrow term$ for some $j$.

(2) If for some $j$, $\vdash^{[-,j,-,1]} M \Leftarrow term$, then there exists a unique term $t$ such that $\ulcorner t \urcorner = M$.

To show (1), we proceed by induction on $t$.

Case $t = 0, 1, x$, we let $j = 1$, and we directly have the corresponding encoding. Uniqueness follows by cases on the translation.

Case $t = t_1 + t_2$, by IH, we have $\vdash^{[-,j_1,-,1]} \ulcorner t_1 \urcorner \Leftarrow term$ for some $j_1$, $\vdash^{[-,j_2,-,1]} \ulcorner t_1 \urcorner \Leftarrow term$ for some $j_2$. We let $j = max(j_1, j_2)+1$, and we have $\vdash^{[-,j,-,1]} \ulcorner t \urcorner \Leftarrow term$. Uniqueness follows from the induction hypothesis.

The case or $t = t_1 \times t_2$ is analogous.

To show (2), we proceed by induction on the derivation $\vdash^{[-,j,-,1]} M \Leftarrow term$.

By rules, $j$ must be sufficiently large such that for any subderivation $\vdash^{D} M \Leftarrow term$, $D(1) > 0$.

$M$ must be of the form $c\, M_1\, M_n$, and $c : P_1 \to \ldots P_n \to term$. Then $c$ can only be $var$, $sum$, $prod$, 0, 1.

1. Case $c = var$, then $\vdash^{[-,j-1,-,1]} M_1 \Leftarrow Var$, and by similar arguments, $M_1$ must be one of the variables, $x$, then the resulting term $t = x$.

2. Case $c = 0$, then $t = 0$.
3. Case $c = 1$, then $t = 1$.
4. Case $c = sum$, then $\vdash^{[-,j-1,-,1]} M_1 \Leftarrow term$, $\vdash^{[-,j-1,-,1]} M_2 \Leftarrow term$, by induction hypothesis, there exists $t_1$, $t_2$, such that $\ulcorner t_1 \urcorner = M_1$ and $\ulcorner t_2 \urcorner = M_2$. Then $t = t_1 + t_2$,
5. the case for $c = prod$ is exactly similar.

□

## 8.2 Encoding of the Proof Rules

Because Fortier and Santocanale's system admits cut elimination, and we encountered a number of difficulties when trying to represent the cut rule. In particular, the cut rule introduces the complications into the validity condition that are difficult to encode with our current design of CoLF[1]. We choose to represent the cut-free fragment. The cut-free fragment of the system consists of the following basic proof rules:

$$\frac{}{t \vdash t}\mathrm{Id}$$

$$\frac{s_0 \vdash t}{s_0 \times s_1 \vdash t}\mathrm{L}\times_0 \qquad \frac{s_1 \vdash t}{s_0 \times s_1 \vdash t}\mathrm{L}\times_1 \qquad \frac{s \vdash t_0 \qquad s \vdash t_1}{s \vdash t_0 \times t_1}\mathrm{R}\times \qquad \frac{}{t \vdash 1}\mathrm{RAx}$$

$$\frac{}{0 \vdash t}\mathrm{LAx} \qquad \frac{s_0 \vdash t \qquad s_1 \vdash t}{s_0 + s_1 \vdash t}\mathrm{L}+ \qquad \frac{s \vdash t_0}{s \vdash t_0 + t_1}\mathrm{R}+_0 \qquad \frac{s \vdash t_1}{s \vdash t_0 + t_1}\mathrm{R}+_1$$

The inductive fragment contains the following two additional rules.

$$\frac{\tau(x) \vdash t}{x \vdash t}\mathrm{L}\mu x \qquad \frac{s \vdash \tau(x)}{s \vdash x}\mathrm{R}\mu x$$

The coinductive fragment contains the following two additional rules.

$$\frac{\tau(x) \vdash t}{x \vdash t}\mathrm{L}\nu x \qquad \frac{s \vdash \tau(x)}{s \vdash x}\mathrm{R}\nu x$$

The validity condition states that any cycle in the infinitary rational derivation in the cut-free fragment must satisfy one of the following conditions: (1) it contains a left fixed

point rule and the highest priority (the lowest number) of the left fixed point rules is odd or (2) it contains a right fixed point rule and the highest priority (the lowest number) of the right fixed point rules is even. When restricted to the purely inductive systems, only condition (1) applies and when restricted to the purely coinductive systems, only condition (2) applies.

We then encode the provability of $s \vdash t$ as follows.

We first encode the common clauses of both the inductive and the coinductive fragment.

$\Sigma_0 =$

$$
\begin{array}{lcl}
seq & : & term \to term \to type. \\
Id & : & \Pi t : term. seq\, t\, t. \\
L\times_0 & : & \Pi s_0 : term. \Pi s_1 : term. \Pi t : term. seq\, s_0\, t \to seq\, (prod\, s_0\, s_1)\, t. \\
L\times_1 & : & \Pi s_0 : term. \Pi s_1 : term. \Pi t : term. seq\, s_1\, t \to seq\, (prod\, s_0\, s_1)\, t. \\
R\times & : & \Pi s : term. \Pi t_0 : term. \Pi t_1 : term. seq\, s\, t_0 \to seq\, s\, t_1 \to seq\, s\, (prod\, t_0\, t_1). \\
RAx & : & \Pi t : term. seq\, t\, 1. \\
LAx & : & \Pi t : term. seq\, 0\, t. \\
L+ & : & \Pi s_0 : term. \Pi s_1 : term. \Pi t : term. seq\, s_0\, t \to seq\, s_1\, t \to seq\, (sum\, s_0\, s_1)\, t. \\
R+_0 & : & \Pi s : term. \Pi t_0 : term. \Pi t_1 : term. seq\, s\, t_0 \to seq\, s\, (sum\, t_0\, t_1). \\
R+_1 & : & \Pi s : term. \Pi t_0 : term. \Pi t_1 : term. seq\, s\, t_1 \to seq\, s\, (sum\, t_0\, t_1).
\end{array}
$$

For the purely inductive fragment, we append the following clause. Note that both $lseq$'s and $rseq$'s have higher priorities (lower priority number) than $seq$. In order to represent different priorities, we have one new type per fixed point definition.

$\Sigma_1 =$

$$
\begin{array}{lcl}
\{R\mu x & : & \Pi s : term. seq\, s\, (\ulcorner \tau(x) \urcorner) \to seq\, s\, (Var\, x).\}_{x \in X} \\
\{lseq_x & : & term \to term \to cotype^{\pi(x)}.\}_{x \in X} \\
\{shiftL_x & : & \Pi s : term. \Pi t : term. lseq_x\, s\, t \to seq\, s\, t.\}_{x \in X} \\
\{L\mu x & : & \Pi t : term. seq\, (\ulcorner \tau(x) \urcorner)\, t \to lseq\, (Var\, x)\, t.\}_{x \in X}
\end{array}
$$

For the purely coinductive fragment, we append the following clause

$\Sigma_2 =$

$$
\begin{array}{lcl}
\{L\nu x & : & \Pi t : term. seq\, (\ulcorner \tau(x) \urcorner)\, t \to seq\, (Var\, x)\, t.\}_{x \in X} \\
\{rseq_x & : & term \to term \to cotype^{\pi(x)}.\}_{x \in X} \\
\{shiftR_x & : & \Pi s : term. \Pi t : term. rseq_x\, s\, t \to seq\, s\, t.\}_{x \in X} \\
\{R\nu x & : & \Pi s : term. seq\, s\, (\ulcorner \tau(x) \urcorner) \to rseq\, s\, (Var\, x).\}_{x \in X}
\end{array}
$$

That is, the signature of the proof system encoding for the purely inductive fragment consists of clauses $\Sigma_0 \cup \Sigma_1$, and for the purely coinductive fragment consists of clauses $\Sigma_0 \cup \Sigma_2$.

1. $\ulcorner \dfrac{}{t \vdash t}\mathrm{Id}\urcorner = Id\,(\ulcorner t \urcorner)$

2. $\ulcorner \dfrac{s_0 \vdash t}{s_0 \times s_1 \vdash t}\mathrm{L}\times_0\urcorner = L \times_0\ (\ulcorner s_0 \urcorner)\,(\ulcorner s_1 \urcorner)\,(\ulcorner t \urcorner)\,(\ulcorner s_0 \vdash t \urcorner)$

3. $\ulcorner \dfrac{s_1 \vdash t}{s_0 \times s_1 \vdash t}\mathrm{L}\times_1\urcorner = L \times_1\ (\ulcorner s_0 \urcorner)\,(\ulcorner s_1 \urcorner)\,(\ulcorner t \urcorner)\,(\ulcorner s_1 \vdash t \urcorner)$

4. $\ulcorner \dfrac{s \vdash t_0 \qquad s \vdash t_1}{s \vdash t_0 \times t_1}\mathrm{R}\times\urcorner = R \times\ (\ulcorner s \urcorner)\,(\ulcorner t_0 \urcorner)\,(\ulcorner t_1 \urcorner)\,(\ulcorner s \vdash t_0 \urcorner)\,(\ulcorner s \vdash t_1 \urcorner)$

5. $\ulcorner \dfrac{}{t \vdash 1}\mathrm{RAx}\urcorner = RAx\,(\ulcorner t \urcorner)$

6. $\ulcorner \dfrac{}{0 \vdash t}\mathrm{LAx}\urcorner = LAx\,(\ulcorner t \urcorner)$

7. $\ulcorner \dfrac{s_0 \vdash t \qquad s_1 \vdash t}{s_0 + s_1 \vdash t}\mathrm{L}+\urcorner = L +\ (\ulcorner s_0 \urcorner)\,(\ulcorner s_1 \urcorner)\,(\ulcorner t \urcorner)\,(\ulcorner s_0 \vdash t \urcorner)\,(\ulcorner s_1 \vdash t \urcorner)$

8. $\ulcorner \dfrac{s \vdash t_0}{s \vdash t_0 + t_1}\mathrm{R}+_0\urcorner = R +_0\ (\ulcorner s \urcorner)\,(\ulcorner t_0 \urcorner)\,(\ulcorner t_1 \urcorner)\,(\ulcorner s \vdash t_0 \urcorner)$

9. $\ulcorner \dfrac{s \vdash t_1}{s \vdash t_0 + t_1}\mathrm{R}+_1\urcorner = R +_1\ (\ulcorner s \urcorner)\,(\ulcorner t_0 \urcorner)\,(\ulcorner t_1 \urcorner)\,(\ulcorner s \vdash t_1 \urcorner)$

10. $\ulcorner \dfrac{\tau(x) \vdash t}{x \vdash t}\mathrm{L}\mu x\urcorner = shiftL_x\,(\ulcorner x \urcorner)\,(\ulcorner t \urcorner)\,(L\mu x\,x\,(\ulcorner t \urcorner)\,(\ulcorner \tau(x) \vdash t \urcorner))$

11. $\ulcorner \dfrac{s \vdash \tau(x)}{s \vdash x}\mathrm{R}\mu x\urcorner = R\mu x\,x\,(\ulcorner s \urcorner)\,(\ulcorner s \vdash \tau(x) \urcorner)$

12. $\ulcorner \dfrac{\tau(x) \vdash t}{x \vdash t}\mathrm{L}\nu x\urcorner = L\mu x\,x\,(\ulcorner t \urcorner)\,(\ulcorner \tau(x) \vdash t \urcorner)$

13. $\ulcorner \dfrac{s \vdash \tau(x)}{s \vdash x}\mathrm{R}\nu x\urcorner = shiftR_x\,(\ulcorner x \urcorner)\,(\ulcorner s \urcorner)\,(R\mu x\,x\,(\ulcorner s \urcorner)\,(\ulcorner s \vdash \tau(x) \urcorner))$

**Theorem 8.2.1** (Adequacy)**.** *The encoding of the inductive cut-free fragment and the coinductive cut-free fragment is adequate. That is, given two terms $s$ and $t$, there is a bijection between valid circular proofs $s \vdash t$ of Fortier and Santocanale's system (of each fragment) and the canonical terms of the type $seq\,(\ulcorner s \urcorner)\,(\ulcorner t \urcorner)$.*

*Proof.* The adequacy of signature $\Sigma_0$ is immediate by induction on both sides.

For the inductive fragment, we want to show the proofs of $s \vdash t$ are in one-to-one correspondence to $M = \ulcorner s \vdash t \urcorner$, such that $\vdash {}^{[x_0,-,x_1,-,\ldots,x_n,s,-,j,-,k]}M \Leftarrow seq\,(\ulcorner s \urcorner)\,(\ulcorner t \urcorner)$, where $x_i$ denotes the current index of $i$th variable in the depth context, $s$ denotes the current index of the sequent, $j$ denotes the current index of the type $term$ and $k$ denotes the current index of the type $var$.

If the proofs of Fortier and Santocanale's system are rational, and the encoding is syntax-directed, then the corresponding term of CoLF$^1$ is rational.

For the forward direction, we can directly verify that $M = \ulcorner s \vdash t \urcorner$ satisfies the corresponding typing rules by induction on the number of $L\mu$ rules we could observe under.

For the reverse direction, we do the proof by induction on the lexicographic order of $[x_0, -, x_1, -, \ldots, x_n, s, -, j, -, k]$. In particular, there is exactly one $shiftL$ rule immediately followed by an application of the $L\mu$ rule per variable definition. Once we make the observation, the step context will get smaller, and we can retrieve the corresponding informal proof object from the induction hypothesis and use the retrieved proof to synthesize the goal.

The validity of Fortier and Santocanale's system exactly corresponds to the validity of CoLF$^1$, in the sense that we must encounter a left rule (and thus decreasing the index of one of $x_i$'s), before we can close the derivation.

The case for the coinductive fragment is exactly similar.

□

# Chapter 9

# Conclusion and Future Work

In this thesis, we investigated the preliminary questions in building a mixed inductive and coinductive logical framework. We proposed CoLF$^1$, the first order fragment of a mixed inductive and coinductive logical framework. We iteratively designed a semantic type theory in which all infinitary objects can be adequately represented, a semantic type theory on rational terms where type checking is decidable, and a syntactic type theory which can be easily implemented. Through the case study on the subtyping algorithms of isorecursive types and the circular proof systems with inductive and coinductive definitions, we showed that when dealing with systems that have internal notions of circularity, CoLF$^1$ can adequately encode a decent fragment of those systems and the proof checking is decidable.

This work is only a first step towards a full-fledged mixed inductive and coinductive logical framework. As part of the future work, the proposed type theories can be extended with quantification over function types, automatic proof search capabilities and constructs for representing and proving metatheorems.

# Bibliography


[AC93] Roberto M. Amadio and Luca Cardelli. "Subtyping Recursive Types". In: *ACM Transactions on Programming Languages and Systems* 15.4 (1993), pp. 575–631.

[Ahm04] Amal Jamil Ahmed. "Semantics of Types for Mutable State". PhD thesis. Princeton University, Nov. 2004.

[AM01] Andrew W. Appel and David A. McAllester. "An Indexed Model of Recursive Types for Foundational Proof-Carrying Code". In: *Transactions on Programming Languages and Systems* 23.5 (2001), pp. 657–683.

[BH98] Michael Brandt and Fritz Henglein. "Coinductive Axiomatization of Recursive Type Equality and Subtyping". In: *Fundamenta Informaticae* 33.4 (1998), pp. 309–338.

[Bro05] James Brotherston. "Cyclic Proofs for First-Order Logic with Inductive Definitions". In: *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2005)*. Ed. by B. Beckert. Koblenz, Germany: Springer LNCS 3702, Sept. 2005, pp. 78–92.

[BS11] James Brotherston and Alex Simpson. "Sequent Calculi for Induction and Infinite Descent". In: *Journal of Logic and Computation* 21.6 (2011), pp. 1177–1216.

[Cer+02] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. *A Concurrent Logical Framework II: Examples and Applications*. Tech. rep. CMU-CS-02-102. Revised May 2003. Department of Computer Science, Carnegie Mellon University, 2002.

[Cha+98] Witold Charatonik, David A. McAllester, Damian Niwinski, Andreas Podelski, and Igor Walukiewicz. "The Horn Mu-calculus". In: *Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science (LICS 1998)*. Indianapolis, IN, USA: IEEE Computer Society Press, June 1998, pp. 58–69.

[Coq93] Thierry Coquand. "Infinite Objects in Type Theory". In: *Types for Proofs and Programs, International Workshop TYPES'93, Nijmegen, The Netherlands, May 24-28, 1993, Selected Papers*. Vol. 806. Lecture Notes in Computer Science. Springer, 1993, pp. 62–78.

[Cou83] Bruno Courcelle. "Fundamental Properties of Infinite Trees". In: *Theoretical Computer Science* 25 (1983), pp. 95–169.

[CP96] Iliano Cervesato and Frank Pfenning. "A Linear Logical Framework". In: *Proceedings of the Eleventh Annual Symposium on Logic in Computer Science*. Ed. by E. Clarke. New Brunswick, New Jersey: IEEE Computer Society Press, July 1996, pp. 264–275.

[DA10] Nils Anders Danielsson and Thorsten Altenkirch. "Subtyping, Declaratively". In: *10th International Conference on Mathematics of Program Construction (MPC 2010)*. Québec City, Canada: Springer LNCS 6120, June 2010, pp. 100–118.

[DP19] Farzaneh Derakhshan and Frank Pfenning. "Circular Proofs as Session-Typed Processes: A Local Validity Condition". In: *CoRR* abs/1908.01909 (Aug. 2019). URL: `http://arxiv.org/abs/1908.01909`.

[DP20] Farzaneh Derakhshan and Frank Pfenning. "Circular Proofs in First-Order Linear Logic with Least and Greatest Fixed Points". In: *CoRR* abs/2001.05132 (Jan. 2020). URL: `http://arxiv.org/abs/2001.05132`.

[FS13] Jérôme Fortier and Luigi Santocanale. "Cuts for Circular Proofs: Semantics and Cut-Elimination". In: *22nd Annual Conference on Computer Science Logic (CSL 2013)*. Ed. by Simona Ronchi Della Rocca. Torino, Italy: LIPIcs 23, Sept. 2013, pp. 248–262.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. "A Framework for Defining Logics". In: *Journal of the Association for Computing Machinery* 40.1 (Jan. 1993), pp. 143–184.

[HL07] Robert Harper and Daniel R. Licata. "Mechanizing metatheory in a logical framework". In: *Journal of Functional Programming* 17.4-5 (2007), pp. 613–673.

[HP05] Robert Harper and Frank Pfenning. "On Equivalence and Canonical Forms in the LF Type Theory". In: *Transactions on Computational Logic* 6 (1 Jan. 2005), pp. 61–101.

[LBN17] Jay Ligatti, Jeremy Blackburn, and Michael Nachtigal. "On Subtyping-Relation Completeness, with an Application to Iso-Recursive Types". In: *ACM Transactions on Programming Languages and Systems* 39.4 (Mar. 2017), 4:1–4:36.

[Pfe01] Frank Pfenning. "Logical Frameworks". In: *Handbook of Automated Reasoning (in 2 volumes)*. Ed. by John Alan Robinson and Andrei Voronkov. Elsevier and MIT Press, 2001, pp. 1063–1147.

[Pfe94] Frank Pfenning. "Elf: A Meta-Language for Deductive Systems (System Description)". In: *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings*. Ed. by Alan Bundy. Vol. 814. Lecture Notes in Computer Science. Springer, 1994, pp. 811–815.

[PS98] Frank Pfenning and Carsten Schürmann. *Twelf User's Guide*. 1.2. Available as Technical Report CMU-CS-98-173, Carnegie Mellon University. Sept. 1998.

[PS99] Frank Pfenning and Carsten Schürmann. "System Description: Twelf - A Meta-Logical Framework for Deductive Systems". In: *Automated Deduction - CADE-16, 16th International Conference on Automated Deduction, Trento, Italy, July 7-10, 1999, Proceedings*. Ed. by Harald Ganzinger. Vol. 1632. Lecture Notes in Computer Science. Springer, 1999, pp. 202–206.

[Sch11] Anders Schack-Nielsen. "Implementing Substructural Logical Frameworks". PhD thesis. IT University of Copenhagen, Jan. 2011.

[Sim06] Luke Evans Simon. "Extending logic programming with coinduction". PhD thesis. University of Texas at Dallas, 2006.

[Wat+02] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. *A Concurrent Logical Framework I: Judgments and Properties*. Tech. rep. CMU-CS-02-101. Revised May 2003. Department of Computer Science, Carnegie Mellon University, 2002.