Mode, Reduction, and Termination Analysis for LolliMon

Ruy Ley-Wild

November 4, 2014
CMU-CS-14-141

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

LolliMon [10] is a linear logic programming language that combines backward-chaining, backtracking semantics for the asynchronous connectives and forward-chaining, committed choice for the synchronous connectives. Mode, reduction, and termination properties are important correctness criteria that can be verified automatically of both backward-chaining and forward-chaining logic programs by establishing suitable moding and subterm relationships between a clause's head and its subgoals. This work combines existing techniques for termination checking of backward-chaining higher-order intuitionistic logic programs and complexity analysis of forward-chaining first-order logic programs, by devising mode, reduction, and termination analyses for the linear logic programming setting in LolliMon.

Note: This document was written as a project report for 15-819K Logic Programming taught by Frank Pfenning in the fall of 2006.

Keywords: linear logic, logic programming, mode analysis, reduction analysis, termination analysis

15-819K: Logic Programming

Final Project Report

Mode, Reduction, and Termination Analysis for LolliMon

Ruy Ley-Wild

Abstract

LolliMon [10] is a linear logic programming language that combines backward-chaining, backtracking semantics for the asynchronous connectives and forward-chaining, committed choice for the synchronous connectives. Mode, reduction, and termination properties are important correctness criteria that can be verified automatically of both backward-chaining and forward-chaining logic programs by establishing suitable moding and subterm relationships between a clause's head and its subgoals. This work combines existing techniques for termination checking of backward-chaining higher-order intuitionistic logic programs and complexity analysis of forward-chaining first-order logic programs, by devising mode, reduction, and termination analyses for the linear logic programming setting in LolliMon.

1 Introduction

LolliMon [10] is a logic programming language based on intuitionistic linear logic that combines backward-chaining, backtracking semantics for the asynchronous connectives and forward-chaining, committed choice for the synchronous connectives. The synchronous fragment is syntactically segregated by a monad and is used to transition from backward-chaining to forward-chaining. The forward-chaining phase in LolliMon finishes and returns to backward-chaining search when the linear context is *quiescent* (no forward steps are possible) and the unrestricted context is *saturated* (forward steps may be possible, but do not generate any new information).

Certain correctness properties of LolliMon logic programs can be checked automatically in a style similar to that found in Twelf [14]. *Mode* checking establishes a groundedness relationship between inputs and outputs,

reduction checking determines a subterm relationship between inputs and outputs, and *termination* checking ensures queries complete in finite time. Mode checking is available in the current implementation of LolliMon, but there is no formal description nor proof of correctness of mode, reduction, or termination analysis for LolliMon. This work gives a refined operational semantics for LolliMon and formal algorithms for mode, reduction, and termination analysis. We state correctness criteria for the analyses, but defer formal proofs of correctness to future work. Besides establishing the correctness of logic programs, these static analyses could be extended to an implementation of CLF in order to verify properties of proofs encoded in the framework.

Two main methods for fully automatic termination checking for higher-order term rewriting systems are van de Pol and Schwichtenberg's strict functionals [20] and Jouannaud and Rubio's recursive path orderings [9]. However, the presence of higher-order subgoals in the LF family of languages prevents direct application of these techniques. Rohwedder and Pfenning [19] develop mode and termination analyses for higher-order logic programs in Elf [13] whereby the properties are manually specified and automatically verified. The mode behavior of programs is given by specifying which parameters are inputs and outputs, and termination is specified by indicating which input terms must decrease in recursive subgoals. The correctness of these analyses is stated with respect to the operational semantics with an explicit success continuation and proven by induction on the computation sequence. Pientka and Pfenning [17, 15, 16] present sequent calculi for deriving subterm relationships in Twelf, reduction checking for proving outputs are subterms of inputs, and termination checking that permits course-of-value induction with the aid of reduction properties. They prove the consistency of the subterm sequent calculi by cut-admissibility, but do not prove the correctness of reduction or termination checking with respect to the operational semantics. Termination checking for higher-order logic programs in Twelf is applicable to the asynchronous fragment of LolliMon but is complicated by the richer formula (and proof term) language and the interaction with forward-chaining.

1.1 LolliMon

The Concurrent Logical Framework (CLF) [21] is a foundational dependent type theory based on linear logic that internalizes the ability to represent state and concurrency. The formula language syntactically distinguishes asynchronous (A) and synchronous (S) formulas, in the terminol-

$$
\begin{array}{rll}
A & ::= P \mid \{S\} \mid \top \mid A_1 \mathbin{\&} A_2 & \text{Asynchronous Formulas} \\
& \quad \mid S_2 \multimap A_1 \mid \forall x{:}A_2.A_1 & \\
S & ::= Q \mid ¡Q \mid 1 \mid S_1 \otimes S_2 & \text{Synchronous Formulas} \\
& \quad \mid \exists x{:}A_2.S_1 \mid A \mid !A & \\
P, Q & ::= a \cdot Z & \text{Atomic Formulas} \\
N & ::= H \cdot Z \mid \lambda x.N & \text{Terms} \\
H & ::= c \mid x & \text{Heads} \\
Z & ::= \text{NIL} \mid N; Z & \text{Spines} \\
\Gamma & ::= \cdot \mid \Gamma, A & \text{Unrestricted context} \\
\Delta & ::= \cdot \mid \Delta, A & \text{Linear context} \\
\Psi & ::= \cdot \mid S, \Psi & \text{Pattern context}
\end{array}
$$

Figure 1: LolliMon Syntax

ogy of Andreoli [1], by segregating the latter in a monad. Linearity internalizes resource consumption and can be used to encode state, while synchronous fragment can be used to encode concurrency. LolliMon [10] is a linear logic programming language that extends a fragment of CLF with an operational proof search semantics. The monad mediates between backward-chaining, backtracking semantics for the asynchronous connectives and forward-chaining, committed choice for the synchronous connectives. Top-down search in the asynchronous fragment permits the representation of backtracking logic programs, while bottom-up search in the synchronous fragment permits the representation of saturating algorithms.

LolliMon differs from CLF in the treatment of the dependent function type and the absence of proof terms. The dependent type $\Pi x{:}A_2.A_1$ of CLF is differentiated as intuitionistic implication $A_2 \supset A_1$ and universal quantification $\forall x{:}A_2.A_1$ in LolliMon, where objects are restricted to a prenex polymorphically typed λ-calculus. An operational semantics of CLF could be given by extending LolliMon to treat the dependent function type and allowing quantification over CLF proof terms.

In this work we consider a variant of LolliMon that generalizes the formula language and makes the operational semantics more explicit. The syntax is given in Figure 1, the judgments in Figure 2, and the operational semantics in Figures 3 and 4. We refer the reader to [10] for a dis-

$\Gamma; \Delta \Rightarrow A$	Right inversion
$\Gamma; \Delta; A \ll P$	Left focusing
$\Gamma; \Delta \rightarrow S$	Forward chaining
$\Gamma; \Delta; A < S$	Monadic left focusing
$\Gamma; \Delta; \Gamma'; \Delta'; \Psi \Leftarrow S$	Left inversion
$\Gamma; \Delta \gg S$	Right focusing
$\Gamma; \Delta; \Psi \Leftarrow_{\mathrm{A}} A$	Left inversion (Asynchronous)

Figure 2: LolliMon Judgments

cussion on the operational semantics of proof search for LolliMon. Here we make saturation and quiescence explicit in by initiating left inversion $\Gamma; \Delta; \Gamma'; \Delta'; \Psi \Leftarrow S$ (rule $\{\}\mathbf{L}$) with separate contexts Γ', Δ' to accumulate the new formulas and only succeeding if new unrestricted or linear hypotheses have been created (rule $\rightarrow\Leftarrow$). An explicit success continuation semantics is obtained by having a stack of goals

$$\mathcal{S} ::= \cdot \mid J/\mathcal{S}$$

where J is one of the judgments, transforming each rule

$$\frac{J_1 \quad \ldots \quad J_n}{J} \qquad \rightsquigarrow \qquad \frac{J_1/\ldots/J_n/\mathcal{S}}{J/\mathcal{S}}$$

and adding a rule that states the empty stack succeeds.

The term language was a prenex polymorphically typed λ-calculus and quantification was restricted to terms of type τ in that type system. In this work restrict terms N to a simply-typed λ-calculus but reuse the formula language as the type system, thus allowing quantification over terms of type A restricted to intuitionistic implication and atomic types.

The original formulation of LolliMon included linear implication $A_2 \multimap A_1$ and intuitionistic implication $A_2 \supset A_1$ where the antecedent is asynchronous, here we generalize the language by allowing a synchronous antecedent in linear implication $S_2 \multimap A_1$ and removing intuitionistic implication because it can be encoded as $!A_2 \multimap A_1$. This means that when left focusing on a linear implication $S_2 \multimap A_1$ (rule $\multimap\mathbf{L}$), we continue left focusing on A_1 and additionally right focus on S_2; and when right inverting

$$\frac{}{\Gamma;\cdot;P \ll P}\ P\mathbf{L} \qquad \frac{\Gamma;\Delta;A \ll P}{\Gamma;\Delta,A \Rightarrow P}\ P\mathbf{R}_x \qquad \frac{\Gamma,A;\Delta;A \ll P}{\Gamma,A;\Delta \Rightarrow P}\ P\mathbf{R}_x \qquad \frac{\Gamma;\Delta \to S}{\Gamma;\Delta \Rightarrow \{S\}}\ \{\}\mathbf{R}$$

(no $\top\mathbf{L}$ rule) $\qquad\qquad \dfrac{}{\Gamma;\Delta \Rightarrow \top}\ \top\mathbf{R}$

$$\frac{\Gamma;\Delta;A_1 \ll P}{\Gamma;\Delta;A_1 \,\&\, A_2 \ll P}\ \&\mathbf{L}_1 \qquad \frac{\Gamma;\Delta;A_2 \ll P}{\Gamma;\Delta;A_1 \,\&\, A_2 \ll P}\ \&\mathbf{L}_2$$

$$\frac{\Gamma;\Delta \Rightarrow A_1 \qquad \Gamma;\Delta \Rightarrow A_2}{\Gamma;\Delta \Rightarrow A_1 \,\&\, A_2}\ \&\mathbf{R}$$

$$\frac{\Gamma;\Delta_1;A_1 \ll P \qquad \Gamma;\Delta_2 \gg S_2}{\Gamma;\Delta_1,\Delta_2;S_2 \multimap A_1 \ll P}\ \multimap\mathbf{L} \qquad \frac{\Gamma;\Delta;S_2 \Leftarrow_{\mathrm{A}} A_1}{\Gamma;\Delta \Rightarrow S_2 \multimap A_1}\ \multimap\mathbf{R}$$

$$\frac{\Gamma;\Delta;[t/x]A_1 \ll P}{\Gamma;\Delta;\forall x{:}A_2.A_1 \ll P}\ \forall\mathbf{L} \qquad \frac{\Gamma;\Delta \Rightarrow [a/x]A_1}{\Gamma;\Delta \Rightarrow \forall x{:}A_2.A_1}\ \forall\mathbf{R}$$

$$\frac{\Gamma;\Delta \Rightarrow A}{\Gamma;\Delta;\cdot \Leftarrow_{\mathrm{A}} A}\ \Rightarrow\!\Leftarrow_{\mathrm{A}} \qquad \frac{\Gamma;\Delta,Q;\Psi \Leftarrow_{\mathrm{A}} S}{\Gamma;\Delta;Q,\Psi \Leftarrow_{\mathrm{A}} S}\ Q\mathbf{L} \qquad \frac{\Gamma,Q;\Delta;\Psi \Leftarrow_{\mathrm{A}} S}{\Gamma;\Delta;¡Q,\Psi \Leftarrow_{\mathrm{A}} S}\ ¡\mathbf{L}_{\mathrm{A}}$$

$$\frac{\Gamma;\Delta;\Psi \Leftarrow_{\mathrm{A}} A}{\Gamma;\Delta;1,\Psi \Leftarrow_{\mathrm{A}} A}\ 1\mathbf{L}_{\mathrm{A}} \qquad \frac{\Gamma;\Delta;S_1,S_2,\Psi \Leftarrow_{\mathrm{A}} A}{\Gamma;\Delta;S_1 \otimes S_2,\Psi \Leftarrow_{\mathrm{A}} A}\ \otimes\mathbf{L}_{\mathrm{A}}$$

$$\frac{\Gamma;\Delta;[a/x]S_1,\Psi \Leftarrow_{\mathrm{A}} A}{\Gamma;\Delta;\exists x{:}A_2.S_1,\Psi \Leftarrow_{\mathrm{A}} A}\ \exists\mathbf{L}_{\mathrm{A}} \qquad \frac{\Gamma;\Delta,A';\Psi \Leftarrow_{\mathrm{A}} A}{\Gamma;\Delta;A',\Psi \Leftarrow_{\mathrm{A}} A}\ A\mathbf{L}_{\mathrm{A}}$$

$$\frac{\Gamma,A';\Delta;\Psi \Leftarrow_{\mathrm{A}} A}{\Gamma;\Delta;!A',\Psi \Leftarrow_{\mathrm{A}} A}\ !\mathbf{L}_{\mathrm{A}}$$

Figure 3: LolliMon Operational Semantics

$$\frac{\Gamma;\Delta \gg S}{\Gamma;\Delta \to S}\gg\!\to \qquad \frac{\Gamma;\Delta;A < S}{\Gamma;\Delta,A \to S}\,x' \qquad \frac{\Gamma,A;\Delta;A < S}{\Gamma,A;\Delta \to S}\,u'$$

(no $P\mathbf{L}'$ rule) $\qquad \dfrac{\Gamma;\Delta;\cdot;\cdot;S' \Leftarrow S}{\Gamma;\Delta;\{S'\} < S}\,\{\}\mathbf{L}$

(no $\top\mathbf{L}'$ rule) $\qquad \dfrac{\Gamma;\Delta;A_1 < S}{\Gamma;\Delta;A_1 \,\&\, A_2 < S}\,\&\mathbf{L}'_1 \qquad \dfrac{\Gamma;\Delta;A_2 < S}{\Gamma;\Delta;A_1 \,\&\, A_2 < S}\,\&\mathbf{L}'_2$

$$\frac{\Gamma;\Delta_2 \gg S_2 \qquad \Gamma;\Delta_1;A_1 < S}{\Gamma;\Delta_1,\Delta_2;S_2 \multimap A_1 < S}\,\multimap\mathbf{L}' \qquad \frac{\Gamma;\Delta;[t/x]A_1 < S}{\Gamma;\Delta;\forall x{:}A_2.A_1 < S}\,\forall\mathbf{L}'$$

$$\frac{\Gamma \not\subseteq \Gamma' \text{ or } \Delta' \neq \cdot \qquad \Gamma,\Gamma';\Delta,\Delta' \to S}{\Gamma;\Delta;\Gamma';\Delta';\cdot \Leftarrow S}\,\to\!\Leftarrow$$

$$\frac{\Gamma;\Delta;\Gamma';\Delta',Q;\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';Q,\Psi \Leftarrow S}\,Q\mathbf{L} \qquad \frac{}{\Gamma;Q \gg Q}\,Q\mathbf{R}_x \qquad \frac{}{\Gamma,Q;\cdot \gg Q}\,Q\mathbf{R}_u$$

$$\frac{\Gamma;\Delta;\Gamma',Q;\Delta';\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';¡Q,\Psi \Leftarrow S}\,¡\mathbf{L} \qquad \frac{}{\Gamma,Q;\cdot \gg ¡Q}\,¡\mathbf{R}$$

$$\frac{\Gamma;\Delta;\Gamma';\Delta';\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';1,\Psi \Leftarrow S}\,1\mathbf{L} \qquad \frac{}{\Gamma;\cdot \gg 1}\,1\mathbf{R}$$

$$\frac{\Gamma;\Delta;\Gamma';\Delta';S_1,S_2,\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';S_1 \otimes S_2,\Psi \Leftarrow S}\,\otimes\mathbf{L} \qquad \frac{\Gamma;\Delta_1 \gg S_1 \qquad \Gamma;\Delta_2 \gg S_2}{\Gamma;\Delta_1,\Delta_2 \gg S_1 \otimes S_2}\,\otimes\mathbf{R}$$

$$\frac{\Gamma;\Delta;\Gamma';\Delta';[a/x]S_1,\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';\exists x{:}A_2.S_1,\Psi \Leftarrow S}\,\exists\mathbf{L} \qquad \frac{\Gamma;\Delta \gg [t/x]S_1}{\Gamma;\Delta \gg \exists x{:}A_2.S_1}\,\exists\mathbf{R}$$

$$\frac{\Gamma;\Delta;\Gamma';\Delta',A;\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';A,\Psi \Leftarrow S}\,A\mathbf{L} \qquad \frac{\Gamma;\Delta \Rightarrow A}{\Gamma;\Delta \gg A}\,\Rightarrow\!\gg$$

$$\frac{\Gamma;\Delta;\Gamma',A;\Delta';\Psi \Leftarrow S}{\Gamma;\Delta;\Gamma';\Delta';!A,\Psi \Leftarrow S}\,!\mathbf{L} \qquad \frac{\Gamma;\cdot \Rightarrow A}{\Gamma;\cdot \gg !A}\,!\mathbf{R}$$

Figure 4: LolliMon Operational Semantics (continued)

a linear implication $S_2 \multimap A_1$ (rule $\multimap\mathbf{R}$), we immediately decompose the antecedent by left inverting on S_2 ($\mathbf{L}_\mathrm{A}$ rules) and then resume by right inverting on A_1 (rule $\Rightarrow\Leftarrow_\mathrm{A}$).

We also generalize asynchronous atoms by syntactically allowing them to be either asynchronous P or synchronous Q. Previously, when monadic left focusing on an implication $A_2 \multimap A_1$ during forward-chaining, the subgoal A_2 would initiate a backward-chaining phase. In particular, since all atomic formulas P were considered asynchronous, an atomic subgoal would initiate a backward-chaining phase by selecting and left focusing on a clause until the head matched the atom and the subgoals were satisfied. It was possible to simulate synchronous atoms that did not require switching to backward-chaining, by syntactically restricting their occurrence in a clause head to be monadic so that an atomic goal would initiate backward-chaining but could only succeed if the atom was already in the context. Here we allow explicit synchronous atoms Q which can only be generated if they occur in a monadic head (rule $Q\mathbf{L}$) and can only be proven as goals during right focusing if they appear in the context (rules $Q\mathbf{R}_x$ and $Q\mathbf{R}_u$). However, due to linearity and the syntax of the unrestricted modality, it would not be possible to generate unrestricted synchronous atoms. Therefore we also include unrestricted synchronous atoms ¡Q which are added to the unrestricted context when they are generated by a forward-chaining clause (rule ¡$\mathbf{L}$) and proven as goals during right focusing (rule ¡$\mathbf{R}$).

2 Project Results

This work presents mode, reduction, and termination analyses for LolliMon as syntax-directed inference rules that verify suitable moding and subterm relationships between terms in a clause's head and its subgoals. These analyses combine existing techniques for higher-order backward-chaining intuitionistic logic programs and complexity analysis of first-order forward-chaining logic programs. The novelty of this work stems from LolliMon's richer formula language, the semantics of linearity, and the mixed backward-chaining and forward-chaining search behavior. We assume mode, reduction, and termination are explicitly specified by mode, red, and term for each atom.

Since the syntax of LolliMon allows clauses such as $S \multimap A_1 \;\&\; A_2$ to have multiple heads, it is possible for a clause to be used in both backward- and forward-chaining. However, focusing on such a clause will only use one of the heads so it is possible to transform a clause with multiple heads

into multiple clauses with one head each. In this work we allow clauses with multiple heads, but restrict the analyses to backward-chaining clauses which only have (asynchronous) atomic or $\top$ heads and forward-chaining clauses which only have monadic heads.

Backward- and forward-chaining clauses satisfy a specified mode or reduction property if the property of the head can be deduced from the properties of the subgoals. Due to the combination of backward- and forward-chaining in the operational semantics, it is necessary to perform a global analysis for termination checking in order to extract a dependence relation between synchronous and asynchronous atoms prior to checking each clause individually. Backward-chaining clauses satisfy termination if the recursive subgoals involve proper subterms of the terms in the head and the nonrecursive subgoals are known to be terminating. These criteria can be extended to mutually recursive atomic predicates [18], although incorporating this analysis is deferred to future work. The proof system for reduction and termination checking of backward-chaining higher-order logic programs in [16] serves as a basis for the analyses in the asynchronous phase. Termination of forward-chaining clauses that only generate unrestricted hypotheses can be verified by determining that the terms in the head are subterms (not necessarily proper) of the terms in the antecedents. Termination of forward-chaining clauses that generate linear hypotheses is subtle and in this work we discuss design considerations for handling the linear case, although the formal analysis does not account for this case.

2.1 Mode Checking

A mode specification μ for an atom a consists of an input $+$ or output $-$ assignment to each of its arguments. Intuitively, an asynchronous atom a_P with mode specification $\mathsf{mode}(a_P) = \mu$ should satisfy the property that whenever a goal $a_P \cdot Z$ is solved with ground input terms and the goal succeeds, then the output terms are also ground. A backward-chaining clause is well-moded if for each head it satisfies the mode specification, namely that if the input terms of a head are ground and the subgoals satisfy their mode property, then the output terms of that head are also ground. Since synchronous atoms are only generated by forward-chaining clauses, in order to maintain a ground context synchronous atoms have assign the output $-$ mode to each argument.

Mode checking for a backward-chaining clause is performed by extracting from the head a proof obligation o of which terms must be ground if the clause returns and an initial abstract substitution $\hat{\sigma}$ of which variables

$\mu ::= \cdot \mid +, \mu \mid -, \mu$	Spine moding
$\hat{\sigma} ::= \cdot \mid \hat{\sigma}, \mathsf{g}/x \mid \hat{\sigma}, \mathsf{u}/x$	Abstract substitution
$o ::= \cdot \mid o, N$	Proof obligation
$\iota ::= \cdot \mid \iota, (\hat{\sigma}; o)$	Substitution and proof obligation pair

Figure 5: Mode Checking Syntax

are assumed to be ground, then traversing the clause from the head outwards refining the abstract substitution assuming the subgoals are well-moded, and checking that the final abstract substitution ensures the terms in the proof obligation are ground. The abstract substitution for backward-chaining clauses is refined by proceeding from the innermost to the outermost subgoal because that is the order in which subgoals are solved according to the operational semantics. Similarly, since forward-chaining clauses are used by solving subgoals from the outside in, the abstract substitution is refined from the outermost to the innermost goal and finally the head is checked to satisfy its mode specification. For synchronous atoms occurring as goals they can be assumed to be ground and occurring as heads they must be shown to be ground. Since there is one proof obligation per head, it is necessary to maintain a list ι of proof obligations and their associated abstract substitutions. Due to presence of higher-order subgoals, it is necessary to also check the well-moding of subclauses that are introduced to the context when solving a subgoal. The syntax of specifications, abstract substitutions, and proof obligations is given in Figure 5.

The auxiliary judgments are given in Figure 6 and the principal judgments are given in Figures 7, 8, and 9. We assume a judgment $\hat{\sigma} \vdash N$ ground that check that a term N is ground under the abstract substitution $\hat{\sigma}$, and lift it to similar judgments for spines and proof obligations. We also assume a judgment $\hat{\sigma} \vdash N/\hat{\sigma}'$ that refines $\hat{\sigma}$ into $\hat{\sigma}'$ assuming N is ground, and lift it to a judgment for spines. The judgment $\vdash \iota$ checks that each abstract substitution satisfies its associated proof obligation. When an atom is encountered at the head of a clause, the proof obligation and initial abstract substitution are extracted from the spine with the judgment $\hat{\sigma}; o \stackrel{\text{MCZ}}{\Longrightarrow} \mu \sim Z \mid \hat{\sigma}'; o'$ by refining the initial substitution $\hat{\sigma}$ into $\hat{\sigma}'$ using terms that are ground on input and extending the initial proof obligations o with terms that should be ground on output. Dually, when an atom is encountered as a subgoal,

the judgment $\hat{\sigma} \overset{\text{MGZ}}{\Longrightarrow} \mu \sim Z \mid \hat{\sigma}'$ checks that the inputs are ground under the abstract substitution $\hat{\sigma}$ and refines it into $\hat{\sigma}'$ assuming the outputs are ground.

The principal judgment $\hat{\sigma} \vdash S\ \mathsf{mode}$ checks that a clause S is well-moded by checking that synchronous atoms are ground and that asynchronous clauses are well-moded according to whether they are backward- or forward-chaining. Backward-chaining clauses are checked by extracting a list of proof obligations and abstract substitution with the judgment $\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A \mid \iota$ and ensuring the proof obligation is satisfied. Subgoals of backward-chaining clauses are used to refine the abstract substitution and check subclauses with the judgments $\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A \mid \hat{\sigma}'$ and $\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} S \mid \hat{\sigma}'$, and a lifted version $\iota \overset{\text{MBGS}}{\Longrightarrow} S \mid \iota'$ for lists that keeps the proof obligation but refines the substitution. Forward-chaining clauses are checked with the judgment $\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A$ by refining the abstract substitution with the judgments the judgments $\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A \mid \hat{\sigma}'$ and $\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} S \mid \hat{\sigma}'$ and checking that the synchronous head is well-moded.

2.2 Reduction Checking

A reduction property is a specification of a subterm relationship between the arguments of an atom, which is useful for course-of-values induction. We assume each atom is assigned a reduction property $\mathsf{red}(a \cdot (x_1; \ldots; x_n)) = \mathcal{P}$ consisting of an order O, a lexicographic $\{O_1, O_2\}$, or simultaneous order $[O_1, O_2]$ between arguments $x_1, \ldots, x_n$. Intuitively, an atom satisfies a reduction property $\mathcal{P}$ if whenever a goal $a_P \cdot Z$ is solved and succeeds, the order relation $\mathcal{P}$ holds of the specified arguments in Z. Backward- and forward-chaining clauses satisfy a reduction property of a head if the reduction properties of the subgoals imply the reduction property of that head. Here we represent parameters as $\forall x{:}A$, variables as $\exists x{:}A$, and subgoals $\exists!u{:}A$ with a mixed prefix context Φ. Figure 10 gives the syntax for orders, order relations, order contexts, and mixed prefix contexts based on the Twelf case.

We assume a sequent calculus with judgment $\Phi; \Omega \longrightarrow \mathcal{P}$, which establishes the subterm relation $\mathcal{P}$ in the context of subterm relations Ω. Such a calculus can be adapted from [16] to the proof term language of LolliMon (and of CLF) and its consistency can be shown by proving cut-admissibility. The judgments $\Phi \overset{\text{RPA}}{\Longrightarrow} A/\Omega$ and $\Phi \overset{\text{RPS}}{\Longrightarrow} S/\Omega$ in Figure 11 extract the reduction property of every atom occurring in a backward- or forward-chaining

$\boxed{\vdash \iota}$ [Obligations satisfied]

$$\frac{}{\vdash \cdot} \qquad \frac{\vdash \iota \qquad \hat{o} \vdash o \ \mathsf{ground}}{\vdash \iota, (\hat{o}; o)}$$

$\boxed{\hat{o}; o \overset{\text{MCZ}}{\Longrightarrow} \mu \sim Z \mid \hat{o}'; o'}$ [Spine moding (clause)]

$$\frac{}{\hat{o}; o \overset{\text{MCZ}}{\Longrightarrow} \cdot \sim \text{NIL} \mid \hat{o}; o} \qquad \frac{\hat{o} \vdash N/\hat{o}' \qquad \hat{o}'; o \overset{\text{MCZ}}{\Longrightarrow} \mu \sim Z \mid \hat{o}''; o''}{\hat{o}; o \overset{\text{MCZ}}{\Longrightarrow} +, \mu \sim N; Z \mid \hat{o}''; o''}$$

$$\frac{\hat{o}; o, N \overset{\text{MCZ}}{\Longrightarrow} \mu \sim Z \mid \hat{o}'; o'}{\hat{o}; o \overset{\text{MCZ}}{\Longrightarrow} -, \mu \sim N; Z \mid \hat{o}'; o'}$$

$\boxed{\hat{o} \overset{\text{MGZ}}{\Longrightarrow} \mu \sim Z \mid \hat{o}'}$ [Spine moding (goal)]

$$\frac{}{\hat{o} \overset{\text{MGZ}}{\Longrightarrow} \cdot \sim \text{NIL} \mid \hat{o}} \qquad \frac{\hat{o} \vdash N \ \mathsf{ground} \qquad \hat{o} \overset{\text{MGZ}}{\Longrightarrow} \mu \sim Z \mid \hat{o}'}{\hat{o} \overset{\text{MGZ}}{\Longrightarrow} +, \mu \sim N; Z \mid \hat{o}'}$$

$$\frac{\hat{o}' \overset{\text{MGZ}}{\Longrightarrow} \mu \sim Z \mid \hat{o}' \qquad \hat{o}' \vdash N/\hat{o}''}{\hat{o} \overset{\text{MGZ}}{\Longrightarrow} -, \mu \sim N; Z \mid \hat{o}''}$$

Figure 6: Mode Checking Auxiliary Judgments

$\boxed{\hat{\sigma} \vdash S \text{ mode}}$ [Well moded (clause)]

$$\frac{\hat{\sigma} \vdash Z \text{ ground}}{\hat{\sigma} \vdash a \cdot Z \text{ mode}}\ Q\text{MODE} \qquad \frac{\hat{\sigma} \vdash Z \text{ ground}}{\hat{\sigma} \vdash ¡a \cdot Z \text{ mode}}\ ¡\text{MODE} \qquad \frac{}{\hat{\sigma} \vdash 1 \text{ mode}}\ 1\text{MODE}$$

$$\frac{\hat{\sigma} \vdash S_1 \text{ mode} \qquad \hat{\sigma} \vdash S_2 \text{ mode}}{\hat{\sigma} \vdash S_1 \otimes S_2 \text{ mode}}\ \otimes\text{MODE} \qquad \frac{\hat{\sigma}, \mathsf{g}/x \vdash S_1 \text{ mode}}{\hat{\sigma} \vdash \exists x{:}A_2.S_1 \text{ mode}}\ \exists\text{MODE}$$

$$\frac{(A \text{ backward}) \qquad \hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A \mid \iota \qquad \vdash \iota}{\hat{\sigma} \vdash A \text{ mode}}\ A\text{MODE}_b$$

$$\frac{(A \text{ forward}) \qquad \hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A}{\hat{\sigma} \vdash A \text{ mode}}\ A\text{MODE}_f$$

$$\frac{(A \text{ backward}) \qquad \hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A \mid \iota \qquad \vdash \iota}{\hat{\sigma} \vdash !A \text{ mode}}\ !\text{MODE}_b$$

$$\frac{(A \text{ forward}) \qquad \hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A}{\hat{\sigma} \vdash !A \text{ mode}}\ !\text{MODE}_f$$

Figure 7: Mode Checking Principal Judgments (1)

$\boxed{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A \mid \iota}$ [Mode check (backward, clause)]

$$\frac{\hat{\sigma}; \cdot \overset{\text{MCZ}}{\Longrightarrow} \mathsf{mode}(a) \sim Z \mid \hat{\sigma}'; o'}{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} a \cdot Z \mid (\hat{\sigma}'; o')}\ P\text{MBCA} \qquad \text{(no } \{\}\text{MBCA rule)}$$

$$\frac{}{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} \top \mid \cdot}\ \top\text{MBCA} \qquad \frac{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A_1 \mid \iota_1 \qquad \hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A_2 \mid \iota_2}{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A_1 \,\&\, A_2 \mid \iota_1, \iota_2}\ \&\text{MBCA}$$

$$\frac{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} A_1 \mid \iota \qquad \iota \overset{\text{MBGS}}{\Longrightarrow} S_2 \mid \iota'}{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} S_2 \multimap A_1 \mid \iota'}\ \multimap\text{MBCA} \qquad \frac{\hat{\sigma}, \mathtt{u}/x \overset{\text{MBCA}}{\Longrightarrow} A_1 \mid \iota}{\hat{\sigma} \overset{\text{MBCA}}{\Longrightarrow} \forall x{:}A_2.A_1 \mid \iota}\ \forall\text{MBCA}$$

$\boxed{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A \mid \hat{\sigma}' \text{ and } \hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} S \mid \hat{\sigma}'}$ [Mode check (backward, goal)]

$$\frac{\hat{\sigma} \overset{\text{MGZ}}{\Longrightarrow} \mathsf{mode}(a) \sim Z \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} a \cdot Z \mid \hat{\sigma}'}\ P\text{MBGA} \qquad \frac{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} S \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} \{S\} \mid \hat{\sigma}'}\ \{\}\text{MBGA}$$

$$\frac{}{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} \top \mid \hat{\sigma}}\ \top\text{MBGA} \qquad \frac{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A_1 \mid \hat{\sigma}' \qquad \hat{\sigma}' \overset{\text{MBGA}}{\Longrightarrow} A_2 \mid \hat{\sigma}''}{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A_1 \,\&\, A_2 \mid \hat{\sigma}''}\ \&\text{MBGA}$$

$$\frac{\hat{\sigma} \vdash S_2\ \mathsf{mode} \qquad \hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A_1 \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} S_2 \multimap A_1 \mid \hat{\sigma}'}\ \multimap\text{MBGA} \qquad \frac{\hat{\sigma}, \mathtt{g}/x \overset{\text{MBGA}}{\Longrightarrow} A_1 \mid \hat{\sigma}', \_/x}{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} \forall x{:}A_2.A_1 \mid \hat{\sigma}'}\ \forall\text{MBGA}$$

$$\frac{\hat{\sigma} \overset{\text{MGZ}}{\Longrightarrow} \mathsf{mode}(a) \sim Z \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} a \cdot Z \mid \hat{\sigma}'}\ Q\text{MBGS} \qquad \frac{\hat{\sigma} \overset{\text{MGZ}}{\Longrightarrow} \mathsf{mode}(a) \sim Z \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} ¡a \cdot Z \mid \hat{\sigma}'}\ ¡\text{MBGS}$$

$$\frac{}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} 1 \mid \hat{\sigma}}\ 1\text{MBGS} \qquad \frac{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} S_1 \mid \hat{\sigma}' \qquad \hat{\sigma}' \overset{\text{MBGS}}{\Longrightarrow} S_2 \mid \hat{\sigma}''}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} S_1 \otimes S_2 \mid \hat{\sigma}''}\ \otimes\text{MBGS}$$

$$\frac{\hat{\sigma}, \mathtt{u}/x \overset{\text{MBGS}}{\Longrightarrow} S_1 \mid \hat{\sigma}', \_/x}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} \exists x{:}A_2.S_1 \mid \hat{\sigma}'}\ \exists\text{MBGS} \qquad \frac{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} A \mid \hat{\sigma}'}\ A\text{MBGS}$$

$$\frac{\hat{\sigma} \overset{\text{MBGA}}{\Longrightarrow} A \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MBGS}}{\Longrightarrow} !A \mid \hat{\sigma}'}\ !\text{MBGS}$$

Figure 8: Mode Checking Principal Judgments (2)

$\boxed{\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A}$ [Mode check (forward, clause)]

$$\frac{\hat{\sigma} \vdash S \;\mathsf{mode}}{\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} \{S\}}\;\{\}\text{MFCA}$$

(no PMFCA rule) (no $\top$MFCA rule)

$$\frac{\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A_1 \qquad \hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A_2}{\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} A_1 \,\&\, A_2}\;\&\text{MFCA} \qquad \frac{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} S_2 \mid \hat{\sigma}' \qquad \hat{\sigma}' \overset{\text{MFCA}}{\Longrightarrow} A_1}{\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} S_2 \multimap A_1}\;\multimap\text{MFCA}$$

$$\frac{\hat{\sigma}, \mathtt{u}/x \overset{\text{MFCA}}{\Longrightarrow} A_1}{\hat{\sigma} \overset{\text{MFCA}}{\Longrightarrow} \forall x{:}A_2.A_1}\;\forall\text{MFCA}$$

$\boxed{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A \mid \hat{\sigma}' \text{ and } \hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} S \mid \hat{\sigma}'}$ [Mode check (forward, goal)]

$$\frac{\hat{\sigma} \overset{\text{MGZ}}{\Longrightarrow} \mathsf{mode}(a) \sim Z \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} a \cdot Z \mid \hat{\sigma}'}\;P\text{MFGA} \qquad \frac{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} S \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} \{S\} \mid \hat{\sigma}'}\;\{\}\text{MFGA}$$

$$\frac{}{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} \top \mid \hat{\sigma}}\;\top\text{MFGA} \qquad \frac{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A_1 \mid \hat{\sigma}' \qquad \hat{\sigma}' \overset{\text{MFGA}}{\Longrightarrow} A_2 \mid \hat{\sigma}''}{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A_1 \,\&\, A_2 \mid \hat{\sigma}''}\;\&\text{MFGA}$$

$$\frac{\hat{\sigma} \vdash S_2 \;\mathsf{mode} \qquad \hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A_1 \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} S_2 \multimap A_1 \mid \hat{\sigma}'}\;\multimap\text{MFGA} \qquad \frac{\hat{\sigma}, \mathtt{g}/x \overset{\text{MFGA}}{\Longrightarrow} A_1 \mid \hat{\sigma}, \_/x}{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} \forall x{:}A_2.A_1 \mid \hat{\sigma}}\;\forall\text{MFGA}$$

$$\frac{\hat{\sigma} \vdash Z/\hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} a \cdot Z \mid \hat{\sigma}'}\;Q\text{MFGS} \qquad \frac{\hat{\sigma} \vdash Z/\hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} ¡a \cdot Z \mid \hat{\sigma}'}\;¡\text{MFGS} \qquad \frac{}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} 1 \mid \hat{\sigma}}\;1\text{MFGS}$$

$$\frac{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} S_1 \mid \hat{\sigma}' \qquad \hat{\sigma}' \overset{\text{MFGS}}{\Longrightarrow} S_2 \mid \hat{\sigma}''}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} S_1 \otimes S_2 \mid \hat{\sigma}''}\;\otimes\text{MFGS} \qquad \frac{\hat{\sigma}, \mathtt{u}/x \overset{\text{MFGS}}{\Longrightarrow} S_1 \mid \hat{\sigma}', \_/x}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} \exists x{:}S_2.S_1 \mid \hat{\sigma}'}\;\exists\text{MFGS}$$

$$\frac{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} A \mid \hat{\sigma}'}\;A\text{MFGS} \qquad \frac{\hat{\sigma} \overset{\text{MFGA}}{\Longrightarrow} A \mid \hat{\sigma}'}{\hat{\sigma} \overset{\text{MFGS}}{\Longrightarrow} !A \mid \hat{\sigma}'}\;!\text{MFGS}$$

Figure 9: Mode Checking Principal Judgments (3)

$$
\begin{array}{lr}
\Omega ::= \cdot \mid \Omega, \mathcal{P} & \text{Order context} \\
\mathcal{P} ::= \mathcal{O}_1 \prec \mathcal{O}_2 \mid \mathcal{O}_1 \preceq \mathcal{O}_2 \mid \mathcal{O}_1 \equiv \mathcal{O}_2 \mid \Pi x{:}A.\mathcal{P} & \text{Order relations} \\
\mathcal{O} ::= N \mid \{\mathcal{O}_1, \mathcal{O}_2\} \mid [\mathcal{O}_1, \mathcal{O}_2] & \text{Orders} \\
\Phi ::= \cdot \mid \Phi, \forall x{:}A \mid \Phi, \exists x{:}A \mid \Phi, \exists! u{:}A & \text{Mixed prefix context}
\end{array}
$$

Figure 10: Reduction Checking Syntax

head and accumulate it into Ω. The judgments $\Phi; \Omega \overset{\text{RGA}}{\Longrightarrow} A$ and $\Phi; \Omega \overset{\text{RGS}}{\Longrightarrow} S$ in Figure 12 traverse a subgoal A or S and verify that subclauses satisfy their corresponding reduction properties. Figures 13, 14, and 15 present the principal judgment $\Phi; \Omega \vdash S\ \mathsf{red}$ that checks the reduction property of backward and forward-chaining clauses via the judgments $\Phi; \Omega \overset{\text{RBCA}}{\Longrightarrow} A$ and $\Phi; \Omega \overset{\text{RFCA}}{\Longrightarrow} A$ which ensure that the reduction properties of the subgoals imply the reduction of the head and that the reduction property holds of each subclause assuming the reduction property of earlier (closer to the head for backward-chaining clauses) or later (further from the head for forward-chaining clauses) subgoals via the judgment $\Phi; \Omega \overset{\text{RBIA}}{\Longrightarrow} S_g; A_c$. In the Twelf case reduction properties are assumed to be valid (true without any hypotheses), so the reduction properties of all subgoals are used to verify the reduction properties of any subclauses. This formulation of reduction checking differs in that only reduction properties that occur earlier (with respect to the proof search order) are used to prove the reduction property of subclauses.

2.3 Termination Checking

The termination property of an asynchronous atom is that left focusing on any clause with that atom in the head fails or succeeds in a finite number of steps. For LolliMon we conservatively establish termination provided subgoals of backward-chaining clauses are well-founded by invoking recursive calls with strictly smaller arguments (in the inductive argument given by $\mathsf{term}(a \cdot Z) = \mathcal{O}$) and forward-chaining clauses only generate unrestricted synchronous atoms with arguments that are no larger than those in the antecedents. We defer addressing forward-chaining clauses that generate linear synchronous atoms or asynchronous formulas to future work. Intuitively, the correctness criteria is that if a program satisfies the termi-

$\boxed{\Phi \overset{\text{RPA}}{\Longrightarrow} A/\Omega \text{ and } \Phi \overset{\text{RPS}}{\Longrightarrow} S/\Omega}$ [Reduction Properties (goal)]

$$\frac{}{\Phi \overset{\text{RPA}}{\Longrightarrow} P/\mathsf{red}(P)}\ P\text{RPA} \qquad \frac{\Phi \overset{\text{RPS}}{\Longrightarrow} S/\Omega}{\Phi \overset{\text{RPA}}{\Longrightarrow} \{S\}/\Omega}\ \{\}\text{RPA} \qquad \frac{}{\Phi \overset{\text{RPA}}{\Longrightarrow} \top/\cdot}\ \top\text{RPA}$$

$$\frac{\Phi \overset{\text{RPA}}{\Longrightarrow} A_1/\Omega_1 \qquad \Phi \overset{\text{RPA}}{\Longrightarrow} A_2/\Omega_2}{\Phi \overset{\text{RPA}}{\Longrightarrow} A_1 \,\&\, A_2/\Omega_1,\Omega_2}\ \&\text{RPA} \qquad \frac{\Phi \overset{\text{RPA}}{\Longrightarrow} A_1/\Omega}{\Phi \overset{\text{RPA}}{\Longrightarrow} S_2 \multimap A_1/\Omega}\ \multimap\text{RPA}$$

$$\frac{\Phi, \forall x{:}A_2 \overset{\text{RPA}}{\Longrightarrow} A_1/\Omega}{\Phi \overset{\text{RPA}}{\Longrightarrow} \forall x{:}A_2.A_1/\Omega}\ \forall\text{RPA}$$

$$\frac{}{\Phi \overset{\text{RPA}}{\Longrightarrow} Q/\mathsf{red}(Q)}\ Q\text{RPS} \qquad \frac{}{\Phi \overset{\text{RPA}}{\Longrightarrow} ¡Q/\mathsf{red}(Q)}\ ¡\text{RPS} \qquad \frac{}{\Phi \overset{\text{RPS}}{\Longrightarrow} 1/\cdot}\ 1\text{RPS}$$

$$\frac{\Phi \overset{\text{RPS}}{\Longrightarrow} S_1/\Omega_1 \qquad \Phi \overset{\text{RPS}}{\Longrightarrow} S_2/\Omega_2}{\Phi \overset{\text{RPS}}{\Longrightarrow} S_1 \otimes S_2/\Omega_1,\Omega_2}\ \otimes\text{RPS} \qquad \frac{\Phi, \exists x{:}A_2 \overset{\text{RPS}}{\Longrightarrow} S_1/\Omega_1}{\Phi \overset{\text{RPS}}{\Longrightarrow} \exists x{:}A_2.S_1/\Omega_1}\ \exists\text{RPS}$$

$$\frac{\Phi \overset{\text{RPA}}{\Longrightarrow} A/\Omega}{\Phi \overset{\text{RPS}}{\Longrightarrow} A/\Omega}\ A\text{RPS} \qquad \frac{\Phi \overset{\text{RPA}}{\Longrightarrow} A/\Omega}{\Phi \overset{\text{RPS}}{\Longrightarrow} !A/\Omega}\ !\text{RPS}$$

Figure 11: Reduction Checking Property Extraction

$\boxed{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A \text{ and } \Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S}$ [Reduces (backward or forward, goal)]

$$\frac{}{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} P}\,P\text{RGA} \qquad \frac{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S}{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} \{S\}}\,\{\}\text{RGA} \qquad \frac{}{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} \top}\,\top\text{RGA}$$

$$\frac{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A_1 \qquad \Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A_2}{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A_1 \,\&\, A_2}\,\&\text{RGA}$$

$$\frac{\Phi;\Omega \vdash S_2 \;\mathsf{red} \qquad \Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A_1}{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} S_2 \multimap A_1}\,\multimap\text{RGA} \qquad \frac{\Phi, \forall x{:}A_2;\Omega \overset{\text{RGA}}{\Longrightarrow} A_1}{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} \forall x{:}A_2.A_1}\,\forall\text{RGA}$$

$$\frac{}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} Q}\,Q\text{RGS} \qquad \frac{}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} ¡Q}\,¡\text{RGS} \qquad \frac{}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} 1}\,1\text{RGS}$$

$$\frac{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S_1 \qquad \Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S_2}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S_1 \otimes S_2}\,\otimes\text{RGS} \qquad \frac{\Phi, \exists x{:}A_2;\Omega \overset{\text{RGS}}{\Longrightarrow} S_1}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} \exists x{:}A_2.S_1}\,\exists\text{RGS}$$

$$\frac{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} A}\,A\text{RGS} \qquad \frac{\Phi;\Omega \overset{\text{RGA}}{\Longrightarrow} A}{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} !A}\,!\text{RGS}$$

Figure 12: Reduction Checking Subgoals

$\boxed{\Phi;\Omega \vdash S\ \mathsf{red}}$ [Reduces (clause)]

$$\frac{\Phi;\Omega \longrightarrow \mathsf{red}(Q)}{\Phi;\Omega \vdash Q\ \mathsf{red}}\ Q\text{RED} \qquad \frac{\Phi;\Omega \longrightarrow \mathsf{red}(Q)}{\Phi;\Omega \vdash ¡Q\ \mathsf{red}}\ ¡\text{RED} \qquad \frac{}{\Phi;\Omega \vdash 1\ \mathsf{red}}\ 1\text{RED}$$

$$\frac{\Phi;\Omega \vdash S_1\ \mathsf{red} \qquad \Phi;\Omega \vdash S_2\ \mathsf{red}}{\Phi;\Omega \vdash S_1 \otimes S_2\ \mathsf{red}}\ \otimes\text{RED} \qquad \frac{\Phi, \exists x{:}A_2;\Omega \vdash S_1\ \mathsf{red}}{\Phi;\Omega \vdash \exists x{:}A_2.S_1\ \mathsf{red}}\ \exists\text{RED}$$

$$\frac{(A\text{ backward}) \qquad \Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} A}{\Phi;\Omega \vdash A\ \mathsf{red}}\ A\text{RED}_b$$

$$\frac{(A\text{ forward}) \qquad \Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} A}{\Phi;\Omega \vdash A\ \mathsf{red}}\ A\text{RED}_f$$

$$\frac{(A\text{ backward}) \qquad \Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} A}{\Phi;\Omega \vdash\ !A\ \mathsf{red}}\ !\text{RED}_b \qquad \frac{(A\text{ forward}) \qquad \Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} A}{\Phi;\Omega \vdash\ !A\ \mathsf{red}}\ !\text{RED}_f$$

Figure 13: Reduction Checking Principal Judgments (1)

$\boxed{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} A}$ [Reduces (backward, clause)]

$$\frac{\Phi;\Omega \longrightarrow \mathsf{red}(P)}{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} P}\, P\text{RBCA} \qquad \text{(no } \{\}\text{RBCA rule)} \qquad \frac{}{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} \top}\, \top\text{RBCA}$$

$$\frac{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} A_1 \qquad \Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} A_2}{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} A_1 \,\&\, A_2}\, \&\text{RBCA}$$

$$\frac{\Phi \overset{\text{RPS}}{\Longrightarrow} S_2/\Omega_2 \qquad \Phi;\Omega,\Omega_2 \overset{\text{RBCA}}{\Longrightarrow} A_1 \qquad \Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S_2;A_1}{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} S_2 \multimap A_1}\, \multimap\text{RBCA}$$

$$\frac{\Phi, \exists x{:}A_2;\Omega \overset{\text{RBCA}}{\Longrightarrow} A_1}{\Phi;\Omega \overset{\text{RBCA}}{\Longrightarrow} \forall x{:}A_2.A_1}\, \forall\text{RBCA}$$

$\boxed{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S_g;A_c}$ [Reduces implication (backward, goal)]

$$\frac{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S}{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;P}\, P\text{RBIA} \qquad \text{(no } \{\}\text{RBIA rule)} \qquad \frac{\Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S}{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;\top}\, \top\text{RBIA}$$

$$\frac{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;A_1 \qquad \Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;A_2}{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;A_1 \,\&\, A_2}\, \&\text{RBIA}$$

$$\frac{\Phi \overset{\text{RPA}}{\Longrightarrow} S_2/\Omega_2 \qquad \Phi;\Omega,\Omega_2 \overset{\text{RBIA}}{\Longrightarrow} S;A_1}{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;S_2 \multimap A_1}\, \multimap\text{RBIA} \qquad \frac{\Phi, \exists x{:}A_2;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;A_1}{\Phi;\Omega \overset{\text{RBIA}}{\Longrightarrow} S;\forall x{:}A_2.A_1}\, \forall\text{RBIA}$$

Figure 14: Reduction Checking Principal Judgments (2)

$\boxed{\Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} A}$ [Reduces (forward, clause)]

(no PRFCA rule) $\quad \dfrac{\Phi;\Omega \vdash S \ \mathsf{red}}{\Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} \{S\}}\ \{\}\text{RFCA} \quad$ (no $\top$RFCA rule)

$$\frac{\Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} A_1 \qquad \Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} A_2}{\Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} A_1 \ \& \ A_2}\ \&\text{RFCA}$$

$$\frac{\Phi \overset{\text{RPS}}{\Longrightarrow} S_2/\Omega_2 \qquad \Phi,\Omega_2;\Omega \overset{\text{RFCA}}{\Longrightarrow} A_1 \qquad \Phi;\Omega \overset{\text{RGS}}{\Longrightarrow} S_2}{\Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} S_2 \multimap A_1}\ \multimap\text{RFCA}$$

$$\frac{\Phi, \forall x{:}A_2;\Omega \overset{\text{RFCA}}{\Longrightarrow} A_1}{\Phi;\Omega \overset{\text{RFCA}}{\Longrightarrow} \forall x{:}A_2.A_1}\ \forall\text{RFCA}$$

Figure 15: Reduction Checking Principal Judgments (3)

nation property, then it's computation sequence is finite independently of whether it succeeds or fails.

In order to prevent obvious loops between backward- and forward-chaining, we require a global analysis that extracts a subordination relation $a_g \trianglelefteq a_c$ which holds if a clause with atomic head a_c has an atomic subgoal a_g, $\{\} \trianglelefteq a_c$ which holds if a clause with atomic head a_c has a subgoal that invokes saturation, and $a_g \trianglelefteq \{\}$ which holds if a forward-chaining clause has an atomic subgoal a_g. We next compute the transitive closure $\trianglelefteq^*$ of $\trianglelefteq$ and require that no asynchronous atom p satisfies both $p \trianglelefteq^* \{\}$ and $\{\} \trianglelefteq^* p$, and no synchronous atom q satisfies $\{\} \trianglelefteq^* q$. Intuitively, this partitions asynchronous atoms into those that use saturation (directly or indirectly) but do not occur as the subgoal of a forward-chaining clause and those that do not depend on saturation and may be used as subgoals in forward-chaining clauses, and no forward-chaining clause invokes saturation. Therefore if an asynchronous atom p invokes saturation, then no forward-chaining clause will attempt to solve p via backward-chaining. In particular, this disallows the pair of clauses $p \circ\!\!- \{S_1\}$ and $p \multimap \{S_2\}$ which would loop when trying to

solve p as follows.

$$\dfrac{\Gamma;\cdot;p \ll p \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma;\cdot \Rightarrow p}{\Gamma;\cdot \gg p}\Rightarrow\gg \qquad \cdots}{\Gamma;\cdot;p \multimap \{S_2\} < S_1}\multimap \mathbf{L}'}{\Gamma;\cdot \to S_1}u'}{\Gamma;\cdot \gg S_1}\{\}\mathbf{R}}{\Gamma;\cdot \gg \{S_1\}}\Rightarrow\gg}{\Gamma;\cdot;\{S_1\} \multimap p \ll p}\multimap \mathbf{L}}{\Gamma;\cdot \Rightarrow p}P\mathbf{R}_u$$

Similarly, the condition prevents forward-chaining clauses from invoking saturation as a subgoal because a synchronous atom q cannot occur in a clause $\{q\} \multimap \{q\}$ which would loop when trying to saturate and solve q as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma;\cdot \Rightarrow \{q\}}{\Gamma;\cdot \gg \{q\}}\Rightarrow\gg \qquad \cdots}{\Gamma;\cdot;\{q\} \multimap \{q\} < q}\multimap \mathbf{L}'}{\Gamma;\cdot \to q}u'}{\Gamma;\cdot \Rightarrow \{q\}}\{\}\mathbf{R}$$

After checking the global subordination condition, we verify that each clause is terminating. The principal judgment $\Phi;\Omega \vdash S\ \mathsf{term}$ (Figure 16) verifies the termination property of a clause by verifying backward- and forward-chaining chaining asynchronous clauses terminate. For backward-chaining clauses, the judgment $\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A;\Phi_g$ (Figure 18) accumulates a stack of subgoals into the context Φ_g and then appeals to the judgment $\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g$ (Figure 17) to check that the subclauses are terminating via the judgments $\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A \prec P$ and $\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S \prec P$ provided the subgoal atoms are known to be terminating or are recursive goals with strictly smaller arguments in the inductive argument, possibly using the reduction property of earlier (closer to the head) subgoals. Forward-chaining clauses are checked with the judgment $\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A;\Phi_g$ (Figure 19) by checking the termination of subclauses via the judgment $\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S \prec P$ and ensuring that unrestricted synchronous atoms generated by the clause satisfy

$\boxed{\Phi;\Omega \vdash S\ \mathsf{term}}$ [Termination (clause)]

$$\frac{}{\Phi;\Omega \vdash Q\ \mathsf{term}}\ Q\text{TERM} \qquad \frac{}{\Phi;\Omega \vdash ¡Q\ \mathsf{term}}\ ¡\text{TERM} \qquad \frac{}{\Phi;\Omega \vdash 1\ \mathsf{term}}\ 1\text{TERM}$$

$$\frac{\Phi;\Omega \vdash S_1\ \mathsf{term} \qquad \Phi;\Omega \vdash S_2\ \mathsf{term}}{\Phi;\Omega \vdash S_1 \otimes S_2\ \mathsf{term}}\ \otimes\text{TERM} \qquad \frac{\Phi, \forall x{:}A_2;\Omega \vdash S_1\ \mathsf{term}}{\Phi;\Omega \vdash \exists x{:}A_2.S_1\ \mathsf{term}}\ \exists\text{TERM}$$

$$\frac{(A\ \text{backward}) \qquad \Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A;\cdot}{\Phi;\Omega \vdash A\ \mathsf{term}}\ A\text{TERM}_b$$

$$\frac{(A\ \text{forward}) \qquad \Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A;\cdot}{\Phi;\Omega \vdash A\ \mathsf{term}}\ A\text{TERM}_f$$

$$\frac{(A\ \text{backward}) \qquad \Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A;\cdot}{\Phi;\Omega \vdash\ !A\ \mathsf{term}}\ !\text{TERM}_b$$

$$\frac{(A\ \text{forward}) \qquad \Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A;\cdot}{\Phi;\Omega \vdash\ !A\ \mathsf{term}}\ !\text{TERM}_f$$

Figure 16: Termination Checking Principal Judgments (1)

a specific subterm property. In particular, the judgment $\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} S;\Phi_g$ verifies that every argument term in the unrestricted synchronous atom $¡Q$ is no larger than the arguments of synchronous atoms in the subgoals. Intuitively, the *subheight* judgment $\Phi;\Omega \overset{\text{SH}}{\Longrightarrow} Q \sqsubseteq \Phi_g$ extracts the (maximum) depth of each existential variable in the head Q and subgoals Φ_g, and ensures that maximum depth of each existential variable in the head is less than or equal to the maximum depth in the antecedent. This ensures that the height of arguments is non-increasing and due to the restriction of the analysis to unrestricted hypotheses in forward-chaining clauses, this bounds the size of the context to all formulas with arguments of size less than or equal to those in the initial database.

$\boxed{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g}$ [Termination stack (backward or forward)]

$$\frac{}{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\cdot}\ \cdot\text{TKA} \qquad \frac{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g}{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g,\forall x{:}A}\ \forall\text{TKA}$$

$$\frac{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g}{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g,\exists x{:}A}\ \exists\text{TKA}$$

$$\frac{\Phi \overset{\text{RPA}}{\Longrightarrow} S/\Omega' \qquad \Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S \prec P \qquad \Phi;\Omega,\Omega' \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g}{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g,\exists!u{:}S}\ \exists!\text{TKA}$$

$\boxed{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A \prec P \text{ and } \Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S \prec P}$ [Termination (backward or forward, goal)]

$$\frac{\Phi;\Omega \longrightarrow \mathsf{term}(a\cdot Z') \prec \mathsf{term}(a\cdot Z)}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} a'\cdot Z' \prec a\cdot Z}\ P\text{TGA} \qquad \frac{a'\neq a \qquad a' \text{ terminating}}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} a'\cdot Z' \prec a\cdot Z}\ P\text{TGA}$$

$$\frac{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S \prec P}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} \{S\} \prec P}\ \{\}\text{TGA} \qquad \frac{}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} \top \prec P}\ \top\text{TGA}$$

$$\frac{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A_1 \prec P \qquad \Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A_2 \prec P}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A_1 \,\&\, A_2 \prec P}\ \&\text{TGA}$$

$$\frac{\Phi;\Omega \vdash S_2\ \mathsf{term} \qquad \Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A_1 \prec P}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} S_2 \multimap A_1 \prec P}\ \multimap\text{TGA} \qquad \frac{\Phi,\forall x{:}A_2;\Omega \overset{\text{TGA}}{\Longrightarrow} A_1 \prec P}{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} \forall x{:}A_2.A_1 \prec P}\ \forall\text{TGA}$$

$$\frac{}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} Q \prec P}\ Q\text{TGS} \qquad \frac{}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} ¡Q \prec P}\ ¡\text{TGS} \qquad \frac{}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} 1 \prec P}\ 1\text{TGS}$$

$$\frac{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S_1 \prec P \qquad \Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S_2 \prec P}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} S_1 \otimes S_2 \prec P}\ \otimes\text{TGS} \qquad \frac{\Phi,\exists x{:}A_2;\Omega \overset{\text{TGS}}{\Longrightarrow} S_1 \prec P}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} \exists x{:}A_2.S_1 \prec P}\ \exists\text{TGS}$$

$$\frac{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A \prec P}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} A \prec P}\ A\text{TGS} \qquad \frac{\Phi;\Omega \overset{\text{TGA}}{\Longrightarrow} A \prec P}{\Phi;\Omega \overset{\text{TGS}}{\Longrightarrow} !A \prec P}\ !\text{TGS}$$

Figure 17: Termination Checking Principal Judgments (2)

$\boxed{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A;\Phi_g}$ [Termination (backward, clause)]

$$\frac{\Phi,\Phi_g;\Omega \overset{\text{TKA}}{\Longrightarrow} P;\Phi_g}{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} P;\Phi_g}\ P\text{TBCA} \qquad \text{(no } \{\}\text{TBCA rule)} \qquad \frac{\Phi;\Omega \overset{\text{TKA}}{\Longrightarrow} a_\top;\Phi_g}{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} \top;\Phi_g}\ \top\text{TBCA}$$

$$\frac{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A_1;\Phi_g \qquad \Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A_2;\Phi_g}{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A_1 \,\&\, A_2;\Phi_g}\ \&\text{TBCA}$$

$$\frac{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A_1;\Phi_g,\exists!u{:}S_2}{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} S_2 \multimap A_1;\Phi_g}\ {\multimap}\text{TBCA} \qquad \frac{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} A_1;\Phi_g,\exists x{:}A_2}{\Phi;\Omega \overset{\text{TBCA}}{\Longrightarrow} \forall x{:}A_2.A_1;\Phi_g}\ \forall\text{TBCA}$$

Figure 18: Termination Checking Principal Judgments (3)

2.4 Examples

In this section we consider several programs and how to apply the mode, reduction, and termination analyses, as well as some programs which can't be handled by the current formulation.

The following forward-chaining program saturates the context with natural numbers between zero and nine.

$$\{¡\mathit{digit}\,(s^9 z)\}.$$
$$¡\mathit{digit}\,(s\,N) \multimap \{¡\mathit{digit}\,N\}.$$

Subordination determines $\mathit{digit} \trianglelefteq \{\}$ which is a valid relation because the synchronous atom *digit* does not invoke saturation. The first clause is terminating because it generates an unrestricted ground atom, so the condition on existential variables is trivial. The second clause is terminating because it generates an unrestricted ground atom and the depth of the variable N is zero in the head and one in the antecedent, so the $\Phi;\Omega \overset{\text{SH}}{\Longrightarrow} \mathit{digit}\,N \sqsubseteq \exists N{:}\mathit{nat}, \exists!u{:}\mathit{digit}\,(sN)$ judgment holds.

The following forward-chaining program generates the symmetric closure s as a synchronous atom of a relation r represented by an asynchronous atom.

$$!r\,X\,Y \multimap \{¡s\,X\,Y\}.$$
$$¡s\,X\,Y \multimap \{¡s\,Y\,X\}.$$

$\boxed{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A;\Phi_g}$ [Termination (forward, clause)]

(no PTFCA rule) $\dfrac{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} S;\Phi_g}{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} \{S\};\Phi_g}\ \{\}\text{TFCA}$ (no $\top$TFCA rule)

$$\frac{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A_1;\Phi_g \qquad \Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A_2;\Phi_g}{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A_1 \,\&\, A_2;\Phi_g}\ \&\text{TFCA}$$

$$\frac{\Phi,\Phi_g;\Omega \overset{\text{TGS}}{\Longrightarrow} S_2 \prec a_q \qquad \Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A_1;\Phi_g,\exists!u{:}S_2}{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} S_2 \multimap A_1;\Phi_g}\ \multimap\text{TFCA}$$

$$\frac{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} A_1;\Phi_g,\exists x{:}A_2}{\Phi;\Omega \overset{\text{TFCA}}{\Longrightarrow} \forall x{:}A_2.A_1;\Phi_g}\ \forall\text{TFCA}$$

$\boxed{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} S;\Phi_g}$ [Termination (forward, head)]

(no QTFHS rule) $\dfrac{\Phi;\Omega \overset{\text{SH}}{\Longrightarrow} Q \sqsubseteq \Phi_g}{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} ¡Q;\Phi_g}\ ¡\text{TFHS}$ $\dfrac{}{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} 1;\Phi_g}\ 1\text{TFHS}$

$$\frac{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} S_1;\Phi_g \qquad \Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} S_2;\Phi_g}{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} S_1 \otimes S_2;\Phi_g}\ \otimes\text{TFHS}$$

$\dfrac{\Phi,\exists x{:}A_2;\Omega \overset{\text{TFHS}}{\Longrightarrow} S_1;\Phi_g}{\Phi;\Omega \overset{\text{TFHS}}{\Longrightarrow} \exists x{:}A_2.S_1;\Phi_g}\ \exists\text{TFHS}$ (no ATFHS rule) (no !TFHS)

Figure 19: Termination Checking Principal Judgments (4)

The mode assignment $\mathsf{mode}(r \cdot (X; Y)) = -, -$ and the default assignment $\mathsf{mode}(s \cdot (X; Y)) = -, -$ are easily verified because no rule proves r and in both clauses the variables are made ground by the antecedents and therefore are ground in the head. Termination is similar to the above case, although this program shows that unlike backward-chaining programs, it is possible to change the order of an atom's arguments in forward-chaining clauses.

Similarly, the transitive closure t of a relation r can be shown to be well-moded and terminating. Moreover, it is possible to show that if the initial relation is reducing, then so is the transitive closure.

$$!r\,X\,Y \multimap \{¡t\,X\,Y\}.$$
$$¡t\,X\,Y \multimap ¡t\,Y\,Z \multimap \{¡t\,X\,Z\}.$$

Assuming $\mathsf{red}(r \cdot (X; Y)) = X \prec Y$, it is possible to show that this program satisfies the reduction property $\mathsf{red}(t \cdot (X; Y)) = X \prec Y$. In the first clause, t satisfies the reduction property $X \prec Y$ because r satisfies it. In the second clause, the inductive reduction properties $X \prec Y$ and $Y \prec Z$ can be used in the subterm sequent calculus to show t satisfies the reduction property $X \prec Z$.

The following mixed-search implementation of CKY parsing in LolliMon can be shown to be well-moded and terminating.

$$!\mathit{rule}\,X\,(\mathit{char}\,C) \multimap\ !\mathit{str}\,I\,C \multimap \{¡\mathit{parse}\,X\,I\,I\}.$$
$$!\mathit{rule}\,X\,(\mathit{jux}\,Y\,Z) \multimap ¡\mathit{parse}\,Y\,I\,J \multimap ¡\mathit{parse}\,Z\,(s\,J)\,K \multimap \{¡\mathit{parse}\,X\,I\,K\}.$$
$$\mathit{load}\,N\,\mathit{nil}\,S \mathrel{\circ\!\!-} \{¡\mathit{parse}\,S\,(s\,z)\,N\}.$$
$$\mathit{load}\,I\,(C :: \mathit{Cs})\,S \mathrel{\circ\!\!-} !(!\mathit{str}\,(s\,I)\,C \multimap \mathit{load}\,(s\,I)\,\mathit{Cs}\,S).$$
$$\mathit{start}\,\mathit{Cs}\,S \mathrel{\circ\!\!-} !\mathit{load}\,0\,\mathit{Cs}\,S.$$

It is possible to verify the clauses are well-moded by assigning modes to the asynchronous atoms

$$\mathsf{mode}(\mathit{str}\,(I; C)) = -, -$$
$$\mathsf{mode}(\mathit{rule}\,(X; R)) = -, -$$
$$\mathsf{mode}(\mathit{load}\,(I; \mathit{Cs}; S)) = +, +, -$$
$$\mathsf{mode}(\mathit{start}\,(\mathit{Cs}; S)) = +, -$$

and the necessary assignment $\mathsf{mode}(\mathit{parse} \cdot (X; I; J)) = -, -, -$ for synchronous atoms. Since *str* and *rule* do not occur in the head of any clause, they trivially satisfy the mode specification. The first two clauses are well-moded

by analyzing each clause from left to right and and establishing that the variables in the head are ground after the subgoals have been solved. The third clause is well-moded because *parse*'s mode guarantees S is ground after saturation. The last two clauses are well-moded by standard reason for backward-chaining clauses. For termination, the subordination analysis determines $str, rule \trianglelefteq parse \trianglelefteq \{\} \trianglelefteq load \trianglelefteq start$ which satisfies the global subordination condition because each asynchronous atom ($str, rule, load, start$) occurs on only one side of $\{\}$ in the relation and the synchronous atom (*parse*) is below $\{\}$. The atoms *str* and *rule* are trivially terminating because they do not occur at the head of any clause. The first clause is terminating because the depth of of the variables X and I in the conclusion is zero, which is equal to their height in the antecedents. Similarly, the second clause satisfies the the depth condition because the existential variables in the head occur at depth zero. The third clause is terminating because the subgoal involves a strictly subordinated atom. The fourth clause is terminating by specifying $\mathsf{term}(load \cdot (I; Cs; S)) = Cs$ because the second argument acts as the induction variable, which is also trivially satisfied by the third clause. The higher-order subgoal of the fourth clause also checks the termination of the subclause $str\,(s\,I)\,C$. Finally, the fifth clause is terminating because the subgoal involves a strictly subordinated atom.

One example that can't be shown to terminate is $¡q\,(s\,X)\,Y \multimap \{¡q\,X\,(sY)\}$, which converges by using the first argument as an induction variable, but our analysis rejects the program because the depth of Y in the head is greater than its depth in the antecedent and it is not possible to specify termination orders of synchronous atoms. Handling forward-chaining clauses that generate linear hypotheses is subtle because of the interaction between linear consumption and linear production. For example, the clauses $\top \multimap \{q\}$ and $p \supset \{q\}$ are always applicable (assuming p is always satisfiable) and saturation would generate an unbounded number of copies of q. This means that a forward-chaining clause that generates linear hypotheses must have at least one non-trivial atomic linear subgoal. However, this is not sufficient because $p \multimap \{q\}$ could still diverge if p is always satisfiable and $q \multimap \{q\}$ clearly loops. The latter example is also problematic because clauses such as $q \multimap q'\,(s\,X) \multimap \{q \otimes q'\,X\}$ consume a token from the context and immediately return it, although the database converges because the second argument is strictly decreasing. Again, the convergence of the clause is conditional on the linearity of q', because the clause would always be applicable if q' were unrestricted. Since linear goals may be solved by unrestricted means, it would be necessary to perform another static analysis that determines whether an atom may occur in the linear or unrestricted

context, or alternatively require a block declaration that makes the shape of the context explicit. Some programming idioms do not satisfy the subterm property between the head and subgoals but still converge because the size of the context is strictly decreasing. For example $p \multimap q\,X \multimap \{q\,(s\,X)\}$ informally counts the number of p's in the context and converges if p is truly linear, although the depth of X in the head is not less than or equal to its height in the antecedents. Due to the variety of termination arguments for forward-chaining clauses with linear heads, it may be necessary to provide multiple ways of specifying the termination property according to which antecedents are truly linear, whether a rule shrinks the context or generates hypotheses with strictly smaller arguments, and addressing the interaction between clauses that terminate for different reasons.

3 Conclusion

We have presented a variant of LolliMon with synchronous atoms and generalized linear implication together with an operational semantics that makes saturation and quiescence explicit, as well as mode, reduction, and termination analyses that handle the mixed backward-chaining and forward-chaining search behavior. The termination analysis only accounts for backward-chaining clauses and forward-chaining clauses that generate unrestricted synchronous atoms, and we have discussed design considerations for handling the general forward-chaining case. The next step will be to prove soundness and completeness of the explicit operational semantics with respect to the original semantics, extend termination analysis to handle general forward-chaining clauses, and prove the correctness of the analyses with respect to the operational semantics. The reduction and termination analyses could be revised using contextual modal type theory [12] instead of the mixed-prefix context presentation.

Termination of forward-chaining logic programs can be refined to establish asymptotic time complexity bounds on the running time of saturating logical algorithms. Ganzinger and McAllester [11, 7, 8] give metacomplexity results for forward-chaining first-order logic programming languages. The first results deal with bottom-up search without deletion and union-find primitives, a richer language incorporates deletion of antecedent formulas and rules with fixed priority which enable an encoding of union-find, and a further extension allows deletion of arbitrary formulas (not only those that appear in the antecedent) and variable priority as a function of the object terms. The metacomplexity results permit the analysis of sat-

urating logic programs whose complexity is given in terms of the size of the initial database and the number of *prefix firings* (the number of ways the antecedent of a clause can be satisfied). The metacomplexity theorems are proven by translating logic programs into a restricted fragment of the language and giving an algorithm that generates the completed database within the purported time complexity. Their example saturating logical algorithms be encoded in LolliMon by simulating deletion with linear resource consumption, although LolliMon does not provide a logical foundation for rule priorities. In their setting deletion is irrevocable so a sufficient termination criterion is that terms in the conclusion are subterms (not necessarily proper) of the terms in the antecedents. Termination for saturating LolliMon logic programs may be verified by means of a similar metacomplexity result, although linearity will likely require a more complicated subterm criterion because linear production and consumption are complementary and certain LolliMon programs may oscillate between states even though the linear context remains bounded. The Ganzinger and McAllester algorithm for computing the saturated completion of a database according to rules with deletion may be adapted to the linear setting by relaxing the saturation and quiescence conditions and employing model checking techniques for determining when a local fixed point has been reached, even if the context oscillates. Another direction for research will be to extend the operational semantics as well as the static analyses to the dependent case of CLF.

Termination analysis of backward-chaining logic programs may also be refined into a *nondeterministic* time complexity metatheorem by considering the running time of a clause as a recurrence relation in terms of the running time of the subgoals. This approach would be similar to the complexity analyses of Debray, López García, Hermenegildo, and Lin [6, 4, 5] and could employ the techniques for automatic solution of recurrence relations developed by Bagnara, Zaccagnini, and Zolo [3, 2].

Once the theoretical foundations for termination and complexity analysis in LolliMon are settled, a termination checker and complexity synthesizer can be implemented in LolliMon, the algorithm in the proof of the metacomplexity theorem can be implemented separately or as a replacement for the existing LolliMon saturation/quiescence engine, and the subterm analysis can be extended to CLF proof terms.

4 Acknowledgments

I am grateful to thank Frank Pfenning for introducing me to the topic and discussing existing work, and to Brigitte Pientka for explaining the design considerations of reduction and termination checking for Twelf.

5 References

[1] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Log. Comput.*, 2(3):297–347, 1992.

[2] Roberto Bagnara and Alessandro Zaccagnini. Checking and bounding the solutions of some recurrence relations. Quaderno 344, Dipartimento di Matematica, Universit di Parma, Italy, 2004.

[3] Roberto Bagnara, Alessandro Zaccagnini, and Tatiana Zolo. The automatic solution of recurrence relations. I. Linear recurrences of finite order with constant coefficients. Quaderno 334, Dipartimento di Matematica, Universit di Parma, Italy, 2003.

[4] Saumya K. Debray, Pedro López García, Manuel Hermenegildo, and Nai-Wei Lin. Estimating the computational cost of logic programs. In Baudoin Le Charlier, editor, *Proceedings of the First International Static Analysis Symposium*, pages 255–265. Springer Verlag, 1994.

[5] Saumya K. Debray, Pedro López García, Manuel Hermenegildo, and Nai-Wei Lin. Lower Bound Cost Estimation for Logic Programs. In *International Symposium on Logic Programming*, oct 1997.

[6] Saumya K. Debray and Nai-Wei Lin. Cost analysis of logic programs. *ACM Transactions on Programming Languages and Systems*, 15(5):826–875, November 1993.

[7] Harald Ganzinger and David A. McAllester. A new meta-complexity theorem for bottom-up logic programs. In *IJCAR*, pages 514–528, 2001.

[8] Harald Ganzinger and David A. McAllester. Logical algorithms. In *ICLP*, pages 209–223, 2002.

[9] Jean-Pierre Jouannaud and Albert Rubio. The higher-order recursive path ordering. In G. Longo, editor, *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS'99)*, pages 402–411, Trento, Italy, 1999. IEEE Computer Society Press.

[10] Pablo López, Frank Pfenning, Jeff Polakow, and Kevin Watkins. Monadic concurrent linear logic programming. In *PPDP*, pages 35–46, 2005.

[11] David A. McAllester. On the complexity analysis of static analyses. *J. ACM*, 49(4):512–537, 2002.

[12] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. In *Transactions on Computational Logic*. ACM, sep 2005. Submitted.

[13] Frank Pfenning. Logic programming in the LF logical framework. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

[14] Frank Pfenning and Carsten Schürmann. System description: Twelf — A meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, 1999. Springer-Verlag LNAI 1632.

[15] Brigitte Pientka. Termination and reduction checking for higher-order logic programs. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *IJCAR*, volume 2083 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 2001.

[16] Brigitte Pientka. Verifying termination and reduction properties about higher-order logic programs. *J. Autom. Reasoning*, 34(2):179–207, 2005.

[17] Brigitte Pientka and Frank Pfenning. Termination and reduction checking in the logical framework. In Carsten Schürmann, editor, *Workshop on Automation of Proofs by Mathematical Induction*, Pittsburgh, Pennsylvania, jun 2000.

[18] Lutz Plumer. *Termination Proofs for Logic Programs*, volume 446 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1990.

[19] Ekkehard Rohwedder and Frank Pfenning. Mode and termination checking for higher-order logic programs. In *ESOP*, pages 296–310, 1996.

[20] Jaco van de Pol and Helmut Schwichtenberg. Strict functionals for termination proofs. In Mariangiola Dezani-Ciancaglini and Gordon D. Plotkin, editors, *TLCA*, volume 902 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 1995.

[21] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.