Coinductive Uniform Proofs: Revision of Annual Technical Report Submitted for PhD Progression, Which Has New Formulation of Sequent Rules

Yue Li
Heriot-Watt University
Edinburgh, Scotland, UK
yl55@hw.ac.uk

June 3, 2018

Executive Summary

Background First-order Horn clause logic programming is about proving logic formulae using a logic inference rule called SLD-resolution. Given an atomic formula A to be proven, one step of SLD-resolution yields a possibly empty set of formulae $A_1, \ldots, A_n$ ($n \geq 0$), the conjunction of which is a sufficient condition for A. Then $A_1, \ldots, A_n$ become formulae to be proven, and SLD-resolution is attempted on $A_1, \ldots, A_n$ separately, again looking for sufficient conditions for each of them, and so on, until there is no formula to be proven, then it is concluded that the initial formula A is proven. A sequence of steps of SLD-resolution is called a SLD-derivation. A formula A is proven if there exist a terminating, i.e. finite length, SLD-derivation starting with A and ending with the empty set.

Motivation However, non-termination of SLD-derivation is common in logic programming and in applications of logic programming, such as programming language type class inference. It is useful to know for sure that a *possibly* non-terminating SLD-derivation is *indeed* non-terminating. A sufficient condition for non-termination is that we can find a pattern from the shape of a segment of SLD-derivation and show that the segment in concern can be extended forever according to this pattern.

Looking for patterns is generally easier for trained human than for computers, but we want to empower computers with this faculty so that they can exhibit some artificial intelligence. In our specific case, we are trying to program computers to recognize patterns in SLD-derivations. In other words, we are trying to automate *coinductive reasoning.* The state of the art in this field of research is that, if SLD-derivations have periodical shapes, then their patterns can be recognized, and, if SLD-derivations are restricted to term rewriting and assume non-periodical shapes, then their patterns can be recognized . These are important steps forward, being theoretically instructive, and has many applications. However, there is still better performance that can be expected from computers, and still there is further achievement that can be made by us researchers. For instance, currently there is no solution if SLD-derivations are *not* restricted to term rewriting and do *not* assume periodical shapes.

Contribution Our contribution in this report is an encouraging theory, which is based on reflection and abstraction of the mentioned state of the art practices in the field, and which paves the way to solving the open problem, by showing that for different kinds of SLD-derivations, those periodic and those non-periodic, those restricted to term rewriting and those using full SLD-resolution, their patterns are all the same in an abstract level, and the ways to build their patterns automatically are all the same, too, in an abstract level. For instance, given an arbitrary potentially non-terminating SLD-derivation, building its pattern would require identifying a special formula M called a *coinductive invariant*, and the pattern itself is then given by a *coinductive uniform proof* of M.

Technically, we extend the uniform proof by Dale Miller et al with a coinductive rule, and syntactically extend their four abstract logic programming languages with fixed-point terms modeling infinite objects. We show that non-terminating SLD-derivations can be categorized by the minimum abstract language in which their coinductive invariants can be expressed and proved, which implies that different categories of non-terminating SLD-derivations can all be captured by the most general abstract language, i.e. the language of higher-order hereditary Harrop formula. We justify our proof system by soundness theorems with respect to an adapted version of the greatest complete Herbrand models, the standard version of which are the declarative semantics of coinductive first-order Horn clause logic programming, and the adaptation is due to our generalization of the concept of logic programs in order to allow asserting coinductively proved lemmas to the program, and that such lemmas may not be first-order Horn clause.

Methodology Researching is essentially about *creation*, and it is not essentially about *reading* others' work. Researchers work on open problems, which are problems whose answers do *not* actually exist, and therefore cannot be found straight from other people's work. For this fact, a researcher should not expect to find the exact answer for his or her problems by referring to information sources. Further, the responsibility of researchers is to *create* answers following a creative process. Discussing by comparison, the fundamental difference between research questions and exam questions is that the answer for the latter has already been created by someone, and can be found directly by referring to the work of others, which is not the case for the former. Answering exam questions and doing research, as a result, are fundamentally different intellectual activities, the former is mainly about referencing, but the latter is about creation.

The technical contribution of this report is a product of creative work. Neither the *theory of coinductive uniform proof*, nor the *soundness theorems*, has ever existed before. The creation of these new knowledge follows a process of learning, experimentation and observation.

Learning is about reading others' work. Papers by other people are about answers that they created for the open questions solved by them. By reading others' work, one can learn about other people's successful approach to their research questions, and some of the methods and conclusions can be reused to help solve one's own open problem. Regarding the work reported here, we obtained some components from existing work, such as Dale Miller et al's uniform proof, the fixed point term definition from Benjamin Pierce, and Fu et al's proof relevant co-recursive resolution; we also learned proof techniques from the work of Alfred Tarski, regarding lattice theoretic fixed point manipulations, and John Lloyd, regarding constructing the limit of an infinite series of finite trees.

Observation played an important role in the creation of the proof of our main soundness theorem. Observation cannot be substituted by reading,it is indispensable, and it is the signature activity in the creative process. Reusing others' work only scratches the skin of the open problems, but observation cracks the shell. By observing behavior of concrete examples related to one's open problem, one can gain the knowledge that can be found from nowhere else, and such knowledge, for this reason, constitutes the original contribution of the researcher.

Experimentation is a trail-and-error process that helped formulate the coinductive uniform proof system. It also played a role in the formulation of several proofs. Experiments test ideas and reveal new facts. Another tactic that is used is *progressive approximation*, which can be put under the umbrella of experimentation. This method has been used in previous software development projects to manage complexity, which is a need that also presented when conducting the reported work. For instance, the main soundness proof is a result of progressive approximation of four steps, where each step proves a related but simpler theorem which provides insights that facilitate further proof of a related but more complicated theorem. Another main theorem also saw its proof developed in this way. From a retrospective view, I conclude that if progressive approximation is used, it means that the final product (proof) is too complicated to be contained by the researcher's mind at the first attempt, rather, the researcher need to walk step by step, starting with a level of complexity that is manageable by the mind, then build upon it, so that the mind is conditioned and prepared for the next level of complexity.

Acknowledgment

My warmest praise goes to my supervisor Dr. Ekaterina Komendantskaya, for her grace, mercy and wisdom. Dr. Henning Basold kindly offered professional and artistic advice regarding the technicalities of this report.

Contents

Chapter 1

Introduction

Coinductive logic programming aims to find finite representations for infinite SLD-derivations [9, 11]. This, given a logic program, involves identifying a special formula, called a *coinductive invariant*, and adopting a *coinductive proof principle*, so that we can prove the coinductive invariant using the coinductive proof principle in addition to other logic inference rules. The proof of the coinductive invariant can serve as a finite representation of a corresponding infinite derivation that is constructed without using the coinductive proof principle. Let us see a series of examples of coinductive logic programming, of different kinds.

Example 1 (Sand Clock)**.** *Consider the* Sand Clock *program consisting of first-order Horn clauses 1.1 and 1.2 on signature* $\{clock : \iota \to \iota \to o, 0 : \iota, s : \iota \to \iota\}$*.*

$$\forall m \quad clock\ m\ 0 \supset clock\ 0\ m \tag{1.1}$$

$$\forall mn \quad clock\ n\ (s\ m) \supset clock\ (s\ n)\ m \tag{1.2}$$

Given a goal: clock $(s\ 0)\ 0$*, using standard SLD-resolution of logic programming (more precisely, SLD-resolution naturally restricted to term rewriting [11]), we have the following cyclic derivation:*

$$clock\ (s\ 0)\ 0 \overset{1.2}{\to} clock\ 0\ (s\ 0) \overset{1.1}{\to} clock\ (s\ 0)\ 0 \overset{1.2}{\to} \dots$$

This infinite derivation can be captured by a coinductive proof of the coinductive invariant:

$$clock\ (s\ 0)\ 0 \tag{1.3}$$

as follows:

$$clock\ (s\ 0)\ 0 \overset{1.2}{\to} clock\ 0\ (s\ 0) \overset{1.1}{\to} clock\ (s\ 0)\ 0 \overset{1.3}{\to} \Box$$

It is intuitive to see that the coinductive proof captures the infinite SLD-derivation.

Example 2 (Signal Amplifier)**.** *Consider the* Signal Amplifier *program consisting of first-order Horn clauses 1.4, 1.5 and 1.6 on signature* $\{scons : \iota \to \iota \to \iota, add : \iota \to \iota \to \iota \to o, lift : \iota \to \iota \to \iota \to o, 0 : \iota, s : \iota \to \iota\}$ *(note that* $[x|y]$ *is short for:* $scons\ x\ y$*).*

$$\forall xyz \forall in \forall out \quad lift\ x\ in\ out \wedge add\ x\ y\ z \supset lift\ x\ [y|in]\ [z|out] \tag{1.4}$$

$$\forall x \quad add\ 0\ x\ x \tag{1.5}$$

$$\forall xyz \quad add\ x\ y\ z \supset add\ (s\ x)\ y\ (s\ z) \tag{1.6}$$

The goal: $lift\ (s\ 0)\ (n\_str\ 0)\ out$*, feeds a constant discrete-time signal* $(n\_str\ 0)$*, denoting a stream of 0, into the amplifier, which then increases the value of each quantity by 1, i.e.* $(s\ 0)$*. The SLD-derivation (involving computation of answer substitutions, thus not restricted to term rewriting [11]) is infinite, and some initial steps are as follows:*

$$lift\ (s\ 0)\ (n\_str\ 0)\ out \xrightarrow{1.4} lift\ (s\ 0)\ (n\_str\ 0)\ out' \wedge add\ (s\ 0)\ 0\ z'$$
$$\xrightarrow{1.6} lift\ (s\ 0)\ (n\_str\ 0)\ out' \wedge add\ 0\ 0\ z'' \xrightarrow{1.5} lift\ (s\ 0)\ (n\_str\ 0)\ out' \xrightarrow{1.4} \dots$$

A substitution — $out := [(s\ 0)|out']$*, is computed from the steps printed above. It seems that the two goals* $(lift\ (s\ 0)\ (n\_str\ 0)\ out)$ *and* $(lift\ (s\ 0)\ (n\_str\ 0)\ out')$ *do not form a cycle since they are not* α*-equivalent. However, as the SLD-derivation infinitely goes on, both variables: out and out′, are being instantiated towards the same limit term* $(n\_str\ (s\ 0))$*, i.e. a stream of 1. Given this fact, the infinite SLD-derivation actually has the shape:*

$$lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \xrightarrow{1.4} lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \wedge add\ (s\ 0)\ 0\ (s\ 0)$$
$$\xrightarrow{1.6} lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \wedge add\ 0\ 0\ 0 \xrightarrow{1.5} lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \xrightarrow{1.4} \dots$$

where there is indeed a cycle featuring the coinductive invariant:

$$lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \tag{1.7}$$

Therefore the infinite SLD-derivation can be captured by a coinductive proof of 1.7 as follows:

$$lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \xrightarrow{1.4} lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \wedge add\ (s\ 0)\ 0\ (s\ 0)$$
$$\xrightarrow{1.6} lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \wedge add\ 0\ 0\ 0 \xrightarrow{1.5} lift\ (s\ 0)\ (n\_str\ 0)\ (n\_str\ (s\ 0)) \xrightarrow{1.7} \Box$$

Comparing infinite SLD-derivations from Examples 1 and 2, the similarity is that they are both cyclic, while the difference is that the former does not involve infinite terms, but the latter does.

Example 3 (Donuts Lover)**.** *Consider the* Donuts Lover *program consisting of first-order Horn clauses 1.8, 1.9 and 1.10 on signature* $\{eat : \iota \to o, like : \iota \to o, 0 : \iota, next : \iota \to \iota\}$.

$$eat\ 0 \tag{1.8}$$
$$\forall d \quad eat\ d \supset eat\ (next\ d) \tag{1.9}$$
$$\forall d \quad eat\ d \wedge like\ (next\ d) \supset like\ d \tag{1.10}$$

Recursively, we use $(next^m\ 0)$ *as a short hand of* $(next\ (next^{m-1}\ 0))$ *for* $m > 0$*, and use* $(next^0\ 0)$ *as synonym for* 0*. We use* $\to_m$ $(m \geq 0)$ *to denote* m *steps of standard SLD-reduction (naturally restricted to term rewriting [11]). The SLD-derivation of: like* $(next^m\ 0)$ *where* $m \geq 0$*, is non-terminating. For any* $m \geq 0$*, we have:*

$$like\ (next^m\ 0) \overset{1.10}{\to} eat\ (next^m\ 0) \wedge like\ (next^{m+1}\ 0)$$
$$\overset{1.9}{\to}_m eat\ 0 \wedge like\ (next^{m+1}\ 0) \overset{1.8}{\to} like\ (next^{m+1}\ 0) \to \dots$$

Since the number of steps required to reduce: eat $(next^m\ 0)$*, to: eat* 0*, increases in every iteration, this derivation is not cyclic. The coinductive invariant for this program is given as:*

$$\forall d \quad eat\ d \supset like\ d \tag{1.11}$$

The coinductive proof of the coinductive invariant is as follows:

$$\forall d\ eat\ d \supset like\ d \overset{\forall R}{\to} eat\ c \supset like\ c \overset{\supset R}{\to} like\ c \overset{1.10}{\to} eat\ c \wedge like\ (next\ c)$$
$$\overset{1.12}{\to} like\ (next\ c) \overset{1.11}{\to} eat\ (next\ c) \overset{1.9}{\to} eat\ c \overset{1.12}{\to} \square$$

where the $\forall R$ *step adds the fresh constant (known as an eigenvariable)* $c : \iota$ *to the signature, while the temporary clause 1.12:*

$$eat\ c \tag{1.12}$$

is a product of the $\supset R$ *step. It is less intuitive, though, to appreciate that the coinductive proof captures the infinite derivation.*

Example 4 (From)**.** *Consider the* From *program consisting of first-order Horn clause 1.13 on signature* $\{from : \iota \to o, 0 : \iota, s : \iota \to \iota, scons : \iota \to \iota \to \iota\}$.

$$\forall xt \quad from\ (s\ x)\ t \supset from\ x\ [x|t] \tag{1.13}$$

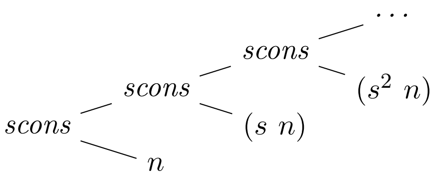
The goal: from $0\ t_0$*, has an infinite SLD-derivation (not restricted to term rewriting) whose initial steps are as follows:*

$$from\ 0\ t_0 \overset{1.13}{\to} from\ (s\ 0)\ t_1 \overset{1.13}{\to} from\ (s^2\ 0)\ t_2 \overset{1.13}{\to} \dots \overset{1.13}{\to} from\ (s^m\ 0)\ t_m \overset{1.13}{\to} \dots$$

A series of substitutions are computed from the printed steps:

$$t_0 := [0|t_1], \quad t_1 := [(s\ 0)|t_2], \quad \ldots, \quad t_{m-1} := [(s^{m-1}\ 0)|t_m], \quad \ldots$$

There are infinite amount of such substitutions, and the effect of their accumulation is such that, if we use $(\mathit{fr\_str}\ n)$ *to represent the infinite tree:*



then, the shape of the infinite SLD-derivation is as follows, which does not contain any cycle:

$$\mathit{from}\ 0\ (\mathit{fr\_str}\ 0) \overset{1.13}{\rightarrow} \mathit{from}\ (s\ 0)\ (\mathit{fr\_str}\ (s\ 0)) \overset{1.13}{\rightarrow} \mathit{from}\ (s^2\ 0)\ (\mathit{fr\_str}\ (s^2\ 0)) \overset{1.13}{\rightarrow} \ldots \overset{1.13}{\rightarrow} \mathit{from}\ (s^m\ 0)\ (\mathit{fr\_str}\ (s^m\ 0)) \overset{1.13}{\rightarrow} \ldots$$

We capture the infinite derivation by proving the coinductive invariant:

$$\forall x \quad \mathit{from}\ x\ (\mathit{fr\_str}\ x) \tag{1.14}$$

as follows:

$$\forall x\ \mathit{from}\ x\ (\mathit{fr\_str}\ x) \overset{\forall R}{\rightarrow} \mathit{from}\ c\ (\mathit{fr\_str}\ c) \overset{1.13}{\rightarrow} \mathit{from}\ (s\ c)\ (\mathit{fr\_str}\ (s\ c)) \overset{1.14}{\rightarrow} \square$$

where $c : \iota$ *is an eigenvariable. Note that the expressions* $(\mathit{fr\_str}\ c)$ *and* $(\mathit{scons}\ c\ (\mathit{fr\_str}\ (s\ c)))$ *denote the same infinite term, and this is part of the reasoning behind the step:*

$$\mathit{from}\ c\ (\mathit{fr\_str}\ c) \overset{1.13}{\rightarrow} \mathit{from}\ (s\ c)\ (\mathit{fr\_str}\ (s\ c))$$

Examples 3 and 4 both involve non-cyclic infinite SLD-derivations, while they differ in that the former does not involve infinite terms but the latter does.

For programs like in Examples 1, 2 and 3, there exist heuristic algorithms [9, 7] to discover coinductive invariants and perform coinductive proof. For Example 4, currently there is *no* such algorithms.

Logic programming features goal-directed proof search, meaning, given a goal formula G that contains some top-level logical connective, the sub-goal(s) to prove next is solely determined by the top-level logical connective of G. A proof built in this manner is called a *uniform proof.* Miller et. al. [16] identified four sets of logic formulae, for each of which, a uniform proof of a goal formula G exists if and only if a classical or intuitionistic proof of G

exists. For this reason, uniform proof serves as a proof-theoretic framework for logic programming, and the four sets of formulae serve as four different abstract logic programming languages. These sets of formulae are: first-order Horn clause (*fohc*), higher-order Horn clause (*hohc*), first-order hereditary Harrop formula (*fohh*) and higher-order hereditary Harrop formula (*hohh*).

Logic programming is usually within *fohc* [13]. However, when we study coinduction in *fohc*, we often encounter *fohc* programs that have coinductive invariants that cannot be captured by the syntax of *fohc* goal formulae, and thus cannot be proved by just using the inference rules of *fohc* (cf. Examples 2, 3 and 4). Rather, we found that such coinductive invariants can usually be captured by the syntax of goal formulae of a syntactically richer logic, like *fohh* and *hohh*, and then be proved using additional inference rules thereof. To elaborate on this discovery, we herein formally specify the proof system in which we can observe these phenomena and conduct these proof.

The proof system in concern is called *coinductive uniform proof*, obtained by firstly extending the uniform proof system by Miller et. al. [16, 15] with a coinductive proof principle called CO-FIX (COinductive FIXed-point). The CO-FIX rule is a variety of the Löb rule [20], presented below, whose varieties are involved in many coinductive calculi and algorithms of automated inference in first-order logic [5, 8, 1].

$$\frac{\Sigma; \Gamma, \lhd F \longrightarrow F}{\Sigma; \Gamma \longrightarrow F}\ \textit{Löb}$$

In the Löb rule. Γ is a first-order theory, F is a first-order formula, and $\lhd F$ is a coinductive hypothesis that can be applied later in the proof. The main challenge involved in augmenting uniform proof with the Löb rule is preservation of uniformity in goal-directed proof search, which is the principle underlying uniform proof.

In order to capture some productive non-terminating SLD-derivations [11] (also cf. Examples 2 and 4), we further include in our proof system the mechanism to encode infinite first-order terms as (possibly higher-order) finite lambda terms. For instance, we enrich the syntax of logic formulae of Miller et. al. with the *fix* primitive [18, 6] to denote fixed-points of lambda terms.

The contribution of this report is thus towards meta-theory of coinduction in first-order Horn clause logic programming: we formulate a proof-theoretic framework for coinduction exemplified in Examples 1, 2, 3 and 4, and explain, by means of soundness theorems and particularly proofs of these theorems, the reason that, for instance, coinductive proofs in Examples 1, 2, 3 and 4 amount to finite representation of corresponding infinite SLD-derivations. The reported theoretic work prepares for future development of heuristic algorithms that can fill the gap left by currently available algorithms.

The rest of the report is planned as follows. We introduce our term system in Section 2. In Section 3 we present the syntax of our extended logic languages and rules of the coinductive uniform proof system. Section 4 discusses soundness results of our proof system and Section 5 gives full proof for the main theorem.

Chapter 2

Preliminaries, Fixed-point Terms

We formalize logic formulae and infinite terms involved in first-order Horn clause logic programming [14] as finite lambda terms. Our lambda term system is simply typed lambda calculus [17, 15] extended with the *fix* primitive [18, 6].

Definition 5 (Types)**.** Let $\mathbb{V} = \{o, \iota, \ldots\}$ be a countable set of *type variables*, and $\mathbb{T}$ be the set of all *simple types* (in short, *types*) in the abstract syntax:

$$\mathbb{T} ::= \mathbb{V} \mid (\mathbb{T} \to \mathbb{T})$$

We use τ, σ and variants thereof to denote an arbitrary type.

Definition 6 (Functional\non-functional types)**.** A type $\sigma \in \mathbb{V}$ is called a *non-functional* type. A type of the form $\tau \to \tau'$ is called a *functional* type.

Definition 7 (Argument\target types)**.** Given the right associativity of the functional type constructor $(\to)$, we can remove all redundant parentheses and depict a type τ in the form $\tau_1 \to \cdots \to \tau_n \to \sigma \;\; (n \geq 0)$, where the non-functional type σ is the *target type* of τ, and $\tau_1, \ldots \tau_n$ are *argument types* of τ.

Definition 8 (Order of types, $\mathit{Ord}\,(\tau)$)**.** The *order* of a type τ, denoted $\mathit{Ord}\,(\tau)$, is

$$\begin{aligned} \mathit{Ord}\,(\tau) &= 0 \quad \text{(provided that } \tau \text{ is non-functional)} \\ \mathit{Ord}\,(\tau_1 \to \tau_2) &= \max\{\mathit{Ord}\,(\tau_1) + 1,\; \mathit{Ord}\,(\tau_2)\} \end{aligned}$$

Example 9 (Order of types)**.**

$$\mathit{Ord}\,(\iota) = 0 \qquad \mathit{Ord}\,(\iota \to \iota) = 1 \qquad \mathit{Ord}\,((\iota \to \iota) \to \iota) = 2$$

To be able to work with infinite data structures like streams, we extend the syntax of lambda terms with the *fix* primitive, which behaves in a similar way as fixed-point operators in untyped lambda calculus.

Definition 10 (Terms)**.** Let $\mathit{Var} = \{x, y, z, \ldots\}$ be a countable set of *variables*, and $\mathit{Cst} = \{a, b, c, \ldots\}$ be a countable set of *constants* . The set Λ of all *lambda terms* (in short: *terms*) is defined by

$$\Lambda ::= \mathit{Var} \mid \mathit{Cst} \mid (\Lambda\ \Lambda) \mid (\lambda\, \mathit{Var} \,.\, \Lambda) \mid (\mathit{fix}\ \Lambda)$$

A term of the form $\Lambda\ \Lambda$ is called an *application*, and a term of the form $\lambda\, \mathit{Var} \,.\, \Lambda$ – *abstraction*. A term of the form $\mathit{fix}\ \Lambda$ is called a *fixed-point term*. We use letters M, N, L and variants thereof to denote members of Λ.

There are two schools of syntactic definition for fixed-point terms [2]: one school introduces a special binder, say *fix*, in addition to the existing binder λ, and defines fixed-point terms as having the shape $\mathit{fix}\ x \,.\, M$ which is syntactically similar to $\lambda x \,.\, M$ except the binder; the other school [18], which is followed here, regards *fix* as a prefixing unary operator that behaves like fixed-point operators that cannot be typed using simple types [17].

Definition 11 (Free variable, $FV(\cdot)$)**.** Given a term M, the set $FV(M)$ of *free variables* is defined in the standard way as:

1. $FV(x) = \{x\}$ if $x \in \mathit{Var}$
2. $FV(c) = \emptyset$ if $c \in \mathit{Cst}$
3. $FV(M\ N) = FV(M) \cup FV(M)$
4. $FV(\lambda x \,.\, M) = FV(M) \setminus \{x\}$
5. $FV(\mathit{fix}\ M) = FV(M)$

Definition 12 (Sub-term, $\mathit{Sub}(\cdot)$)**.** Given a term M, the set $\mathit{Sub}(M)$ of sub-terms of M is defined in the standard way as:

1. $\mathit{Sub}(x) = \{x\}$ if $x \in \mathit{Var}$ or $x \in \mathit{Cst}$
2. $\mathit{Sub}(M\ N) = \{M\ N\} \cup \mathit{Sub}(M) \cup \mathit{Sub}(N)$
3. $\mathit{Sub}(\lambda x \,.\, M) = \{\lambda x \,.\, M\} \cup \mathit{Sub}(M)$
4. $\mathit{Sub}(\mathit{fix}\ M) = \{\mathit{fix}\ M\} \cup \mathit{Sub}(M)$
5. $\mathit{Sub}(Z) = \{Z\}$ if Z is not analyzable.

Definition 13 (Open\closed term)**.** Term M is *closed*, if $FV(M) = \emptyset$. Otherwise, M is *open*.

Example 14 (Open\closed terms)**.** *$\mathit{fix}\ \lambda x\,.\,y\ x$ is an open term, and $\mathit{fix}\ \lambda x\,.\,a\ x$ is a closed term.*

Definition 15 (Signature Σ, context Γ)**.** A *signature*, denoted by Σ, is a partial function mapping from *Cst* to the set $\mathbb{T}$ of simple types. A *context*, denoted by Γ, is a partial function mapping from *Var* to $\mathbb{T}$.

If Σ maps constant c to type τ, we write $c : \tau \in \Sigma$. Moreover we write $\Sigma, c : \tau$ (or $c : \tau, \Sigma$), to denote $\Sigma \cup \{c : \tau\}$. Similar notations apply to Γ.

Definition 16 (Well-typed term)**.** A term M is *well typed* if it satisfies type judgment $\Sigma, \Gamma \vdash M : \tau$ for some Σ, Γ and τ, using the standard typing rules as follows.

$$\frac{c : \tau \in \Sigma}{\Sigma, \Gamma \vdash c : \tau}\ \mathit{con} \qquad \frac{x : \tau \in \Gamma}{\Sigma, \Gamma \vdash x : \tau}\ \mathit{var} \qquad \frac{\Sigma, \Gamma \vdash M : \tau_1 \to \tau_2 \quad \Sigma, \Gamma \vdash N : \tau_1}{\Sigma, \Gamma \vdash (M\ N) : \tau_2}\ \mathit{app}$$

$$\frac{\Sigma, \Gamma, x : \tau_1 \vdash M : \tau_2}{\Sigma, \Gamma \vdash (\lambda x\ .\ M) : \tau_1 \to \tau_2}\ \mathit{abs} \qquad \frac{\Sigma, \Gamma \vdash M : \tau \to \tau}{\Sigma, \Gamma \vdash (\mathit{fix}\ M) : \tau}\ \mathit{fp}$$

We will only work with well typed terms. All *free* variables in a term must have their types assigned by Γ. Therefore, we use $\Sigma, \emptyset \vdash M : \sigma$ to imply that M is a *closed* term.

Definition 17 (Logical\non-logical constants)**.** We use the symbols $\wedge$ for *conjunction*, $\vee$ for *disjunction*, $\supset$ for *implication*, and $\top$ for the *true* proposition. Given some type τ, we use $\forall_\tau$ for *universal quantification*, and $\exists_\tau$ for *existential quantification*, over terms of type τ. These symbols are special constants, called *logical constants* and their types are as follows: $\wedge, \vee, \supset$: $o \to o \to o$, and $\top : o$, and $\forall_\tau, \exists_\tau : (\tau \to o) \to o$. A constant is *non-logical* if it is not a logical constant.

Quantification over a term M is short for applying the quantifier to an abstraction over M, for instance, $\forall_\tau x\ M$ is short for $\forall_\tau\ \lambda x.M$. We assume that a signature Σ always contains at least all logical constants, but we may omit them when we print members of a signature.

Definition 18 (Predicate)**.** A *predicate* is a variable or a non-logical constant of type τ such that either $\tau = o$ or the target type of τ is o.

Definition 19 (Atomic formula: rigid\flexible)**.** An *atomic formula* (in short, *atom*) is a lambda term of type o, in the form $h\ M_1\ \ldots\ M_n$, where $n \geq 0$, h is a predicate, and $M_1, \ldots, M_n$ are lambda terms; if h is a variable, then we say the atom is *flexible*; if h is a non-logical constant, then the atom is *rigid*.

We observe the following syntactic conventions. The outermost parentheses for a term can be omitted. Application associates to the left. Application binds more tightly than abstraction, therefore $\lambda x\,.\,M\ N$ stands for $\lambda x\,.\,(M\ N)$. The constants $\wedge, \vee, \supset$ are used as infix operators with precedence decreasing in the same order therein, and they all bind less tightly than application but more tightly than abstraction. For instance, $\lambda x\ .\ p\ x \supset q\ x$ stands for $\lambda x\ .\ ((p\ x) \supset (q\ x))$. We may combine successive abstraction under one λ, for instance, $\lambda xy\,.\,M$ instead of $\lambda x.\lambda y\,.\,M$. Successive quantification with the same quantifier can be combined under a single quantifier, for instance, we may write $\exists_\iota xy\ M$ instead of $\exists_\iota x \exists_\iota y\ M$.

Definition 20 (First-order\higher-order term)**.** A term $M : \sigma$ is called *first-order* if the following conditions are satisfied.

1. $\mathit{Ord}\,(\sigma) = 0$, and
2. $\{\mathit{Ord}\,(\tau) \mid N : \tau \in \mathit{Sub}\,(M) \wedge N \in \mathit{Cst}\} \subseteq \{0, 1\}$, and
3. $\{\mathit{Ord}\,(\tau) \mid N : \tau \in \mathit{Sub}\,(M) \wedge N \in \mathit{Var}\} \subseteq \{0\}$, and
4. $o \notin \{\tau \mid N : \tau \in \mathit{Sub}\,(M)\}$.

Otherwise a term is of *higher-order.*

Definition 21 (First-order\higher-order predicate)**.** We say that a predicate is *first-order* if it is a non-logical constant, whose type expression is of order at most 1, and the type o does not occur as its argument type. Otherwise a predicate is of *higher-order.*

Definition 22 (First-order\higher-order atom)**.** If h is a first-order predicate, and $M_1, \ldots, M_n$ are first-order terms, then the atom $h\ M_1\ \ldots\ M_n$ is of *first-order.* Otherwise the atom is of *higher-order.*

Definition 23 (First-order signature)**.** A signature Σ is *first-order* if for all non-logical constants b in Σ, (i) the type expression of b is at most of order 1, and (ii) the type o is involved in the type expression of b only if b is a first-order predicate.

Let $M^{x \to y}$ denote the result of replacing every free occurrence of variable x in lambda term M by variable y.

Definition 24 (α*-equivalence* $=_\alpha$, identity $\equiv$)**.** The relation α*-equivalence*, expressed with symbol $=_\alpha$, is defined as: $\lambda x\,.\,M =_\alpha \lambda y\,.\,M^{x \to y}$, provided that y does not occur in M. Moreover, for arbitrary term M, N, L and variable z, we define that $M =_\alpha M$, and that if $M =_\alpha N$, then $N =_\alpha M$, $M\ L =_\alpha N\ L$, $L\ M =_\alpha L\ N$, $\lambda z\,.\,M =_\alpha \lambda z\,.\,N$ and $\mathit{fix}\ M =_\alpha \mathit{fix}\ N$, and that if both $M =_\alpha N$ and $N =_\alpha L$, then $M =_\alpha L$. We use $\equiv$ to denote the relation of *syntactical identity modulo* α*-equivalence.*

Definition 25 (Substitution)**.** *Substitution* of all free occurrences of a variable x in a term M by a term N, denoted $M\,[x := N]$ is defined as standard:

1. $x\,[x := N] \equiv N$
2. $y\,[x := N] \equiv y$, if $y \in \mathit{Var}$ is distinct from x, or $y \in \mathit{Cst}$
3. $(M\ L)\,[x := N] \equiv (M\,[x := N])\ \ (L\,[x := N])$
4. $(\lambda y\,.\,M)\,[x := N] \equiv \lambda z\,.\ (M^{y \to z}\,[x := N])$, if both $\lambda y\,.\,M =_\alpha \lambda z\,.\,M^{y \to z}$ and $z \notin FV\,(N)$
5. $(\mathit{fix}\ M)\,[x := N] \equiv \mathit{fix}\ \ (M\,[x := N])$

Moreover, we call $[x_1 := N_1] \ldots [x_m := N_m]$ a *substitution* when we have

$$M\,[x_1 := N_1] \ldots [x_m := N_m]$$

which denotes $(\ldots(M\,[x_1 := N_1])\ldots)\,[x_m := N_m]$.

Definition 26 ($\to_\beta$, $=_\beta$)**.** *One step β-reduction*, denoted $\to_\beta$, is defined as $(\lambda x\,.\,M)\,N \to_\beta M\,[x := N]$. Moreover, it is defined that if $M \to_\beta N$, then $M\ L \to_\beta N\ L$, $L\ M \to_\beta L\ N$, $\lambda z\,.\,M \to_\beta \lambda z\,.\,N$ and $\mathit{fix}\ M \to_\beta \mathit{fix}\ N$. We define *$\beta$-equivalence*, denoted $=_\beta$, as a transitive and reflexive closure of $\to_\beta$.

If there is no term N such that $M \to_\beta N$, then M is *β-normal.* It well known that β-reduction for simply typed lambda terms without the *fix* primitive are strongly normalizing [17]. Because the *fix* primitive is treated as a constant symbol when doing β-reduction, it does not affect strong normalization of β-reduction. In this paper we assume that all terms are in β-normal form unless otherwise stated.

Definition 27 ($\to_{\mathit{fix}}$)**.** *One step fix-reduction*, denoted $\to_{\mathit{fix}}$, is defined as

$$\mathit{fix}\ M \to_{\mathit{fix}} M\ \ (\mathit{fix}\ M)$$

Moreover, we require that, if $M \to_{\mathit{fix}} N$, then for arbitrary z and L, $M\ L \to_{\mathit{fix}} N\ L$, $L\ M \to_{\mathit{fix}} L\ N$, $\lambda z\,.\,M \to_{\mathit{fix}} \lambda z\,.\,N$ and $\mathit{fix}\ M \to_{\mathit{fix}} \mathit{fix}\ N$.

Definition 28 ($\to_{\mathit{fix}\beta}$)**.** *One step fixβ-reduction*, denoted $\to_{\mathit{fix}\beta}$, is defined as

$$M \to_{\mathit{fix}\beta} N \quad \text{if } M \to_\beta N \text{ or } M \to_{\mathit{fix}} N$$

Definition 29 ($=_{\mathit{fix}\beta}$)**.** *Fixβ-equivalence*, denoted $=_{\mathit{fix}\beta}$, is defined as a transitive and reflexive closure of $\to_{\mathit{fix}\beta}$. In other words, $M =_{\mathit{fix}\beta} N$ if there is $n \geq 0$ and there are terms M_0 to M_n such that $M_0 \equiv M$, $M_n \equiv N$, and for all i such that $0 \leq i < n$, $M_i \to_{\mathit{fix}\beta} M_{i+1}$ or $M_{i+1} \to_{\mathit{fix}\beta} M_i$.

Example 30 (Fixed-point terms and their reductions)**.** *Assume a signature that contains* $0 : \iota$*,* $scons : \iota \to \iota \to \iota$ *(*$scons$ *can be identified with* $[\_|\_]$*) and* $s : \iota \to \iota$*.*

Stream of zeros *We represent the stream of zeros in the form of a (first-order) fixed-point term* $z\_str =_{def} \mathit{fix}\ \lambda x\,.\, scons\ \ 0\ \ x$*. The following relations justify that* z_str *models the stream of zeros:*

$$\begin{aligned} z\_str &=_{def} \mathit{fix}\ \lambda x\,.\, scons\ \ 0\ \ x \\ &=_{\mathit{fix}\beta} scons\ \ 0\ \ z\_str \\ &=_{\mathit{fix}\beta} scons\ \ 0\ \ (scons\ \ 0\ \ z\_str) \\ &\dots \end{aligned}$$

Constant stream *The stream of zeros can be generalized to a stream of any particular number, represented by the (higher-order) fixed-point term:* $n\_str\ =_{def}\ \mathit{fix}\ \lambda f n\,.\, scons\ n\,(f\ \ n)$*. It takes a term* $x : \iota$ *as parameter and returns a stream of* x*'s:*

$$\begin{aligned} n\_str\ x &=_{def} (\mathit{fix}\ \lambda f n\,.\, scons\ n\,(f\ \ n))\ \ x \\ &=_{\mathit{fix}\beta}\ \ scons\ x\,(n\_str\ x) \\ &=_{\mathit{fix}\beta}\ \ scons\ x\,(scons\ x\,(n\_str\ x)) \\ &\dots \end{aligned}$$

Stream of successive numbers *We represent, in the form of a (higher-order) fixed-point term, the family of streams of successive natural numbers:* $fr\_str =_{def} \mathit{fix}\ \lambda f n\ .\ scons\ \ n\ \ (f\ \ (s\ n))$*. Again, the following relations hold for all* $n : \iota$ *(for instance,* n *can be* 0*,* $s\,0$*, ...):*

$$\begin{aligned} fr\_str\ n &=_{def}\ \ \ (\mathit{fix}\ \lambda f n\ .\ scons\ \ n\ \ (f\ \ (s\ n)))\ \ n \\ &=_{\mathit{fix}\beta} scons\ \ n\ \ (fr\_str\,(s\ n)) \\ &=_{\mathit{fix}\beta}\ \ scons\ \ n\ \ \big(scons\ \ (s\ n)\ \ \big(fr\_str\,\big(s^2\ n\big)\big)\big) \\ &=_{\mathit{fix}\beta}\ \ scons\ \ n\ \ \big(scons\ \ (s\ n)\ \ \big(scons\,\big(s^2\ n\big)\ \ \big(fr\_str\,\big(s^3\ n\big)\big)\big)\big) \\ &\dots \end{aligned}$$

Note that fr_str *contains a bound variable of functional type, which is necessary for modeling irregular infinite terms.*

In line with the standard literature on this subject [3, 4], we now need to introduce guardedness conditions on fixed-point definitions, otherwise some of them may not define any infinite objects. A detailed discussion of term productivity or various guardedness conditions that insure this property are beyond the scope of this report. We simply adapt standard guardedness conditions [8], although other methods available in the literature (e.g. [4]) may work as well.

Definition 31 (Guarded fixed-point terms)**.** A guarded fixed-point term has the form

$$\mathit{fix}\ \lambda x \,.\, \lambda y_1 \ldots y_m \,.\, f\ L_1 \ldots L_k\ \ (x\ N_1 \ldots N_m)\ \ L_{k+1} \ldots L_r \qquad \text{where}$$

1. x is a variable of type $\tau_1 \to \cdots \to \tau_m \to \iota$, where $\tau_1 = \ldots = \tau_m = \iota$ and $m \geq 0$.
2. f is a constant of type $\sigma_1 \to \cdots \to \sigma_{r+1} \to \iota$ where $\sigma_1 = \ldots = \sigma_{r+1} = \iota$ and $r \geq 0$.
3. $L_1, \ldots, L_r, N_1, \ldots, N_m$ are first order terms of type ι, but do *not* contain the *fix* primitive.
4. $x \notin \{y_1 : \iota, \ldots, y_m : \iota\} = FV\ (L_1) \cup \cdots \cup FV\ (L_r) \cup FV\ (N_1) \cup \cdots \cup FV\ (N_m)$.

Clearly, the generic guarded fixed-point term in Definition 31 is closed.

Example 32. *All fixed-point terms shown in Example 30 are guarded..*

Definition 33 (Guarded full terms)**.** A guarded full term is

1. a first-order term *not* involving the *fix* primitive, or
2. a (possibly higher-order) term in the form $F\ M_1 \ldots M_n$ where $F : \tau_1 \to \cdots \to \tau_n \to \iota$ is a guarded fixed-point term, and $M_1 : \iota, \ldots, M_n : \iota$ are first-order terms *not* involving the *fix* primitive, provided $\tau_1 = \ldots = \tau_n = \iota$ and $n \geq 0$, or
3. a term that is *fixβ*-equivalent to any term that satisfies 2 above.

Example 34 (Guarded full terms)**.** *Note that fixed-point terms below are from Example 30.*

The following are guarded full terms:

- $(f\ \ x)$*, where* $f : \iota \to \iota \in \Sigma$, $x : \iota \in \Gamma$.
- $\mathit{fix}\ \lambda x \,.\, \mathit{scons}\ \ 0\ \ x$
- $\mathit{scons}\ 0\ (\mathit{fix}\ \lambda x \,.\, \mathit{scons}\ \ 0\ \ x)$
- $(\mathit{fix}\ \lambda f n\ .\ \mathit{scons}\ \ n\ \ (f\ \ (s\ n)))\ \ (s\ 0)$

The following are not *guarded full terms:*

- $(x\ \ a) : \iota$*, where* $a : \iota \in \Sigma$, $x : \iota \to \iota \in \Gamma$.
- $f\ \ (\mathit{fix}\ \lambda x \,.\, \mathit{scons}\ \ 0\ \ x)$*, where* $f : \iota \to \iota \in \Sigma$, $(\mathit{fix}\ \lambda x \,.\, \mathit{scons}\ \ 0\ \ x) : \iota$.
- $\mathit{fix}\ \lambda f n\ .\ \mathit{scons}\ \ n\ \ (f\ \ (s\ n))$

Definition 35 (Guarded Atoms)**.** If h is a first-order predicate, and $M_1, \ldots, M_n$ are guarded full terms, then the atom $A =_{def} h\ M_1\ \ldots\ M_n$ is guarded.

Following the definitions, if A is a guarded atom, then all atoms A' such that $A' =_{\mathit{fix}\beta} A$ are guarded.

Chapter 3

Coinductive Uniform Proofs

We introduce the novel machinery of *coinductive uniform proof* to reveal the proof-theoretic nature of proving a wide range of coinductive invariants arising from first-order Horn clause logic programming.

We start with extending the four abstract logic programming languages by Miller et.al. [16, 15] with fixed-point terms. The resulting languages are named by prefixing *co-* to the corresponding acronyms of names of Miller et. al.'s languages: *co-fohc*, *co-fohh*, *co-hohc* and *co-hohh*. As in the original setting, the distinction between Horn clauses and hereditary Harrop formulae lies in enriched syntax for goals: the latter admits universally quantified and implicative goals; the distinction between first-order and higher-order logic is made by allowing or disallowing higher-order terms, as defined in the previous section.

Definition 36 ($\mathcal{U}_1^\Sigma$, $\mathcal{U}_2^\Sigma$)**.** Assume a signature Σ. Let $\mathcal{U}_1^\Sigma$ be the set of all terms over Σ that do *not* contain the logical constants $\forall$ and $\supset$, and $\mathcal{U}_2^\Sigma$ be the set of all terms over Σ that do *not* contain the logical constant $\supset$. [1]

Definition 37 (D-formulae and G-formulae)**.** Generally, let A and A_r denote the sets of atoms and rigid atoms on Σ, respectively, and let A_1 denote the first-order fraction of A_r. Specifically, in the setting of *co-hohc* (respectively, *co-hohh*), A and A_r are sets of atoms from $\mathcal{U}_1^\Sigma$(respectively, $\mathcal{U}_2^\Sigma$). Table 3.1 defines, for each of our four languages, the set D of *program clauses* and the set G of *goals*.

To help clarify technicalities involved in coinductive sequent proofs, we adapt the standard definition of a sequent in uniform proof [15] by adding one more independent field Δ to the left side of a typical sequent [2]. We will see that Δ collects the coinductive hypothesis involved in a coinductive proof.

[1] Miller and Nadathur called the sets $\mathcal{U}_1^\Sigma$ and $\mathcal{U}_2^\Sigma$ "Herbrand universes" [15, §5.2]. Here we reserve the name "Herbrand" for later modal-theoretic study of the languages.

Definition 38 (Sequent $\Sigma; P; \Delta \longrightarrow G$)**.** Given a signature Σ, a *logic program* P which is a finite set of D-formulae over Σ, and an auxiliary finite set Δ of D-formulae over Σ , a *sequent* is an expression of the form $\Sigma; P; \Delta \longrightarrow G$, encoding the proposition that the goal formula G is intuitionistically provable from $P \cup \Delta$.

Note that in a sequent $\Sigma; P; \Delta \longrightarrow G$, the two sets P and Δ are considered as separate fields, *not* as their union, and this fact is signified by the semicolon in notation $P; \Delta$. In a sequent, two sets P and Q that are considered as their union are delimited by comma, as in P, Q. Also note that in a sequent, to denote a set of formulae, we omit the braces. For example, instead of writing $\Sigma; P \cup \{D_1\}; \{D_2\} \longrightarrow G$, we write $\Sigma; P, D_1; D_2 \longrightarrow G$. Similarly, the expression $c_1 : \tau_1, \ldots, c_n : \tau_n, \Sigma$ is short for $\Sigma \cup \{c_1 : \tau_1, \ldots, c_n : \tau_n\}$.

Definition 39 (Uniform proof rules)**.** We present the *uniform proof* rules in Figure 3.1.

Our uniform proof rules are adapted from Miller et al [15] to support fixed-point terms and the additional field Δ. We now introduce the main rule for coinductive extension of uniform proof. We take inspiration from [20, 5, 8], but make a few adaptations to the uniform proof syntax. We distinguish *coinductive* entailment from entailment by the original uniform rules.

Definition 40 (Sequent $\Sigma; P \looparrowright G$)**.** An expression of the form $\Sigma; P \looparrowright G$, also called a *sequent*, means: the goal formula G is *coinductively* provable from the logic program P on the signature Σ.

Definition 41 (Core formula)**.** Given a language (*co-fohc*, *co-fohh*, *co-hohc* or *co-hohh*), a formula M is a *core formula* of that language if M simultaneously satisfies the syntactic rule for D-formulae and that for G-formulae. The set of core formulae for each of the languages is given in Table 3.2.

Definition 42 (Coinductive fixed-point rule: CO-FIX)**.** The CO-FIX rule for $\looparrowright$ is given in Figure 3.2.

Note that the coinductive invariant M must occur simultaneously as a program clause and as a goal on both sides of the sequent arrow in CO-FIX, therefore, M must be a core formulae. To guard the CO-FIX rule from unsound applications, we introduce the guarding notation $\langle\ \rangle$ within the CO-FIX rule, signifying the guarded status of a formula. A sequent involving guarded formulae is reduced using auxiliary rules.

Definition 43 (Auxiliary rules)**.** *Auxiliary rules* of coinductive uniform proof is given in Figure 3.3.

The auxiliary rules resemble their counterparts in the uniform proof rules. They serve to pass on the guard $\langle\ \rangle$ until the guarded goal is reduced to being atomic, and to ensure that a suitable clause is selected for back chaining. The main difference to notice between the auxiliary rules and their uniform proof counterparts is the restriction imposed on the DECIDE$\langle\rangle$ rule regarding the set from where it can select clauses.

Definition 44 (Safeguard)**.** The combination of the guarding notation $\langle\ \rangle$ and the auxiliary rules constitutes a *safeguard* for the consistency of coinductive uniform proofs.

Appendix A.1 illustrates the inconsistency that can result from a lack of safeguard. The design choice related to the $\supset R\langle\rangle$ rule is discussed in Appendix A.2.

Definition 45 (Coinductive uniform proof rules)**.** The *coinductive uniform proof rules* consist of uniform proof rules in Figures 3.1, coinductive fixed-point rule in Figure 3.2, and auxiliary rules in Figure 3.3.

Definition 46 (Proof in *co-fohc*, *co-fohh*, *co-hohc* or *co-hohh*)**.** A *proof* for a sequent is a finite tree constructed using coinductive uniform proof rules and such that the root is labeled with $\Sigma; P \looparrowright M$ or $\Sigma; P; \Delta \longrightarrow G$, and leaves are labeled with *initial sequents*, i.e. sequents that can occur as a lower sequent in the rule INITIAL. We say that a proof is constructed in *co-fohc* (respectively, *co-fohh*, *co-hohc* or *co-hohh*) if the set of all formulae occurring in it satisfy the syntax of *co-fohc* (respectively, *co-fohh*, *co-hohc* or *co-hohh*).

Discussion

It is not surprising that the complexity of coinductive invariants allowable in coinductive reasoning is proportionate to the expressiveness of the language in which we perform the reasoning. However, the CO-FIX rule adds one more dimension as to why core formulae are of interest: core formulae were studied because they can be stored as proved lemmas and be reused in later proof [16, 10]; now due to the same syntactic character, core formulae constitute the restricted set of formulae that we can coinductively prove using a Löb style coinductive principle. The fact that the CO-FIX rule can only be applied once and as the first step in a proof, is a simplification that helps highlighting the basic proof-theoretic procedure of coinduction. The absence of nested coinduction in a proof can be mitigated by a model extension theorem (Theorem 85) which effectively takes up the role of a coinductive cut rule.

Example 47 (*co-hohh* proof)**.** *The* co-hohh *proof for the sequent*

$$\Sigma; P \looparrowright \forall x(\mathit{from}\ x\ (\mathit{fr\_str}\ x))$$

where program P consists of clause 1.13, is given in Figure 3.4.

	Program Clauses	Goals
co-fohc	$D \ ::= A_1 \mid G \supset D \mid D \wedge D \mid \forall Var\ D$	$G \ ::= A_1 \mid G \wedge G \mid G \vee G \mid \exists Var\ G$
co-hohc	$D \ ::= A_r \mid G \supset D \mid D \wedge D \mid \forall Var\ D$	$G \ ::= A \ \mid G \wedge G \mid G \vee G \mid \exists Var\ G$
co-fohh	$D \ ::= A_1 \mid G \supset D \mid D \wedge D \mid \forall Var\ D$	$G \ ::= A_1 \mid G \wedge G \mid G \vee G \mid \exists Var\ G \mid D \supset G \mid \forall Var\ G$
co-hohh	$D \ ::= A_r \mid G \supset D \mid D \wedge D \mid \forall Var\ D$	$G \ ::= A \ \mid G \wedge G \mid G \vee G \mid \exists Var\ G \mid D \supset G \mid \forall Var\ G$

Table 3.1: D- and G-formulae for abstract logic programming languages extended with fixed-point terms.

co-fohc:	$M := A_1 \mid M \wedge M$	*co-fohh:*	$M := A_1 \mid M \wedge M \mid M \supset M \mid \forall Var\ M$
co-hohc:	$M := A_r \mid M \wedge M$	*co-hohh:*	$M := A_r \mid M \wedge M \mid M \supset M \mid \forall Var\ M$

Table 3.2: The set of core formulae, denoted M, in each language.

Example 48 (*co-fohh* proof)**.** *Consider the program* P : $\{1.\ \forall x, p(f\ x) \wedge q(x) \supset p(x);\quad 2.\ q(a);\quad 3.\ \forall x, q(x) \supset q(f\ x)\}$ *which is a simplied version of the program in Example 3. The coinductive invariant* M *for program* P *is given as* $M =_{def} \forall x(q(x) \supset p(x))$. *We prove* $\Sigma; P \looparrowright M$ *in co-fohh in Figure 3.5.*

$$\frac{\Sigma; P, D; \Delta \longrightarrow G}{\Sigma; P; \Delta \longrightarrow D \supset G} \supset\! R \qquad \frac{c : \tau, \Sigma; P; \Delta \longrightarrow G\,[x := c]}{\Sigma; P; \Delta \longrightarrow \forall_\tau x\ G} \forall R$$

$$\frac{\Sigma; P; \Delta \longrightarrow G\,[x := N]}{\Sigma; P; \Delta \longrightarrow \exists_\tau x\ G} \exists R \qquad \frac{\Sigma; P; \Delta \longrightarrow G_1 \quad \Sigma; P; \Delta \longrightarrow G_2}{\Sigma; P; \Delta \longrightarrow G_1 \wedge G_2} \wedge R$$

$$\frac{\Sigma; P; \Delta \longrightarrow G_1}{\Sigma; P; \Delta \longrightarrow G_1 \vee G_2} \vee R \qquad \frac{\Sigma; P; \Delta \longrightarrow G_2}{\Sigma; P; \Delta \longrightarrow G_1 \vee G_2} \vee R$$

$$\frac{\Sigma; P; \Delta \xrightarrow{D \in P, \Delta} A}{\Sigma; P; \Delta \longrightarrow A} \textsc{decide} \qquad \frac{\Sigma; P; \Delta \xrightarrow{D} A \quad \Sigma; P; \Delta \longrightarrow G}{\Sigma; P; \Delta \xrightarrow{G \supset D} A} \supset\! L$$

$$\frac{\Sigma; P; \Delta \xrightarrow{D_1} A}{\Sigma; P; \Delta \xrightarrow{D_1 \wedge D_2} A} \wedge L \qquad \frac{\Sigma; P; \Delta \xrightarrow{D_2} A}{\Sigma; P; \Delta \xrightarrow{D_1 \wedge D_2} A} \wedge L$$

$$\frac{}{\Sigma; P; \Delta \xrightarrow{A'} A} \textsc{initial} \qquad \frac{\Sigma; P; \Delta \xrightarrow{D[x := N]} A}{\Sigma; P; \Delta \xrightarrow{\forall_\tau x\ D} A} \forall L$$

Figure 3.1: Uniform proof rules. *Restrictions:* In $\exists R$ and $\forall L$, $\Sigma; \emptyset \vdash N : \tau$, and if used in *co-fohc* or *co-fohh*, then N is first-order; if used in *co-hohc*, then $N \in \mathcal{U}_1^\Sigma$; if used in *co-hohh*, then $N \in \mathcal{U}_2^\Sigma$. In $\forall R$, $c : \tau \notin \Sigma$ (c is also known as an *eigenvariable*). In DECIDE, $D \in P \cup \Delta$. In the rule INITIAL , $A =_{\mathit{fix}\beta} A'$.

$$\frac{\Sigma; P; M \longrightarrow \langle M \rangle}{\Sigma; P \looparrowright M} \textsc{co-fix}$$

Figure 3.2: Coinductive fixed point rule. M is a core formula. In the upper sequent of CO-FIX rule, the left occurrence of M is called a *coinductive hypothesis*, and the right occurrence of M marked by $\langle\rangle$ is called a *coinductive goal.*

$$\frac{c : \tau, \Sigma; P; \Delta \longrightarrow \langle M\,[x := c] \rangle}{\Sigma; P; \Delta \longrightarrow \langle \forall_\tau x\ M \rangle} \forall R\langle\rangle \qquad \frac{\Sigma; P, M_1; \Delta \longrightarrow \langle M_2 \rangle}{\Sigma; P; \Delta \longrightarrow \langle M_1 \supset M_2 \rangle} \supset\! R\langle\rangle$$

$$\frac{\Sigma; P; \Delta \longrightarrow \langle M_1 \rangle \quad \Sigma; P; \Delta \longrightarrow \langle M_2 \rangle}{\Sigma; P; \Delta \longrightarrow \langle M_1 \wedge M_2 \rangle} \wedge R\langle\rangle \qquad \frac{\Sigma; P; \Delta \xrightarrow{D \in P} A}{\Sigma; P; \Delta \longrightarrow \langle A \rangle} \textsc{decide}\langle\rangle$$

Figure 3.3: Rules for guarded coinductive goals. *Restrictions*: In DECIDE$\langle\rangle$, $D \in P$ but *not* $D \in \Delta$. The restriction for $\forall R\langle\rangle$ is the same as for $\forall R$.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{\textsc{initial}✓}
\UnaryInfC{$c, \Sigma; P; M \overset{\mathit{from}\ c\ (\mathit{scons}\ c\ (\mathit{fr\_str}\ (s\ c)))}{\longrightarrow} \mathit{from}\ c\ (\mathit{fr\_str}\ c)$}
\AxiomC{}
\RightLabel{\textsc{initial}}
\UnaryInfC{$c, \Sigma; P; M \overset{\mathit{from}\ (s\ c)\ (\mathit{fr\_str}\ (s\ c))}{\longrightarrow} \mathit{from}\ (s\ c)\ (\mathit{fr\_str}\ (s\ c))$}
\RightLabel{$\forall L$}
\UnaryInfC{$c, \Sigma; P; M \xrightarrow{M} \mathit{from}\ (s\ c)\ (\mathit{fr\_str}\ (s\ c))$}
\RightLabel{\textsc{decide}}
\UnaryInfC{$c, \Sigma; P; M \longrightarrow \mathit{from}\ (s\ c)\ (\mathit{fr\_str}\ (s\ c))$}
\RightLabel{$\supset L$}
\BinaryInfC{$c, \Sigma; P; M \overset{\mathit{from}\ (s\ c)(\mathit{fr\_str}\ (s\ c))\ \supset \mathit{from}\ c\ (\mathit{scons}\ c\ (\mathit{fr\_str}\ (s\ c)))}{\longrightarrow} \mathit{from}\ c\ (\mathit{fr\_str}\ c)$}
\RightLabel{$\forall L\,(2 \text{ times})$}
\UnaryInfC{$c, \Sigma; P; M \overset{\forall xy\ \mathit{from}\ (s\ x)\ y \supset \mathit{from}\ x\ (\mathit{scons}\ x\ y)}{\longrightarrow} \mathit{from}\ c\ (\mathit{fr\_str}\ c)$}
\RightLabel{\textsc{decide}$\langle\rangle$}
\UnaryInfC{$c, \Sigma; P; M \longrightarrow \langle \mathit{from}\ c\ (\mathit{fr\_str}\ c) \rangle$}
\RightLabel{$\forall R\langle\rangle$}
\UnaryInfC{$\Sigma; P; M \longrightarrow \langle \forall x (\mathit{from}\ x\ (\mathit{fr\_str}\ x)) \rangle$}
\RightLabel{\textsc{co-fix}}
\UnaryInfC{$\Sigma; P \looparrowright \forall x (\mathit{from}\ x\ (\mathit{fr\_str}\ x))$}
\end{prooftree}
$$

Figure 3.4: Note that fr_str is defined in Example 30, c is an arbitrary eigenvariable, M abbreviates the coinductive hypothesis $\forall x(\mathit{from}\ x\ (\mathit{fr\_str}\ x))$, and the step marked by ✓ indicates involvement of the relation $\mathit{from}\ c\ (\mathit{scons}\ c\ (fr\_str\ (s\ c))) =_{\mathit{fix}\beta} \mathit{from}\ c\ (\mathit{fr\_str}\ c)$. The first two $\forall L$ steps involve the substitutions $x := c, y := (fr\_str\ (s\ c))$. The last $\forall L$ step involves the substitution $x := s\ c$.

(1)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c,\Sigma;P,q(c);M \xrightarrow{p(c)} p(c)}\,\textsc{initial} \qquad \dfrac{\cdots}{c,\Sigma;P,q(c);M \longrightarrow p(f\ c)\wedge q(c)}\,\wedge R}{c,\Sigma;P,q(c);M \xrightarrow{p(f\ c)\wedge q(c)\supset p(c)} p(c)}\,\supset L}{c,\Sigma;P,q(c);M \xrightarrow{P(1)} p(c)}\,\forall L}{c,\Sigma;P,q(c);M \longrightarrow \langle p(c)\rangle}\,\textsc{decide}\langle\rangle}{c,\Sigma;P;M \longrightarrow \langle q(c)\supset p(c)\rangle}\,\supset R\langle\rangle}{\Sigma;P;M \longrightarrow \langle\forall x(q(x)\supset p(x))\rangle}\,\forall R\langle\rangle}{\Sigma;P \looparrowright \forall x(q(x)\supset p(x))}\,\textsc{co-fix}$$

(2)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c,\Sigma;P,q(c);M \xrightarrow{p(f\ c)} p(f\ c)}\,\textsc{initial} \qquad \dfrac{\dfrac{\dfrac{\cdots}{c,\Sigma;P,q(c);M \xrightarrow{q(c)\supset q(f\ c)} q(f\ c)}\,\supset L}{c,\Sigma;P,q(c);M \xrightarrow{P(3)} q(f\ c)}\,\forall L}{c,\Sigma;P,q(c);M \longrightarrow q(f\ c)}\,\textsc{decide}}{c,\Sigma;P,q(c);M \xrightarrow{q(f\ c)\supset p(f\ c)} p(f\ c)}\,\supset L}{c,\Sigma;P,q(c);M \xrightarrow{M} p(f\ c)}\,\forall L}{c,\Sigma;P,q(c);M \longrightarrow p(f\ c)}\,\textsc{decide} \qquad \spadesuit}{c,\Sigma;P,q(c);M \longrightarrow p(f\ c)\wedge q(c)}\,\wedge R$$

(3)

$$\dfrac{\dfrac{}{c,\Sigma;P,q(c);M \xrightarrow{q(f\ c)} q(f\ c)}\,\textsc{initial} \qquad \dfrac{\dfrac{}{c,\Sigma;P,q(c);M \xrightarrow{q(c)} q(c)}\,\textsc{initial}}{c,\Sigma;P,q(c);M \longrightarrow q(c)}\,\textsc{decide}}{c,\Sigma;P,q(c);M \xrightarrow{q(c)\supset q(f\ c)} q(f\ c)}\,\supset L$$

Figure 3.5: An example of *co-fohh* proof. The sub-proof denoted by ♠ in pane (2) is a copy of the right branch of the sub-proof in pane (3).

Chapter 4

Soundness of Coinductive Uniform Proof

The ability of *co-hohh* to capture a wide range of coinductive invariants arising in first-order Horn clause logic programming, is formalized as a soundness theorem (Theorem 81).

Definition 49 (H-formula)**.** By *H-formula*, we mean a closed formula of the form $\forall_\iota x_1 \dots x_m \quad A_1 \wedge \dots \wedge A_n \supset A \quad (m, n \geq 0)$ where $A, A_1, \dots, A_n$ are guarded atoms.

We study soundness of *co-hohh* proof of a generic sequent in the form $\Sigma; P \looparrowright M$ where Σ is a first-order signature, P is a set of H-formulae and M is an H-formula[1].

A logic program is now a set of *any* H-formulae, rather than, as in the standard setting [14], a set of H-formulae not involving *fix* primitive. This generalization allows us to discuss models of programs that are standard first-order Horn clause programs augmented with coinductively proved lemmas that are H-formulae. As a result, our definitions of immediate consequence operators are *not* standard, and in turn our definitions of coinductive models are *not* standard. More precisely, in those aspects our definitions generalize the standard ones because an H-formula generalizes a standard first-order Horn clause. Since our sets of H-formulae are based on first-order signatures, our definitions of (complete) Herbrand universe and (complete) Herbrand base are still standard.

By definition, a proof in any of the three languages *is* a proof in *co-hohh*. In consequence, the soundness theorem for the *co-hohh* calculus subsumes soundness theorems for the rest calculi. Therefore, we will focus on soundness result for *co-hohh* without loss of generality.

[1] Note that the syntax of *co-hohh* core formulae includes all H-formulae.

4.1 Coinductive Models of Logic Programs

We establish in Section 4.1.2 a connection between guarded atoms and (possibly infinite) tree-atoms that inhabit the complete Herbrand base of a logic program. Then we define coinductive models for sets of H-formulae. We put standard and non-standard model-theoretic definitions in Sections 4.1.1 and 4.1.3 respectively.

4.1.1 Standard model-theoretic definitions

Definition 50 (Set of finite list, ω^*)**.** We write ω for the set of all non-negative integers, and write ω^* for the set of all finite lists of non-negative integers. Lists are denoted by $[i, \dots, j]$ where $i, \dots, j \in \omega$. The empty list is denoted ϵ. If $w, v \in \omega^*$, then $[w, v]$ denotes the list which is the concatenation of w and v. If $w \in \omega^*$ and $i \in \omega$, then $[w, i]$ denotes the list $[w, [i]]$.

Definition 51 (Tree Language)**.** A set $L \subseteq \omega^*$ is a *(finitely branching) tree language* provided: i) for all $w \in \omega^*$ and all $i, j \in \omega$, if $[w, j] \in L$ then $w \in L$ and, for all $i < j$, $[w, i] \in L$; and ii) for all $w \in L$, the set of all $i \in \omega$ such that $[w, i] \in L$ is finite. A non-empty tree language always contains ϵ, which we call its *root*.

Definition 52 (Finite\infinite tree language)**.** A tree language is *finite* if it is a finite subset of ω^*, and *infinite* otherwise.

Definition 53 (Depth)**.** We use $|w|$ to denote the length of list w, called the *depth* of the node $w \in L$ if L is a tree language.

Given a first-order signature Σ, the type of a non-logical constant c in Σ can be depicted as $\iota \to \dots \to \iota \to \tau$ where τ is either ι or o.

Definition 54 (Arity)**.** Given a first-order signature Σ, the *arity* of $c : \sigma \in \Sigma$ is defined as the number of occurrences of $\to$ in σ. A variable of type ι has arity 0.

Definition 55 (Tree-Terms, Tree-Atoms)**.** If L is a non-empty tree language, Σ is a first-order signature, and Γ is a context that assigns the type ι to variables, then a *tree-term* over Σ is a function $t : L \to \Sigma \cup \Gamma$ such that, i) for all $w \in L$, $t\,(w)$ is either a variable in Γ, or a non-logical constant in Σ which is not a predicate, and ii) the arity of $t\,(w)$ equals to the cardinality of the set $\{i \in \mathbb{N} \mid [w, i] \in L\}$. If, for $t(w)$, $\{i \in \mathbb{N} \mid [w, i] \in L\} = \emptyset$, we say $t(w)$ is a *leaf* of the tree-term t. A *tree-atom* over Σ is defined in the same way as a tree-term except that the root of L is mapped to a predicate in Σ.

We use s, t to denote arbitrary tree-terms (tree-atoms).

Definition 56 (Open\closed\finite \infinite Tree-term\atom)**.** The domain of a tree-term (tree-atom) t is denoted $Dom\,(t)$. A tree-term (tree-atom) t is *closed* if no $w \in Dom\,(t)$ is mapped to a variable, otherwise t is *open*. Tree-terms (tree-atom) are finite (respectively, infinite) if their domains are finite (respectively, infinite).

Example 57 (Tree-term)**.** *Let* Σ *contain non-logical constants* $0 : \iota$*,* $nil : \iota$ *and* $scons : \iota \to \iota \to \iota$*. Let* Γ_1 *contain* $x : \iota$*.*

1. *Let* $L = \{\epsilon, [0]\,, [1]\}$*. A finite closed tree-term is defined by the mapping* $t_1 : L \to \Sigma \cup \Gamma_1$*, such that* $t_1\,(\epsilon) = scons$*,* $t_1\,([0]) = 0$*,* $t_1\,([1]) = nil$*.*

2. *Let* $W_1 ::= \epsilon \mid [W_1, 1]$*, denoting the set of all finite (possibly empty) lists of 1's. Let* L' *be the smallest tree language containing* W_1*, which amounts to the union of* W_1 *and* W_2*, where* $W_2 ::= [W_1, 0]$*. An infinite open tree-term is defined by the mapping* $t_2 : L' \to \Sigma \cup \Gamma_1$*, such that for all* $w \in W_1$*,* $t_2\,(w) = scons$*, and for all* $v \in W_2$*,* $t_2\,(v) = x$ *.*

Definition 58 (Substitution for tree-term\atom)**.** *Substitution* of a tree-term s for all occurrences of a variable x in a tree-term (or tree-atom) t, denoted $t\,[x := s]$, is defined as follows: let t' be the result of the substitution, then i) $Dom\,(t')$ is the union of $Dom\,(t)$ with the set $\{[w, v] \mid w \in Dom\,(t)\,, v \in Dom\,(s)\,, t\,(w) = x\}$, and ii) $t'\,(w) = t\,(w)$ if $w \in Dom\,(t)$ and $t\,(w) \neq x$; $t'\,([w, v]) = s\,(v)$ if $w \in Dom\,(t)\,, v \in Dom\,(s)\,,$ and $t\,(w) = x$.

Definition 59 (Tree-Term (Tree-Atom) Metric d)**.** Given a term t on Σ where $\star : \iota \notin \Sigma$, the *truncation* of a tree-term (or tree-atom) t at depth $n \in \omega$, denoted by $\gamma\,'(n, t)$, is constructed as follows:
(a) the domain $Dom\,(\gamma\,'(n, t))$ of the term $\gamma'(n, t)$ is $\{m \in Dom(t) \mid |m| \leq n\}$;
(b)

$$\gamma\,'(n, t)\ (m) = \begin{cases} t(m) & \text{if } |m| < n \\ \star & \text{if } |m| = n \end{cases}$$

For tree-terms (or tree atoms) t, s, we define $\gamma(s, t) = min\{n \mid \gamma\,'(n, s) \neq \gamma\,'(n, t)\}$, so that $\gamma(s, t)$ is the least depth at which t and s differ. We define $d(s, t) = 0$ if $s = t$ and $d(s, t) = 2^{-\gamma(s,t)}$ otherwise.

The set of tree-terms and atoms on first-order signature Σ equipped with metric d is an ultrametric space [14].

Definition 60 (complete Herbrand universe, $\mathcal{H}^{\Sigma}$)**.** Given a first order signature Σ and a logic program P_{Σ}, the *complete Herbrand universe* of P_{Σ}, denoted $\mathcal{H}^{\Sigma}$, is the set of all finite and infinite closed tree-terms on Σ.

Definition 61 (complete Herbrand base, $\mathcal{B}^{\Sigma}$)**.** Given a first order signature Σ and a logic program P_{Σ}, the *complete Herbrand base* of P_{Σ}, denoted $\mathcal{B}^{\Sigma}$, is the set of all finite and infinite closed tree-atoms on Σ.

Definition 62 (Herbrand universe, $\mathcal{H}^{\Sigma}_{-}$)**.** Given a first order signature Σ and a logic program P_{Σ}, the *Herbrand universe* of P_{Σ}, denoted $\mathcal{H}^{\Sigma}_{-}$, is the set of all finite closed tree-terms on Σ.

Definition 63 (Herbrand base, $\mathcal{B}^{\Sigma}_{-}$)**.** Given a first order signature Σ and a logic program P_{Σ}, the *Herbrand base* of P_{Σ}, denoted $\mathcal{B}^{\Sigma}_{-}$, is the set of all finite closed tree-atoms on Σ.

Definition 64 (Complete Herbrand interpretation, I)**.** A *complete Herbrand interpretation*, denoted I, is any subset of $\mathcal{B}^{\Sigma}$.

Definition 65 (Herbrand interpretation, I_{-})**.** A *Herbrand interpretation*, denoted I_{-}, is any subset of $\mathcal{B}^{\Sigma}_{-}$.

Definition 66 (*glb,lub*,complete lattice)**.** Let set S be a set equipped with a partial order $\leq$, denoted $\langle S, \leq \rangle$, and let $S' \subseteq S$, $a \in S$ and $b \in S$. If for all $x \in S'$, $a \leq x$, then a is called a *lower bound* of S'. If for all $x \in S'$, $x \leq b$, then b is called a *upper bound* of S'. Further, let a, b be a lower bound and a upper bound of S', respectively, then, if for all lower bounds a' of S', $a' \leq a$, then a is the *greatest lower bound* of S', denoted $glb\,(S')$, and, if for all upper bounds b' of S', $b \leq b'$, then b is the *least upper bound* of S', denoted $lub\,(S')$. A partially ordered set $\langle S, \leq \rangle$ is a *complete lattice* if for all subset $S' \subseteq S$, there exist $glb\,(S')$ and $lub\,(S')$.

Example 67. *Given a set S, its power set is denoted $Pow\,(S)$. Then $\langle Pow\,(S), \subseteq \rangle$ is a complete lattice, as for each subset X of $Pow\,(S)$, $glb\,(X)$ is given by $\bigcap X$, and $lub\,(X)$ is given by $\bigcup X$.*

Definition 68 (Increasing; pre-\post-\least (*lfp*)\greatest (*gfp*) fixed points)**.** Given a complete lattice $\langle L, \leq \rangle$, a function $f : L \mapsto L$ is *increasing* if $f\,(x) \leq f\,(y)$ whenever $x \leq y$. Moreover, given an $x \in L$, x is a *fixed point* of f if $x = f\,(x)$; x is a *pre-fixed point* of f if $f\,(x) \leq x$; x is a *post-fixed point* of f if $x \leq f\,(x)$. We call $a \in L$ the *least fixed point* of f, denoted *lfp* (f), if a is a fixed point of f, and for all fixed points a' of f, $a \leq a'$. The *greatest fixed point* of f, denoted *gfp* (f), is defined similarly.

Theorem (Knaster-Tarski). *Given a complete lattice $\langle L, \leq \rangle$ and an increasing function $f : L \mapsto L$,*

$$\begin{aligned} \mathit{lfp}\,(f) &= \mathit{glb}\{x \mid x = f\,(x)\} = \mathit{glb}\{x \mid f\,(x) \leq x\} \\ \mathit{gfp}\,(f) &= \mathit{lub}\{x \mid x = f\,(x)\} = \mathit{lub}\{x \mid x \leq f\,(x)\} \end{aligned}$$

4.1.2 Productivity of guarded atoms

The connection between guarded atoms on Σ and the complete Herbrand base of P_{Σ} is a form of productivity result for guarded fixed-point terms. If A is a first-order atom (without involving *fix*), then we will denote the equivalent tree-atom by A^T.

Definition 69 (Truncation of guarded atom, $A\diamond$)**.** Let A be a guarded atom on Σ and let $\diamond : \iota \notin \Sigma$, then the *truncation* of A, denoted $A\diamond$, is the atom obtained by replacing sub-terms in A with $\diamond$, in such a way that a sub-term N in A is replaced by $\diamond$ if and only if N is a guarded full term that involves a fixed-point term.

Clearly, $A\diamond$ is a first-order atom.

Example 70. *The truncation of guarded atom*

$$p\,(scons\; 0\;\; (scons\; 0\; (\mathit{fix}\; \lambda x\,.\, scons\;\; 0\;\; x)))$$

is

$$p\,(scons\; 0\;\; (scons\; 0\; \diamond))$$

Definition 71 (fairness of *fixβ*-reduction)**.** A guarded atom has a *fair* infinite sequence of *fixβ*-reductions if every *fixβ*-reducible sub-term is reduced within a finite number of steps.

Lemma 72 (Productivity lemma)**.** *Let A be a guarded atom on a first-order signature Σ and $\diamond : \iota \notin \Sigma$. If A has a fair infinite sequence of one step fixβ-reductions $A \rightarrow_{\mathit{fix}\beta} A'_1,\; A_1 \rightarrow_{\mathit{fix}\beta} A'_2,\; A_2 \rightarrow_{\mathit{fix}\beta} A'_3, \ldots$, where A_k is the β-normal form of A'_k, then there exist an infinite tree-atom A^T such that $d((A_k\diamond)^T, A^T) \rightarrow 0$ as $k \rightarrow \infty$. Moreover, if A is a closed guarded atom, then $A^T \in \mathcal{B}^\Sigma$.*

Proof. Similar to the proof in Komendantskaya and Li [11, Lemma 4.1]. □

Lemma 72 allows us to extend the notation A^T, from requiring A be a first-order atom without *fix* primitive, to allowing A to be any guarded atom, and shows that $A^T \in \mathcal{B}^\Sigma$ if A is closed.

Corollary 73 allows us to use the notation M^T to denote the (infinite) tree-term represented by a guarded full term M.

Corollary 73 (Productivity of guarded full terms)**.** *Let term M be on signature Σ and* involve *the fix primitive. If M is a guarded full term, then there exist an equivalent infinite tree-term M^T. Further, if M is closed, then $M^T \in \mathcal{H}^\Sigma$.*

Corollary 74 is about tree sharing among *fixβ*-equivalent guarded atoms.

Corollary 74 (Tree sharing)**.** *If $A_1, \ldots, A_n$ are guarded atoms and they are pairwise fixβ-equivalent, then $A_1^T = \ldots = A_n^T$.*

4.1.3 Non-standard model-theoretic definitions

We now proceed to consider coinductive models of logic programs. It is known that $\langle Pow\left(\mathcal{B}^{\Sigma}\right), \subseteq\rangle$ and $\langle Pow\left(\mathcal{B}^{\Sigma}_{-}\right), \subseteq\rangle$ are complete lattices (cf. Example 67). Next we define increasing functions on these lattices, known as *immediate consequence operators*. Then coinductive Herbrand models will be defined as greatest fixed-points of these increasing functions.

Definition 75 (Tree-form\term-form ground instance)**.** Let K be an H-formula

$$K =_{def} \forall_\iota x_1 \ldots x_m \; A_1 \wedge \ldots \wedge A_n \supset A$$

- A *tree-form ground instance* $\lfloor K \rfloor^T$ of K is

$$\left(A_1^T \wedge \ldots \wedge A_n^T \supset A^T\right)[x_1 := N_1] \cdots [x_m := N_m]$$

  where $N \in \mathcal{H}^{\Sigma}$.

- A *term-form ground instance* $\lfloor K \rfloor$ of K is

$$(A_1 \wedge \ldots \wedge A_n \supset A)\left[x_1 := N'_1\right] \cdots \left[x_m := N'_m\right]$$

  where N' is a closed guarded full term in $\mathcal{U}^{\Sigma}_1$ and all resulting atoms are guarded.

Definition 76 (*head* K, *body* K)**.** If $K =_{def} A'_1 \wedge \ldots \wedge A'_n \supset A'$ is a ground instance (in either tree-form or term-form), then we denote A' by *head* K and denote the set $\{A'_1, \ldots, A'_n\}$ by *body* K.

Definition 77 (Immediate consequence operators $\mathcal{T}$, $\mathcal{T}_{-}$)**.** Let Σ be first-order and P_Σ be a set of H-formulae, equipped with $\mathcal{B}^{\Sigma}$ and $\mathcal{B}^{\Sigma}_{-}$.

- $\mathcal{T} : Pow\left(\mathcal{B}^{\Sigma}\right) \mapsto Pow\left(\mathcal{B}^{\Sigma}\right)$, is defined as

$$\mathcal{T}(I) = \{B \in \mathcal{B}^{\Sigma} \mid F \in P_\Sigma,\ \textit{head}\ \lfloor F \rfloor^T = B,\ \textit{body}\ \lfloor F \rfloor^T \subseteq I\ \}$$

- $\mathcal{T}_{-} : Pow\left(\mathcal{B}^{\Sigma}_{-}\right) \mapsto Pow\left(\mathcal{B}^{\Sigma}_{-}\right)$, is defined as

$$\mathcal{T}_{-}(I_{-}) = \{B \in \mathcal{B}^{\Sigma}_{-} \mid F \in P_\Sigma,\ \textit{head}\ \lfloor F \rfloor^T = B,\ \textit{body}\ \lfloor F \rfloor^T \subseteq I_{-}\ \}$$

Using the fact that $\mathcal{T}$ and $\mathcal{T}_{-}$ are increasing, we can rely on the Knaster-Tarski theorem to assert that their respective greatest fixed points $\mathcal{M}$ and $\mathcal{M}_{-}$ exist.

Definition 78 (Coinductive models $\mathcal{M}$ and $\mathcal{M}_{-}$)**.** Given a first-order signature Σ and a set P_Σ of H-formulae, equipped with $\mathcal{T}$ and $\mathcal{T}_{-}$, let

$$\mathcal{M} = \textit{gfp}\,(\mathcal{T}) = \bigcup\{I \mid I = \mathcal{T}(I)\} = \bigcup\{I \mid I \subseteq \mathcal{T}(I)\}$$
$$\mathcal{M}_{-} = \textit{gfp}\,(\mathcal{T}_{-}) = \bigcup\{I_{-} \mid I_{-} = \mathcal{T}_{-}(I_{-})\} = \bigcup\{I_{-} \mid I_{-} \subseteq \mathcal{T}_{-}(I_{-})\}$$

$\mathcal{M}_{-}$ is the *finite-term coinductive model* and $\mathcal{M}$ is the *finite-and-infinite-term coinductive model* of P_Σ.

The implication from right to left in Lemma 79 is an instance of the *coinductive proof principle*, as formulated e.g. in Sangiorgi [19, §2.4]. We additionally show that the reverse is also true. Lemma 79 will be used in both directions in later proofs.

Lemma 79 (Coinductive proof principle)**.** *Let P_Σ be a set of H-formulae on first-order signature Σ, with the operator $\mathcal{T}$ and the model $\mathcal{M}$. Given a set S, $S \subseteq \mathcal{M}$, if and only if, there exist a complete Herbrand interpretation I for P_Σ, such that $S \subseteq I$ and I is a post-fixed point of $\mathcal{T}$. Concerning $\mathcal{T}_-$, $\mathcal{M}_-$ and I_-, a similar statement can be made.*

Proof. We first prove the implication from right to left. This is justified by the definition of $\mathcal{M}$.

Then we prove from left to right. $S \subseteq \mathcal{M}$ implies that for all $x \in S$, $x \in \mathcal{M}$. Then, by the definition of $\mathcal{M}$, for all $x \in S$, there exist an I_x, such that $x \in I_x$ and $I_x \subseteq \mathcal{T}(I_x)$. Then let $I = \bigcup\{I_x \mid x \in S\}$. By the construction of I, we have $S \subseteq I$. Also we have $I_x \subseteq I$, for all I_x. Since $\mathcal{T}$ is increasing, $\mathcal{T}(I_x) \subseteq \mathcal{T}(I)$, for all I_x. By transitivity of $\subseteq$, $I_x \subseteq \mathcal{T}(I)$, for all I_x. So $I \subseteq \mathcal{T}(I)$, by the construction of I .

The proof concerning $\mathcal{T}_-$, $\mathcal{M}_-$ and I_- is similar. □

4.2 Soundness Theorems and Corollaries

We are now ready to formulate soundness of coinductive uniform proofs.

Definition 80 ($\mathcal{M} \vDash H$)**.** We use $\mathcal{M} \vDash H$ to denote that for all tree-form ground instances $\lfloor H \rfloor^T$ of H, *head* $\lfloor H \rfloor^T \in \mathcal{M}$ whenever *body* $\lfloor H \rfloor^T \subseteq \mathcal{M}$. As special cases, $\mathcal{M} \vDash \forall \bar{x}\, A$ denotes that for all tree-form ground instances $\lfloor \forall \bar{x}\, A \rfloor^T$ of $\forall \bar{x}\, A$, $\lfloor \forall \bar{x}\, A \rfloor^T \in \mathcal{M}$, and $\mathcal{M} \vDash A$ denotes that $A^T \in \mathcal{M}$.

We use bullet points (•) to highlight the fact that the conditions of Theorem 81 are part of the conditions of Theorem 85, which in turn are part of the conditions of Theorem 88. Corollaries 89 and 90 also respectively reuse conditions of Corollaries 86 and 87, and the reused conditions are highlighted by •.

Theorem 81 (Soundness of *co-hohh* proofs)**.**

- *Let Σ be a first-order signature, P be a set of H-formulae, and H be an H-formula. Let $\Sigma; P \looparrowright H$ have a proof in co-hohh which involves only guarded atoms, and $\mathcal{M}$ be the finite-and-infinite-term coinductive model of P.*

Then, $\mathcal{M} \vDash H$.

Proof Sketch. Assume H has the form $\forall_\iota x_1 \dots x_m\ A_1 \wedge \dots \wedge A_n \supset A$, whose arbitrary tree-form ground instance is denoted $A'_1 \wedge \dots \wedge A'_n \supset A'$. By definition of $\mathcal{M} \vDash H$, we need to show $\{A'_1, \dots A'_n\} \subseteq \mathcal{M}$ implies $\{A'\} \subseteq \mathcal{M}$.

We use Lemma 79 from right to left, which means, to show that $\{A'\} \subseteq \mathcal{M}$, we look for a set I such that the *requirements* $\{A'\} \subseteq I$ and $I \subseteq \mathcal{T}(I)$ are satisfied. The proof follows an *Analysis–Construction–Verification* structure, where we first study the proof of the *root sequent* $\Sigma; P \looparrowright \forall_\iota x_1 \ldots x_m \; A_1 \wedge \ldots \wedge A_n \supset A$ which provides information for constructing a candidate set I. In this construction, we use Lemma 72. Finally we verify that the set I so constructed satisfies the *requirements.* The exact details of these three steps are given in Chapter 5. □

Corollary 82 (Soundness of *co-hohc* proofs). *Let Σ be a first-order signature, P be a set of H-formulae with the finite-and-infinite-term coinductive model $\mathcal{M}$, A be an atomic H-formula, and $\Sigma; P \looparrowright A$ have a proof in co-hohc which involves only guarded atoms. Then, $\mathcal{M} \vDash A$.*

Corollary 83 (Soundness of *co-fohh* proofs). *Let Σ be a first-order signature, P be a set of H-formulae with coinductive models $\mathcal{M}$ and $\mathcal{M}_-$, H be an H-formula, and $\Sigma; P \looparrowright H$ have a proof in co-fohh which involves only guarded atoms. Then, $\mathcal{M} \vDash H$. Moreover, if the co-fohh proof does not involve the fix primitive, then $\mathcal{M}_- \vDash H$.*

Corollary 84 (Soundness of *co-fohc* proofs). *Let Σ be a first-order signature, P be a set of H-formulae with coinductive models $\mathcal{M}$ and $\mathcal{M}_-$, A be a first-order atomic H-formula, and $\Sigma; P \looparrowright A$ have a proof in co-fohc which involves only guarded atoms. Then, $\mathcal{M} \vDash A$. Moreover, if the co-fohc proof does not involve the fix primitive, then $\mathcal{M}_- \vDash A$.*

Next, we show that extending logic programs with coinductively proven lemmas is sound.

Theorem 85 (Model-conservative program extension, I).

- *Let Σ be a first-order signature, P be a set of H-formulae, and H be an H-formula. Let $\Sigma; P \looparrowright H$ have a proof in co-hohh which involves only guarded atoms, and $\mathcal{M}$ be the finite-and-infinite-term coinductive model of P.*
- *Let $H_1, \ldots, H_n$ be distinct term-form ground instances of H, and such that for each H_k ($1 \leq k \leq n$), if $A \in body\, H_k$, then $A^T \in \mathcal{M}$. Let $P \cup \{H_1, \ldots, H_n\}$ have the finite-and-infinite-term coinductive model $\mathcal{M}'$.*

Then, $\mathcal{M} = \mathcal{M}'$.

Proof. The proof shows equality of the two sets by proving that one is a subset of the other, and vice versa. Theorem 81 and Lemma 79 are used.

We first show $\mathcal{M} \subseteq \mathcal{M}'$. Let $\mathcal{T}_P$ and $\mathcal{T}_P^+$ denote immediate consequence operators for P and $P \cup \{H_1, \ldots, H_n\}$, respectively. By definition of a coinductive model, we have

$$\mathcal{M} = \bigcup\{I \mid I \subseteq \mathcal{T}_P(I)\} \qquad \mathcal{M}' = \bigcup\{I \mid I \subseteq \mathcal{T}_P^+(I)\}$$

Note that

$$\{I \mid I \subseteq \mathcal{T}_P(I)\} \subseteq \{I \mid I \subseteq \mathcal{T}_P^+(I)\}$$

therefore $\mathcal{M} \subseteq \mathcal{M}'$.

Then we show $\mathcal{M}' \subseteq \mathcal{M}$. Given a set I that is a post-fixed point of $\mathcal{T}_P^+$, we distinguish two cases concerning how I is a post-fixed point, as follows.

1. For all $x \in I$, there exist $F \in P$, with tree-form ground instance F', such that x is the head of F', and the body of F' is a subset of I. In this case, I is also a post-fixed point of $\mathcal{T}_P$.

2. There exist $x_1, \ldots, x_m \in I$ ($1 \leq m \leq n$, where the inequality $m \leq n$ is explained later), called *outstanding members* of I, such that for all x_i ($1 \leq i \leq m$)

   (a) there exist some H_k, whose head is denoted B and whose body is denoted $\{B_1, \ldots, B_b\}$, such that $x_i = B^T$ and $\{B_1^T, \ldots, B_b^T\} \subseteq I$, and

   (b) there does *not* exist $F \in P$, with tree-form ground instance F' such that x_i is the head of F', and the body of F' is a subset of I.

   Note that the total number of outstanding members in a given post-fixed point of $\mathcal{T}_P^+$ must be finite, and more precisely, no more than n, which is the number of H's instances used to augment the program, to which outstanding members are associated. Because of existence of outstanding members, I is *not* a post-fixed point of $\mathcal{T}_P$, but in below we can show that there exist a post-fixed point I' of $\mathcal{T}_P$, such that $I \subseteq I'$. As the theorem assumes that $\{B_1^T, \ldots, B_b^T\}$ is a subset of $\mathcal{M}$, then by Theorem 81, $x_i (= B^T) \in \mathcal{M}$, for all $1 \leq i \leq m$. In other words, $\{x_1, \ldots, x_m\} \subseteq \mathcal{M}$. Then by Lemma 79, there exist a post-fixed point I_x of $\mathcal{T}_P$, such that $\{x_1, \ldots, x_m\} \subseteq I_x$. Then let $I' = I \cup I_x$. It is easy to verify that I' is a post-fixed point of $\mathcal{T}_P$, and obviously $I \subseteq I'$.

Now we have shown that a member of $\{I \mid I \subseteq \mathcal{T}_P^+(I)\}$ is either a member of $\{I \mid I \subseteq \mathcal{T}_P(I)\}$, or a subset of a member of the latter set. Therefore $\mathcal{M}' \subseteq \mathcal{M}$. The proof is complete. □

Corollary 86 (Model-conservative program extension, II). • *Let Σ be a first-order signature, P be a set of H-formulae, and H be an H-formula. Let $\Sigma; P \looparrowright H$ have a proof in co-hohh which involves only guarded atoms, and $\mathcal{M}$ be the finite-and-infinite-term coinductive model of P. Moreover, for all tree-form ground instances $\lfloor H \rfloor^T$ of H, body $\lfloor H \rfloor^T \subseteq \mathcal{M}$.*

Let $P \cup \{H\}$ have the finite-and-infinite-term coinductive model $\mathcal{M}'$. Then, $\mathcal{M} = \mathcal{M}'$.

Corollary 87 (Model-conservative program extension, III).

- *Let Σ be a first-order signature, P be a set of H-formulae with the finite-and-infinite-term coinductive model $\mathcal{M}$, $\forall\bar{x}\ A$ be an H-formula where A is a guarded atom. Let the sequent $\Sigma; P \looparrowright \forall\bar{x}\ A$ have a co-hohh proof that only involves guarded atoms.*

Let $P \cup \{\forall\bar{x}\ A\}$ have the finite-and-infinite-term coinductive model $\mathcal{M}'$. Then, $\mathcal{M} = \mathcal{M}'$.

Theorem 88 (Using coinductively proved lemmas, I).

- *Let Σ be a first-order signature, P be a set of H-formulae, and H be an H-formula. Let $\Sigma; P \looparrowright H$ have a proof in co-hohh which involves only guarded atoms, and $\mathcal{M}$ be the finite-and-infinite-term coinductive model of P.*

- *Let $H_1, \ldots, H_n$ be distinct term-form ground instances of H, and such that for each H_k ($1 \leq k \leq n$), if $A \in$ body H_k, then $A^T \in \mathcal{M}$. Let $P \cup \{H_1, \ldots, H_n\}$ have the finite-and-infinite-term coinductive model $\mathcal{M}'$.*

- *Let the sequent $\Sigma; P, H_1, \ldots, H_n \looparrowright H'$ have a co-hohh proof that only involves guarded atoms.*

Then, $\mathcal{M} \vDash H'$.

Proof. By Theorem 81, $\mathcal{M}' \vDash H'$. By Theorem 85, $\mathcal{M} = \mathcal{M}'$. Therefore $\mathcal{M} \vDash H'$.

□

Corollary 89 (Using coinductively proved lemmas, II).

- *Let Σ be a first-order signature, P be a set of H-formulae, and H be an H-formula. Let $\Sigma; P \looparrowright H$ have a proof in co-hohh which involves only guarded atoms, and $\mathcal{M}$ be the finite-and-infinite-term coinductive model of P. Moreover, for all tree-form ground instances $\lfloor H \rfloor^T$ of H, body $\lfloor H \rfloor^T \subseteq \mathcal{M}$.*

Let the sequent $\Sigma; P, H \looparrowright H'$ have a co-hohh proof that only involves guarded atoms. Then, $\mathcal{M} \vDash H'$.

Corollary 90 (Using coinductively proved lemmas, III).

- *Let Σ be a first-order signature, P be a set of H-formulae with the finite-and-infinite-term coinductive model $\mathcal{M}$, $\forall\bar{x}\ A$ be an H-formula where A is a guarded atom. Let the sequent $\Sigma; P \looparrowright \forall\bar{x}\ A$ have a co-hohh proof that only involves guarded atoms.*

Let the sequent $\Sigma; P, \forall\bar{x}\ A \looparrowright H$ have a co-hohh proof that only involves guarded atoms. Then, $\mathcal{M} \vDash H$.

Chapter 5

Full Proof of the Soundness Theorem

We give the full proof of Theorem 81 in Section 5.1. Section 5.2 gives an example to illustrate two key observations formalized in the proof: the *composition effect* which is used in post-fixed point construction, and the *node hopping effect* which is used in verifying the constructed post-fixed point.

5.1 Proof of Theorem 81

We reproduce Theorem 81 as follows:

> Let Σ be a first-order signature, P be a set of H-formulae, and H be an H-formula. Let $\Sigma; P \looparrowright H$ have a proof in *co-hohh* which involves only guarded atoms, and $\mathcal{M}$ be the finite-and-infinite-term coinductive model of P. Then, $\mathcal{M} \vDash H$.

The steps in the proof are numbered, and are divided into three high-level steps: analysis (Section 5.1.1), construction (Section 5.1.2) and verification (Section 5.1.3).

5.1.1 Analysis of the *co-hohh* proof

1. Assume H has the form $\forall_\iota x_1 \ldots x_m \; A_1 \wedge \ldots \wedge A_n \supset A$, whose arbitrary tree-form ground instance is denoted $A'_1 \wedge \ldots \wedge A'_n \supset A'$. By definition of $\mathcal{M} \vDash H$, we need to show $\{A'_1, \ldots A'_n\} \subseteq \mathcal{M}$ implies $\{A'\} \subseteq \mathcal{M}$.

2. We use Lemma 79 from right to left, which means, to show that $\{A'\} \subseteq \mathcal{M}$, we look for a set I such that the *requirements* $\{A'\} \subseteq I$ and $I \subseteq \mathcal{T}(I)$ are satisfied.

3. The proof follows an Analysis–Construction–Verification structure, where we first study the proof of the *root sequent*

$$\Sigma; P \looparrowright \forall_\iota x_1 \ldots x_m\ A_1 \wedge \ldots \wedge A_n \supset A$$

which provides a clue for constructing a candidate set I. Finally we verify that the set I so constructed satisfies the *requirements*.

4. The proof for the root sequent starts with the following steps.

$$\dfrac{\dfrac{\dfrac{\overline{c:\iota}, \Sigma; P; ch \longrightarrow \langle (A_1 \wedge \ldots \wedge A_n \supset A)\,\overline{[x := c]}\,\rangle}{\vdots \quad \textit{(Using only } \forall R\langle\rangle \textit{ until)}}\ \forall R\langle\rangle}{\Sigma; P; ch \longrightarrow \langle \forall_\iota x_1 \ldots x_m\ A_1 \wedge \ldots \wedge A_n \supset A\rangle}\ \forall R\langle\rangle}{\Sigma; P \looparrowright \forall_\iota x_1 \ldots x_m\ A_1 \wedge \ldots \wedge A_n \supset A}\ \text{CO-FIX}$$

where ch (the coinductive hypothesis) is

$$\forall_\iota x_1 \ldots x_m\ A_1 \wedge \ldots \wedge A_n \supset A$$

and $\overline{c:\iota}$ is a short hand for $c_1 : \iota, \ldots, c_m : \iota$, and $\overline{[x := c]}$ is a short hand for

$$[x_1 := c_1] \ldots [x_m := c_m]$$

5. The $\forall R\langle\rangle$ steps in 4 extend the signature from Σ to $\overline{c:\iota}, \Sigma$. Given a formula K, we will distinguish its term-form ground instance based on Σ (denoted with $\lfloor K \rfloor$) from its term-form ground instance based on $\overline{c:\iota}, \Sigma$ (denoted with $\lfloor K \rfloor^c$). Moreover, if a lambda term L on $\overline{c:\iota}, \Sigma$, contains some (or none) of the eigenvariables $c_1, \ldots, c_m$, we will use notation L^c to refer to it, and particularly, we use B^c if and only if such a term is an atomic H-formula. We define a *referent* of $(B^c)^T$ (provided B^c is a guarded atom, and recall that we add superscript T to a guarded atom to denote the corresponding tree-atom), as

$$(B^c)^T\,[c_1 := N_1] \ldots [c_m := N_m]$$

where $N \in \mathcal{H}^\Sigma$, and, regarding the definition of substitution, $c_1, \ldots, c_m$ are treated as free variables in $(B^c)^T$.

6. The proof for the root sequent proceeds as follows:

$$\dfrac{\dfrac{\overline{c:\iota}, \Sigma; P, hp; ch \xrightarrow{D^*} A\overline{[x := c]}}{\overline{c:\iota}, \Sigma; P, hp; ch \longrightarrow \langle A\overline{[x := c]}\,\rangle}\ \text{DECIDE}\langle\rangle}{\overline{c:\iota}, \Sigma; P; ch \longrightarrow \langle (A_1 \wedge \ldots \wedge A_n \supset A)\,\overline{[x := c]}\,\rangle}\ \supset R\langle\rangle$$

where hp (the hypothesis) is $(A_1 \wedge \ldots \wedge A_n)\,\overline{[x := c]}$ and $D^* \in P \cup \{hp\}$.

7. Note that the DECIDE⟨⟩ step in 6 is the only occasion in the proof where the DECIDE⟨⟩ is used, which removes the guard ⟨⟩ in its upper sequent, so from now on, uniform proof rules are applicable to further reduce the sequent

$$\overline{c:\iota}, \Sigma; P, hp; ch \xrightarrow{D^*} A\overline{[x := c]}$$

8. We will use $\forall_\iota \bar{x}\ A_\alpha$ as the general form of a *fact* in P, and use

$$\forall_\iota \bar{x}\ A_{\alpha_1} \wedge \ldots \wedge A_{\alpha_u} \supset A_{\alpha_0}$$

as the general form of a *rule* in P, and write $A'_{\alpha_1} \wedge \ldots \wedge A'_{\alpha_u} \supset A'_{\alpha_0}$ for

$$\lfloor \forall_\iota \bar{x}\ A_{\alpha_1} \wedge \ldots \wedge A_{\alpha_u} \supset A_{\alpha_0} \rfloor^c$$

D^* can be a fact, a rule, or *hp*. D^* *cannot* be *ch*. This is due to the restriction on the DECIDE⟨⟩ rule.

9. If D^* is a fact, then the proof for the root sequent proceeds from 7 as follows:

$$\dfrac{\dfrac{\dfrac{}{\overline{c:\iota}, \Sigma; P, hp; ch \xrightarrow{\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c} A\overline{[x := c]}}\ \textsc{initial}}{\vdots \quad \textit{(Using only } \forall L \textit{ until)}}\ \forall L}{\overline{c:\iota}, \Sigma; P, hp; ch \xrightarrow{\forall_\iota \bar{x}\ A_\alpha} A\overline{[x := c]}}\ \forall L$$

where $\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c =_{\mathit{fix}\beta} A\overline{[x := c]}$. This situation is trivial and we can do the post-fixed point construction and verification immediately, as follows. Note that A' (which the head of an arbitrary tree-form ground instance of the goal mentioned in the theorem) is a referent of $\left(A\overline{[x := c]}\right)^T$. Since $\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c =_{\mathit{fix}\beta} A\overline{[x := c]}$, then by Corollary 74, $\left(A\overline{[x := c]}\right)^T = (\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c)^T$, so A' is a referent of $(\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c)^T$. A referent of $(\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c)^T$, i.e. A', is a tree-form ground instance of $\forall_\iota \bar{x}\ A_\alpha$. Then, let $I = \{A'\}$, we have $I \subseteq \mathcal{T}(I)$. Now we have verified that I satisfies the requirements (cf. 2).

10. If D^* is *hp*, then the proof for the root sequent proceeds from 7 as follows:

$$\dfrac{\dfrac{\dfrac{}{\overline{c:\iota}, \Sigma; P, hp; ch \xrightarrow{A_k\overline{[x:=c]}} A\overline{[x := c]}}\ \textsc{initial}}{\vdots \quad \textit{(Using only } \wedge L \textit{ until)}}\ \wedge L}{\overline{c:\iota}, \Sigma; P, hp; ch \xrightarrow{(A_1 \wedge \ldots \wedge A_n)\overline{[x:=c]}} A\overline{[x := c]}}\ \wedge L$$

where $A\overline{[x := c]} =_{\mathit{fix}\beta} A_k\overline{[x := c]}$ for some k such that $1 \leq k \leq n$. Then by Corollary 74, $A_k^T = A^T$, implying that the coinductive goal is a tautology.

11. If D^* is a rule, then the proof for the root sequent proceeds from 7 as follows:

$$\dfrac{\dfrac{\dfrac{\textit{SubProof 1} \quad \textit{SubProof 2}}{\overline{c:\iota}, \Sigma; P, hp; ch \overset{\lfloor \forall_\iota \bar{x}\ A_{\alpha_1} \wedge \ldots \wedge A_{\alpha_u} \supset A_{\alpha_0} \rfloor^c}{\longrightarrow} \overline{A[x := c]}} \supset L}{\vdots \quad \textit{(Using only } \forall L \textit{ until)}} \forall L}{\overline{c:\iota}, \Sigma; P, hp; ch \overset{\forall_\iota \bar{x}\ A_{\alpha_1} \wedge \ldots \wedge A_{\alpha_u} \supset A_{\alpha_0}}{\longrightarrow} \overline{A[x := c]}} \forall L$$

where *SubProof 1* is

$$\dfrac{}{\overline{c:\iota}, \Sigma; P, hp; ch \overset{A'_{\alpha_0}}{\longrightarrow} \overline{A[x := c]}} \text{INITIAL}$$

and the root of *SubProof 2* is labelled by

$$\overline{c:\iota}, \Sigma; P, hp; ch \longrightarrow A'_{\alpha_1} \wedge \ldots \wedge A'_{\alpha_u}$$

12. It is important to observe that *SubProof 2* has the following properties. We pay attention to two kinds of sequents involved in *SubProof 2*:

    (a) Sequents, whose right side is a conjunction of (at least two) atoms, in the form

    $\overline{c:\iota}, \Sigma; P, hp; ch \longrightarrow B_1^c \wedge \ldots \wedge B_p^c$

    For instance, the sequent labelling the root of *SubProof 2* is in this form.

    (b) Sequents, whose right side is a single atom, in the form

    $\overline{c:\iota}, \Sigma; P, hp; ch \longrightarrow B^c$

13. In *SubProof 2*, for each sequent in the form (12a), a succession of $\wedge R$ steps are used to break it down to a series of sequents

$$\overline{c:\iota}, \Sigma; P, hp; ch \longrightarrow B_k^c \quad 1 \leq k \leq p$$

which are in the form (12b).

14. In *SubProof 2*, for each sequent in the form (12b), only the DECIDE rule can be used as the next step of reduction, and there are four cases of possible development, as follows.

    (a) DECIDE selects a fact from P.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{c:\iota}, \Sigma; P, hp; ch \overset{\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c}{\longrightarrow} B^c} \text{INITIAL}}{\vdots \quad \textit{(Using only } \forall L \textit{ until)}} \forall L}{\overline{c:\iota}, \Sigma; P, hp; ch \overset{\forall_\iota \bar{x}\ A_\alpha}{\longrightarrow} B^c} \forall L}{\overline{c:\iota}, \Sigma; P, hp; ch \longrightarrow B^c} \text{DECIDE}$$

where $B^c =_{\mathit{fix}\beta} \lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c$.

(b) DECIDE selects a rule from P.

$$\dfrac{\dfrac{\dfrac{\dfrac{\textit{SubProof 1}'\quad\textit{SubProof 2}'}{\overline{c:\iota},\Sigma;P,hp;ch\xrightarrow{\lfloor\forall_\iota\bar{x}\ A_{\alpha_1}\wedge\ldots\wedge A_{\alpha_u}\supset A_{\alpha_0}\rfloor^c}B^c}\supset L}{\vdots\quad\textit{(Using only }\forall L\textit{ until)}}\forall L}{\overline{c:\iota},\Sigma;P,hp;ch\xrightarrow{\forall_\iota\bar{x}\ A_{\alpha_1}\wedge\ldots\wedge A_{\alpha_u}\supset A_{\alpha_0}}B^c}\forall L}{\overline{c:\iota},\Sigma;P,hp;ch\longrightarrow B^c}\text{DECIDE}$$

where *SubProof 1′* is

$$\dfrac{}{\overline{c:\iota},\Sigma;P,hp;ch\xrightarrow{A'_{\alpha_0}}B^c}\text{INITIAL}$$

and the root of *SubProof 2′* is labelled by

$$\overline{c:\iota},\Sigma;P,hp;ch\longrightarrow A'_{\alpha_1}\wedge\ldots\wedge A'_{\alpha_u}$$

which is a sequent of the form (12a). Note that the selected rule is possibly different from the rule selected in 11 although the same formula is used to depict the selected rule.

(c) DECIDE selects the coinductive hypothesis ch.

$$\dfrac{\dfrac{\dfrac{\dfrac{\textit{SubProof 1}''\quad\textit{SubProof 2}''}{\overline{c:\iota},\Sigma;P,hp;ch\xrightarrow{\lfloor\forall_\iota x_1\ldots x_m\ A_1\wedge\ldots\wedge A_n\supset A\rfloor^c}B^c}\supset L}{\vdots\quad\textit{(Using only }\forall L\textit{ until)}}\forall L}{\overline{c:\iota},\Sigma;P,hp;ch\xrightarrow{\forall_\iota x_1\ldots x_m\ A_1\wedge\ldots\wedge A_n\supset A}B^c}\forall L}{\overline{c:\iota},\Sigma;P,hp;ch\longrightarrow B^c}\text{DECIDE}$$

where $\lfloor\forall_\iota x_1\ldots x_m\ A_1\wedge\ldots\wedge A_n\supset A\rfloor^c$ refers to

$$(A_1\wedge\ldots\wedge A_n\supset A)\,[x_1:=L_1^c]\ldots[x_m:=L_m^c]$$

for some $L_1^c\ldots L_m^c$ — these terms are closed guarded full terms: since the theorem assumes that all atoms involved in the sequent proof are guarded, both before substitution and after substitution, a case analysis on any variable being substituted, reveals that the substituting term must be a guarded full term. *SubProof 1″* is

$$\dfrac{}{\overline{c:\iota},\Sigma;P,hp;ch\xrightarrow{A[x_1:=L_1^c]\ldots[x_m:=L_m^c]}B^c}\text{INITIAL}$$

and the root of *SubProof 2″* is labelled by

$$\overline{c:\iota},\Sigma;P,hp;ch\longrightarrow(A_1\wedge\ldots\wedge A_n)\,[x_1:=L_1^c]\ldots[x_m:=L_m^c]$$

which is a sequent of the form (12a). Let δ denote the substitution

$$[x_1 := L_1^c] \ldots [x_m := L_m^c]$$

Later we will see that the above mentioned δ plays a central role in the construction of the post-fixed point.

(d) DECIDE selects the hypothesis *hp*.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{c:\iota},\Sigma;P,hp;ch \overset{A_k\overline{[x:=c]}}{\longrightarrow} B^c}\;\text{INITIAL}}{\vdots\quad \textit{(Using only } \wedge L \textit{ until)}}\;\wedge L}{\overline{c:\iota},\Sigma;P,hp;ch \overset{(A_1\wedge\ldots\wedge A_n)\overline{[x:=c]}}{\longrightarrow} B^c}\;\wedge L}{\overline{c:\iota},\Sigma;P,hp;ch \longrightarrow B^c}\;\text{DECIDE}$$

where $B^c =_{fix\beta} A_k\overline{[x := c]}$ for some k such that $1 \leq k \leq n$.

5.1.2 Construction of a post-fixed point

15. We now use the above analysis of the proof of the root sequent to construct the post-fixed point I. It remains to work with the case when the program clause D^* selected by DECIDE$\langle\rangle$ is a rule (cf. 6 – 11). The main difficulty in construction in this situation arises when the post-fixed point in question is an infinite set, which happens when a proof exhibits certain irregularity.

16. Firstly, let the formula $A_1' \wedge \ldots \wedge A_n' \supset A'$ (as mentioned in 1) be computed by
$$\left(A_1^T \wedge \ldots \wedge A_n^T \supset A^T\right)[x_1 := \mathcal{N}_1] \ldots [x_m := \mathcal{N}_m]$$
for some arbitrary $\mathcal{N}_1, \ldots, \mathcal{N}_m \in \mathcal{H}^\Sigma$.

17. Note that the rules of our coinductive calculus allow us to form only one coinductive hypothesis in a proof for the root sequent, but the proof may use this coinductive hypothesis more than once. Suppose that the coinductive hypothesis *ch* is used in the given proof $s \in \omega$ times. Let $\psi = \{1, \ldots, s\}$.

18. Consider the s times when the coinductive hypothesis was used. In each case, a substitution δ was constructed, as described above in 14c. Let us denote all these substitutions by $\delta_1, \ldots, \delta_s$.

19. Each δ_j $(1 \leq j \leq s)$ is given by
$$\left[x_1 := \mathcal{L}^c_{(j,1)}\right] \ldots \left[x_m := \mathcal{L}^c_{(j,m)}\right]$$
where $\mathcal{L}^c_{(j,i)}$ is given by L_i^c in δ_j, as in 14c. (Note that each time the same coinductive hypothesis is used.)

20. Given that the proof of the root sequent uses exactly the eigenvariables $c_1, \ldots c_m$, we want to define a set of mapping for the eigenvariables into $\mathcal{H}^{\Sigma}$ in order to construct the post-fixed point. So, we define ψ^* to denote the set of all finite lists on ψ, including the empty list ϵ, and define the set $\{\Theta(w) \mid w \in \psi^*\}$ of substitutions, where Θ is a function defined by recursion, as follows.

    (a) Base case:
    $$\Theta(\epsilon) = [c_1 := \mathcal{N}_1] \ldots [c_m := \mathcal{N}_m]$$

    (b) Recursive case: If $w \in \psi^*,\ j \in \psi$, then
    $$\Theta([w, j]) = \left[c_1 := \left(\left(\mathcal{L}^c_{(j,1)}\right)^T \Theta(w)\right)\right] \ldots \left[c_m := \left(\left(\mathcal{L}^c_{(j,m)}\right)^T \Theta(w)\right)\right]$$
    where the notation $\left(\mathcal{L}^c_{(j,m)}\right)^T \Theta(w)$ is used to denote application of the substitution $\Theta(w)$ to the tree-term $\left(\mathcal{L}^c_{(j,m)}\right)^T$. Note that $\left(\mathcal{L}^c_{(j,m)}\right)^T$ is a valid notation due to Corollary 73 and the fact that $\mathcal{L}^c_{(j,m)}$ is a guarded full term (cf. 14c).

    Since ψ^* is an infinite set, this construction is potentially infinite, and moreover, through recursive application of substitutions at each iteration of Θ it can construct an infinite set of substitutions.

21. Note that the above construction involving Θ can and will be used to construct models generally, even when the coinductive hypothesis was not applied (i.e applied $s = 0$ times).

22. We define the set I^c, as the collection of all atoms $(E^c)^T$, where E^c is the atom on the right side of some sequent Q, where Q is in *SubProof 2* (cf. 11). In addition, we add $\left(\overline{A[x := c]}\right)^T$ to I^c, i.e. we add the tree corresponding to the atom occurring in the lower sequent in the DECIDE$\langle\rangle$ step (cf. 6). Note that $\overline{A[x := c]}$ in that sequent should intuitively be seen as a coinductive conclusion.

23. We use $I^c\Theta(w)$ to denote the set resulted from applying substitution $\Theta(w)$ to all members of I^c. We define I_1 as
    $$I_1 = \bigcup_{w \in \psi^*} I^c\Theta(w)$$

24. Given $\{A'_1, \ldots A'_n\} \subseteq \mathcal{M}$, by Lemma 79 (used from left to right), there exist a set I_2, such that $\{A'_1, \ldots A'_n\} \subseteq I_2$ and I_2 is a post-fixed point of $\mathcal{T}$.

25. Now we construct a candidate post-fixed point I, given as
    $$I = I_1 \cup I_2$$

5.1.3 Verification of the constructed post-fixed point

26. We verify that set I satisfies the two requirements (cf. 2).

27. $\{A'\} \subseteq I$ is proved by (cf. 16)

$$\begin{aligned} & \{A'\} \\ = & \left\{ \left(A\overline{[x := c]} \right)^T \Theta(\epsilon) \right\} \\ \subseteq & \, I^c \Theta(\epsilon) \subseteq I_1 \subseteq I \end{aligned}$$

28. It remains to show that I is a post-fixed point of $\mathcal{T}$. In other words, we need to show that each member of I is also a member of $\mathcal{T}(I)$.

29. For all $y \in I_2$, $y \in \mathcal{T}(I)$, as proven by the following argument:

$$\begin{array}{rl} I_2 \subseteq I & \text{by def. of } I \\ \mathcal{T}(I_2) \subseteq \mathcal{T}(I) & \mathcal{T} \text{ is increasing} \\ I_2 \subseteq \mathcal{T}(I_2) & \text{by def. of } I_2 \\ I_2 \subseteq \mathcal{T}(I) & \text{by transitivity of } \subseteq \end{array}$$

30. By definition of I_1, for all $y \in I_1$, $y \in I^c\Theta(w)$ for some $w \in \psi^*$, and y has an underlying atom $(F^c)^T \in I^c$ such that

$$y = (F^c)^T \Theta(w)$$

31. For an arbitrary $y \in I_1$, with underlying atom $(F^c)^T$ and substitution $\Theta(w)$ (cf. 30), let R be the sequent from where we get the atom F^c, and we show $y \in \mathcal{T}(I)$ using a case analysis 31a – 31d, with respect to the ways 14a – 14d in which the sequent R is reduced.

    (a) If R is reduced in the way 14a, then there exist a fact $\forall_\iota \bar{x}\ A_\alpha$ in P, such that $F^c =_{\mathit{fix}\beta} \lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c$, then $(F^c)^T = (\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^c)^T$ (by Corollary 74), so there exist tree-form ground instance $\lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^T$ such that $(F^c)^T \Theta(w) = \lfloor \forall_\iota \bar{x}\ A_\alpha \rfloor^T$. So $y \in \mathcal{T}(I)$.

    (b) If R is reduced in the way 14b (note that this is also the case when $F^c \equiv A\overline{[x := c]}$), then there exist a rule $K \in P$ such that $(\mathit{head}\ \lfloor K \rfloor^c) =_{\mathit{fix}\beta} F^c$, and for each $x \in (\mathit{body}\ \lfloor K \rfloor^c)$, $x^T \in I^c$. Note that $(\mathit{head}\ \lfloor K \rfloor^c)^T = (F^c)^T$ by Corollary 74, then there exist a tree-form ground instance $\lfloor K \rfloor^T$ for K, whose head is $(F^c)^T \Theta(w)$, and whose body S is the set such that $x' \in S$ if and only if there exist $x \in (\mathit{body}\ \lfloor K \rfloor^c)$ and $x' = x^T \Theta(w)$. Then, for all $x' \in S$, $x' \in I^c\Theta(w) \subseteq I$. So $y \in \mathcal{T}(I)$.

(c) If R is reduced in the way 14c, then the use of *ch* results in existence of δ_r for some $r \in \psi$, given as

$$\left[x_1 := \mathcal{L}^c_{(r,1)}\right] \dots \left[x_m := \mathcal{L}^c_{(r,m)}\right]$$

and $F^c =_{\mathit{fix}\beta} A\delta_r$. Note that

$$(A\,\delta_r)^T\,\Theta\,(w) \;=\; \left(A\overline{[x := c]}\right)^T \Theta\,([w,r])$$

because, by definition of Θ,

$$\Theta\,([w,r]) = \left[c_1 := \left(\left(\mathcal{L}^c_{(r,1)}\right)^T \Theta\,(w)\right)\right] \dots \left[c_m := \left(\left(\mathcal{L}^c_{(r,m)}\right)^T \Theta\,(w)\right)\right]$$

So $y \in I^c\Theta\,([w,r])$ and this membership is underlain by $\left(A\overline{[x := c]}\right)^T \in I^c$. In light of this discovery, we start a new round of case analysis where we regard the same y as a member of $I^c\Theta\,([w,r])$, then 31b applies, so $y \in \mathcal{T}\,(I)$.

(d) If R is reduced in the way 14d, it implies that $F^c =_{\mathit{fix}\beta} A_k\overline{[x := c]}$ for some k. Further, by Corollary 74, $(F^c)^T = \left(A_k\overline{[x := c]}\right)^T$. Then $(F^c)^T\,\Theta\,(w) = \left(A_k\overline{[x := c]}\right)^T \Theta\,(w)$. We further inspect two sub-cases.

i. $w = \epsilon$. Then $y = A'_k$, and $A'_k \in I_2$, so $y \in \mathcal{T}\,(I)$ (cf. 29).

ii. $w \neq \epsilon$. Then there exist $v \in \psi^*, i \in \psi$ such that $[v,i] = w$. Then, there exist a use of *ch* that gives rise to δ_i, given as

$$\left[x_1 := \mathcal{L}^c_{(i,1)}\right] \dots \left[x_m := \mathcal{L}^c_{(i,m)}\right]$$

In addition, there exist $(A_k\delta_i)^T \in I^c$ (cf. 14c and 22). Note that

$$(A_k\,\delta_i)^T\,\Theta\,(v) \;=\; \left(A_k\overline{[x := c]}\right)^T \Theta\,(w)$$

This is because , by definition of Θ, that

$$\begin{aligned} &\Theta\,(w) = \Theta\,([v,i]) \\ = &\left[c_1 := \left(\left(\mathcal{L}^c_{(i,1)}\right)^T \Theta\,(v)\right)\right] \dots \left[c_m := \left(\left(\mathcal{L}^c_{(i,m)}\right)^T \Theta\,(v)\right)\right] \end{aligned}$$

Thus, $y \in I^c\Theta\,(v)$, underlain by $(A_k\delta_i)^T \in I^c$. In light of this discovery, we start a new round of case analysis where we regard the same y as a member of $I^c\Theta\,(v)$. Note that whenever this branch, i.e. 31(d)ii, is reached, we take a new round of case analysis, thus entering into iteration if we repeatedly reach this branch. Since in each new round, we work with a smaller index, i.e. from $w = [v, i]$ to v, iteration caused by this branch always terminates, ending in one of the branches 31a, 31b, 31c or 31(d)i in the final round. In all these possible final circumstances, previous analysis has shown that $y \in \mathcal{T}(I)$.

32. Now we have finished with showing that $I \subseteq \mathcal{T}(I)$, and the proof is complete.

5.2 An Example for Key Observations

We give an example to assist with understanding the proof of Theorem 81 in Section 5.1. The key observations are the *composition effect*, which backs the formalization in list item[1] 20, and the *node hopping effect*, which backs the formalization in list items 31c and 31d.

Let first-order signature Σ be the union of the set of logical constants, with the set $\{q : \iota \to o,\ g : \iota \to \iota,\ s : \iota \to \iota,\ z : \iota\}$, and P be the following program on Σ adapted from [7].

$$\forall x\ q\,(s\,(g\ x)) \wedge q\,(s\,(g\,(s\ x))) \wedge q\ x \supset q\,(s\ x) \tag{5.1}$$

$$\forall x\ q\ x \supset q\,(g\ x) \tag{5.2}$$

$$q\ z \tag{5.3}$$

For visual simplicity, in the rest of this example we omit all parentheses when writing an atomic formula formed by $q, s, g, z \in \Sigma$ and $c : \iota \notin \Sigma$, which should not cause any ambiguity, considering the type of these symbols.

There is a *co-fohh* proof (which is a *co-hohh* proof as well) of

$$\Sigma; P \looparrowright \forall x\ q\ x \supset q\,(s\ x)$$

(the root sequent), given by Figure 5.1, from where we collect all nodes and form the set I^c (cf. list item 22).

There are three points where the coinductive hypothesis is used, so we have $\psi = \{1, 2, 3\}$ (cf. list item 17). The δ's (cf. list item 14c) are given by:

$$\delta_1 \text{ is } [x := g\,c] \qquad \delta_2 \text{ is } [x := g\,s\,c] \qquad \delta_3 \text{ is } [x := c]$$

Let $\Theta(\epsilon) = [c := z]$. Then

$$\begin{aligned} \Theta([1]) = {} & [c := ((g\,c)\,\Theta(\epsilon))] = [c := g\,z] \\ \Theta([2]) = {} & [c := ((g\,s\,c)\,\Theta(\epsilon))] = [c := g\,s\,z] \\ & \ldots \end{aligned}$$

The construction of Θ is based on the composition effect of δ's, illustrated by Figure 5.2. The verification of the post-fixed point construction makes use of the node hopping effect, illustrated in Figure 5.3.

[1] In this section, when we say "list item", we refer to a list item in Section 5.1.

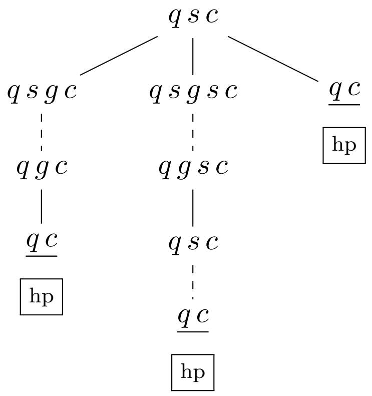


Figure 5.1: Co-fohh Proof, simplified presentation. Dashed edges are from nodes proved using the coinductive hypothesis. Solid edges are from nodes proved using implicative clauses from the program P. Underlined nodes with label “hp” are proved using the formula added by the $\supset R\langle\rangle$ step.

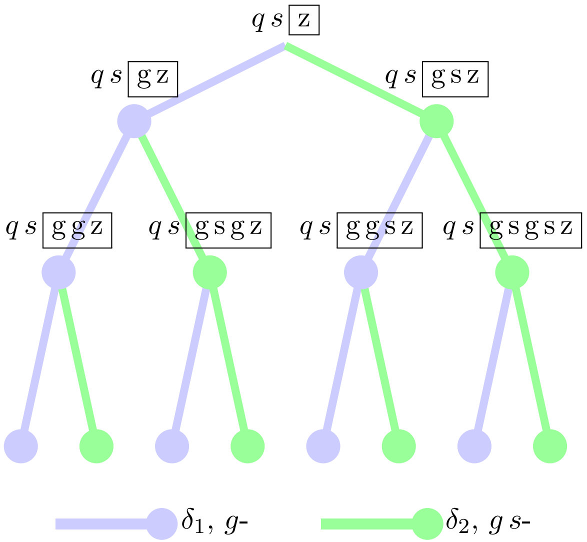


Figure 5.2: Composition effect. The tree shows development of the proof for $q\,s\,z$ using clause (0), which rules that a node in the form $q\,s\,c$ has children in the form $q\,s\,g\,c$ (rendered as left child) and $q\,s\,g\,s\,c$ (rendered as right child). Note that the $g\,c$ part in the left child and the $g\,s\,c$ part in the right child can both be regarded as a whole (rendered as boxed) so that both children again conform to the form $q\,s\,c$. Nodes due to atom $q\ x$ in clause (0) are omitted. The composition effect of δ_1 and δ_2 can be observed in the following way: choose a path in the tree, then follow the path from top to bottom, and at each labelled node on the path, see letters within the box. Moreover, the labelled part of the tree can be regarded as being assembled using trees from Figure 5.3.

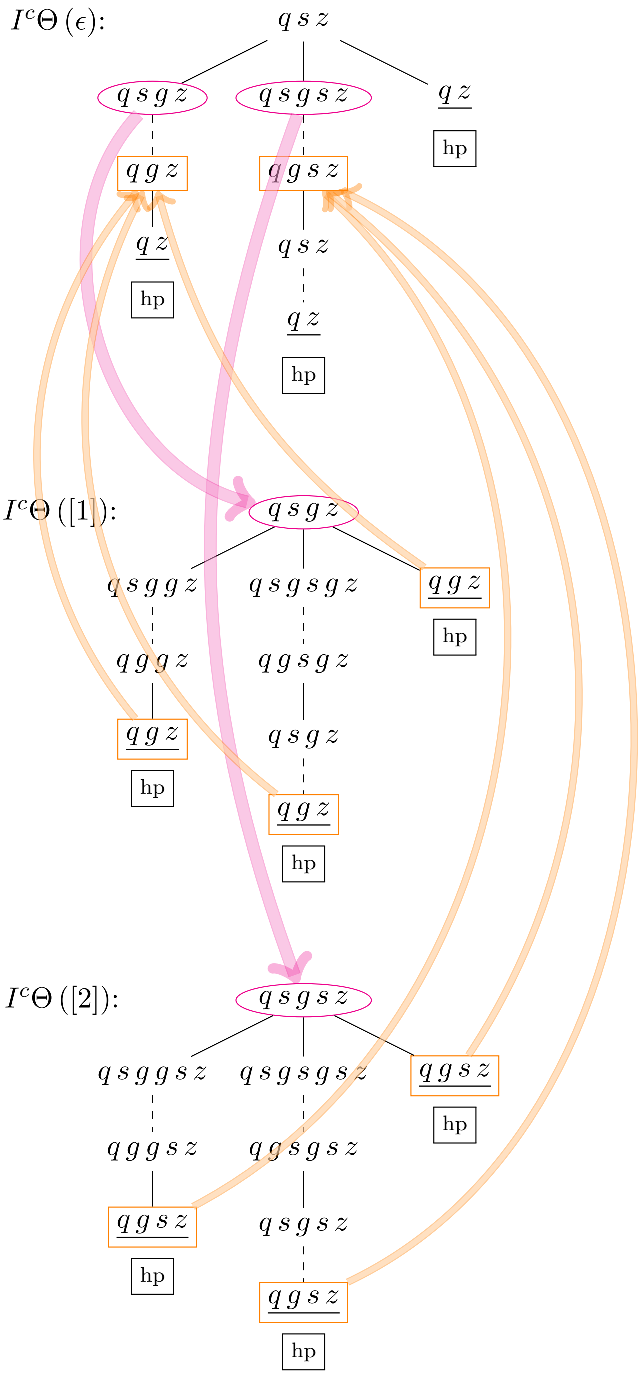


Figure 5.3: Node hopping effect. A node hops upwards in the diagram to find a copy of itself, if and only if its underlying atom in I^c are proved using *hp*. A node hops downwards in the diagram to find a copy of itself, if its underlying atom in I^c is proved using *ch*.

Chapter 6

Conclusion, Discussion and Future Work

6.1 Conclusion

We have introduced coinductive uniform proof as a meta-theory for coinduction in first-order Horn clause logic programming, and proved its soundness w.r.t the Hebrand-style greatest fixed point models. Although automated discovery of coinductive invariants is generally not decidable , this principled approach still helps future development of heuristic algorithms that can advance the state of the art [12].

We can have eight calculi if we further distinguish coinductive uniform proofs that involve fixed point terms (denoted *co-fohc*$^{\mathit{fix}}$, *co-fohh*$^{\mathit{fix}}$, *co-hohc*$^{\mathit{fix}}$ and *co-hohh*$^{\mathit{fix}}$), from those that do not (denoted *co-fohc*, *co-fohh*, *co-hohc* and *co-hohh*). We can then assign different examples to these calculi.

co-fohc$^{\mathit{fix}}$	*co-fohh*$^{\mathit{fix}}$	*co-hohc*$^{\mathit{fix}}$	*co-hohh*$^{\mathit{fix}}$
		Ex.2	Ex.4

co-fohc	*co-fohh*	*co-hohc*	*co-hohh*
Ex.1	Ex.3		

6.2 Discussion

There are cases that coinductive uniform proof cannot handle. Consider the logic program "unbound clock" defining $p : \iota \to \iota \to o$.

$$\forall_\iota m\ (p\ (s\ m)\ 0 \supset p\ 0\ m)$$
$$\forall_\iota mn\ (p\ n\ (s\ m) \supset p\ (s\ n)\ m)$$

Suppose we want to prove $p\ 0\ 0$. The operational view of the SLD-derivation for the goal w.r.t. this program is that it is a non-terminating counter, counting from n back to 0, then reset and count from $n+1$ back to 0, and so on, for $n \geq 0$. There are two possible ways to approach this problem: resorting to higher-order programming, or, including an induction rule in the proof system [12]. Both ways are *not* covered by the soundness theorems, and here we focus on discussing the first way.

The following (higher-order) program P can be introduced [12] to quantify over the predicate p from the "unbound clock" program, which is now set to a variable q of functional type:

1. $\forall_{\iota\to\iota\to\iota} q \forall_{\iota} m \; (\mathit{clock}\ (q\ (s\ m)\ 0) \supset \mathit{clock}\ (q\ 0\ m))$
2. $\forall_{\iota\to\iota\to\iota} q \forall_{\iota} mn \; (\mathit{clock}\ (q\ n\ (s\ m)) \supset \mathit{clock}\ (q\ (s\ n)\ m))$

where $\mathit{clock} : \iota \to o$ is a first-order predicate. A coinductive invariant for program P is

$$\forall_{\iota\to\iota\to\iota} q \; (\mathit{clock}\ (q\ 0\ 0))$$

which is proved in Figure 6.1. Given a constant $f : \iota \to \iota \to \iota$, the atom $\mathit{clock}\ (f\ 0\ 0)$ follows as a corollary.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c,\Sigma;P;CH \xrightarrow{\mathit{clock}\ (c\ 0\ 0)} \mathit{clock}\ (c\ 0\ 0)}\ \text{INIT} \qquad \dfrac{\dfrac{\dfrac{}{c,\Sigma;P;CH \xrightarrow{\mathit{clock}\ (c\ (s\ 0)\ 0)} \mathit{clock}\ (c\ (s\ 0)\ 0)}\ \text{INIT}}{c,\Sigma;P;CH \xrightarrow{CH} \mathit{clock}\ (c\ (s\ 0)\ 0)}\ \forall L}{c,\Sigma;P;CH \longrightarrow \mathit{clock}\ (c\ (s\ 0)\ 0)}\ \text{DEC}\checkmark}{c,\Sigma;P;CH \xrightarrow{\mathit{clock}\ (c\ (s\ 0)\ 0) \supset \mathit{clock}\ (c\ 0\ 0)} \mathit{clock}\ (c\ 0\ 0)}\ \supset L}{c,\Sigma;P;CH \xrightarrow{P(1)} \mathit{clock}\ (c\ 0\ 0)}\ \forall L\ \text{(2 times)}}{c : \iota\to\iota\to\iota,\Sigma;P;CH \longrightarrow \langle \mathit{clock}\ (c\ 0\ 0)\rangle}\ \text{DEC}\langle\rangle}{\Sigma;P;CH \longrightarrow \langle \forall q\ \mathit{clock}\ (q\ 0\ 0)\rangle}\ \forall R\langle\rangle}{\Sigma;P \looparrowright \forall q\ \mathit{clock}\ (q\ 0\ 0)}\ \text{CO-FIX}$$

Figure 6.1: The DECIDE step with ✓ selects $CH =_{def} \forall_{\iota\to\iota\to\iota} q\, (\mathit{clock}\, (q\, 0\, 0))$, and the following $\forall L$ step applies substitution $[q \mapsto \lambda xy\,.\,c\ (s\ x)\ y]$.

Rigid Atoms In the goal $\forall_{\iota\to\iota\to\iota} q\ (\mathit{clock}\ (q\ 0\ 0))$, the *clock* predicate is essential because it allows to quantify over the functional variable q. One may consider using the following program, which involves flexible atoms and gives rise to a similar coinductive uniform proof.

$$\forall_{\iota\to\iota\to o} q \forall_{\iota} m\ (q\ (s\ m)\ 0 \supset q\ 0\ m)$$
$$\forall_{\iota\to\iota\to o} q \forall_{\iota} mn\ (q\ n\ (s\ m) \supset q\ (s\ n)\ m)$$

Miller's uniform proof disallows the above syntax. Following a similar technique to that of Miller, we can show that any *closed* atomic formula B

has a coinductive proof with respect to the above program, indicating that our program is coinductively inconsistent, as follows. First we declare B as a coinductive hypothesis. Then we back-chain with the first clause (or with the second in a similar way), instantiating q with $\lambda xy.B$, and instantiating m by an arbitrary constant, so we get the instance $B \supset B$. The sub-goal is B, which is proved using the coinductive hypothesis B.

Alternative Invariant In fact, the most general coinductive invariant for program P is not $\forall_{\iota\to\iota\to\iota} q\ (\mathit{clock}\ (q\ 0\ 0))$, but $\forall_\iota x\ (\mathit{clock}\ x)$, proved in Figure 6.2.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c:\iota,\Sigma;P;CH \xrightarrow{\mathit{clock}\ c} \mathit{clock}\ c}\ \textsc{init} \qquad \dfrac{\dfrac{\dfrac{}{c:\iota,\Sigma;P;CH \xrightarrow{\mathit{clock}\ c} \mathit{clock}\ c}\ \textsc{init}}{c:\iota,\Sigma;P;CH \xrightarrow{\forall x\ (\mathit{clock}\ x)} \mathit{clock}\ c}\ \forall L}{c:\iota,\Sigma;P;CH \longrightarrow \mathit{clock}\ c}\ \textsc{dec}\ \text{(selects } CH\text{)}}{c:\iota,\Sigma;P;CH \xrightarrow{\mathit{clock}\ c\supset \mathit{clock}\ c} \mathit{clock}\ c}\ \supset L}{c:\iota,\Sigma;P;CH \xrightarrow{\forall q\forall m\ (\mathit{clock}\ (q\ (s\ m)\ 0)\supset \mathit{clock}\ (q\ 0\ m))} \mathit{clock}\ c}\ \forall L\ \text{(2 times)}}{c:\iota,\Sigma;P;CH \longrightarrow \langle \mathit{clock}\ c\rangle}\ \textsc{dec}\langle\rangle\ \text{(selects } P(1)\text{)}}{\dfrac{\Sigma;P;CH \longrightarrow \langle \forall x\ (\mathit{clock}\ x)\rangle}{\Sigma;P \looparrowright \forall x\ (\mathit{clock}\ x)}\ \textsc{co-fix}}\ \forall R\langle\rangle
$$

Figure 6.2: $CH =_{def} \forall x\ (\mathit{clock}\ x)$. $\forall L\langle\rangle$ applies $[q := \lambda xy\,.\,c][m := 0]$. The $\forall L$ step applies the substitution $[x := c]$.

This program P can also find its place in the table of calculi, though the soundness is not confirmed (the latter fact is denoted by "?").

co-fohc^{fix}	*co-fohh*^{fix}	*co-hohc*^{fix}	*co-hohh*^{fix}
co-fohc	*co-fohh*	*co-hohc*	*co-hohh*
			P ?

6.3 Future Work

The table gives direction for future work. We need to discover more examples to fill in the blanks, and we shall explore extending the soundness result. Adding induction rule is also an interesting area to explore.

co-fohc^{fix}	*co-fohh*^{fix}	*co-hohc*^{fix}	*co-hohh*^{fix}
		Ex.2	Ex.4
co-fohc	*co-fohh*	*co-hohc*	*co-hohh*
Ex.1	Ex.3		P ?

A plan for the final submission of my PhD thesis is that I shall make the submission no later than the January of 2020, which is the expiration time of my current UK visa.

Bibliography

[1] H. Basold. *Mixed Inductive-Coinductive Reasoning: Types, Programs and Logic.* PhD thesis, 2018.

[2] Henning Basold. personal communication, 2018.

[3] Y. Bertot and E. Komendantskaya. Inductive and coinductive components of corecursive functions in Coq. *ENTSC*, 203(5):25–47, 2008.

[4] J. Blanchette et al. Foundational nonuniform (co)datatypes for higher-order logic. In *LICS'17*, pages 1–12. IEEE Computer Society, 2017.

[5] T. Coquand. Infinite objects in type theory. In *TYPES'93*, volume 806, pages 62–78, 1994.

[6] G. Dowek and J.-J. Lévy. *Introduction to the Theory of Programming Languages.* Springer London, London, 2011.

[7] Peng Fu, Ekaterina Komendantskaya, Tom Schrijvers, and Andrew Pond. Proof relevant corecursive resolution. In Oleg Kiselyov and Andy King, editors, *Functional and Logic Programming*, pages 126–143, Cham, 2016. Springer International Publishing.

[8] E. Giménez. Structural recursive definitions in type theory. In *ICALP'98*, pages 397–408, 1998.

[9] Gopal Gupta, Ajay Bansal, Richard Min, Luke Simon, and Ajay Mallya. Coinductive logic programming and its applications. In Véronica Dahl and Ilkka Niemelä, editors, *Logic Programming*, pages 27–44, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[10] J. Harland. *On Hereditary Harrop Formulae as a Basis for Logic Programming.* PhD Thesis, University of Edinburgh, Edinburgh, UK, 1991.

[11] E. Komendantskaya and Y. Li. Productive corecursion in logic programming. *J. TPLP (ICLP'17 post-proc.)*, 17(5-6):906–923, 2017.

[12] Ekaterina Komendantskaya. personal communication, 2018.

[13] R. Kowalski. Predicate logic as a programming language. *Information Processing*, 74:569–574, 1974.

[14] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2nd edition, 1987.

[15] D. Miller and G. Nadathur. *Programming with Higher-order logic.* Cambridge University Press, 2012.

[16] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. *Uniform Proofs as a Foundation for Logic Programming*, volume 51 of *Annals of Pure and Applied Logic*, pages 125–157. Elsevier, 1991.

[17] R. Nederpelt and H. Geuvers. *Type Theory and Formal Proof: An Introduction.* Cambridge University Press, New York, NY, USA, 1st edition, 2014.

[18] B.C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

[19] D. Sangiorgi. *Introduction to Bisimulation and Coinduction.* Cambridge University Press, 2011.

[20] Robert M. Solovay. Provability interpretations of modal logic. *Israel Journal of Mathematics*, 3(25):287 – 304, 1976.

Appendix A

Choices in System Design

A.1 Motivation for the Safeguard

Consider an experimental proof system that consists of uniform proof rules (Figure 3.1) and a variety $\textsc{co-fix}'$ of the adopted $\textsc{co-fix}$ rule, so that the system does not involve the safeguard.

$$\frac{\Sigma; P; M \longrightarrow M}{\Sigma; P \looparrowright M}\ \textsc{co-fix}'$$

We show that using the experimental system, any goal can be proven, which implies its inconsistency. Without loss of generality, consider a sequent $\Sigma; \emptyset \looparrowright A$, where A is an arbitrary closed atomic formula.

$$\cfrac{\cfrac{\cfrac{}{\Sigma; \emptyset; A \xrightarrow{A} A}\ \textsc{initial}}{\Sigma; \emptyset; A \longrightarrow A}\ \textsc{decide}}{\Sigma; \emptyset \looparrowright A}\ \textsc{co-fix}'$$

Whereas with the safeguard, the same sequent cannot be proven because nothing can be selected from $\emptyset$, as follows.

$$\cfrac{\cfrac{\dagger}{\Sigma; \emptyset; A \longrightarrow \langle A \rangle}\ \textsc{decide}\langle\rangle}{\Sigma; \emptyset \looparrowright A}\ \textsc{co-fix}$$

Also note that the coinductive models for the empty program $\emptyset$ are both $\emptyset$, therefore without safeguard the system is neither sound.

A.2 Guarded Implication-Right Rule

There is a choice concerning the guarded version of the $\supset R$ rule in uniform proof.

$$\frac{\Sigma; P, D; \Delta \longrightarrow G}{\Sigma; P; \Delta \longrightarrow D \supset G} \supset R$$

There are two options, $\supset R\langle\rangle$ and $\supset R'\langle\rangle$, as follows.

$$\frac{\Sigma; P; \Delta, M_1 \longrightarrow \langle M_2 \rangle}{\Sigma; P; \Delta \longrightarrow \langle M_1 \supset M_2 \rangle} \supset R\langle\rangle \qquad\qquad \frac{\Sigma; P, M_1; \Delta \longrightarrow \langle M_2 \rangle}{\Sigma; P; \Delta \longrightarrow \langle M_1 \supset M_2 \rangle} \supset R'\langle\rangle$$

A.2.1 Motivation

This choice arose when I tried to motivate the difference between $\supset R$ and $\supset R\langle\rangle$ regarding which field to put the left side of $\supset$ in the upper sequent. The former adds the left side of $\supset$ to P while the latter adds the left side of $\supset$ to Δ. In $\supset R\langle\rangle$, the motivation to add M_1 to the Δ field was to prevent it from being selected by the DECIDE$\langle\rangle$ rule, but the further motivation for this act of prevention, despite its symmetry to later modality [1], is not immediately clear. I checked earlier research notes, but there was no record, such as an example showing adverse effect like unsoundness, as to why M_1 cannot be selected DECIDE$\langle\rangle$.

The decision to use a separate field, i.e. Δ in addition to P, for guarded items on the left side of sequents [2, 12] now seems to have provided the clarification that facilitates further exploration.

A.2.2 Observation

We can have an alternative version $\supset R'\langle\rangle$ of $\supset R\langle\rangle$, the former of which follows the style of $\supset R$. The difference between $\supset R\langle\rangle$ and $\supset R'\langle\rangle$ has no influence on the soundness result. It is just a matter of preference. For instance, assume a coinductive goal of the form $A_1 \wedge \cdots \wedge A_n \supset A$ where A and variants thereof denote closed atoms, we shall have the following proof.

$$\begin{array}{cl}
\hline & \textsc{initial} \\
\Sigma; P, A_1 \wedge \cdots \wedge A_n; A_1 \wedge \cdots \wedge A_n \supset A \xrightarrow{A_k} A & \\
\hline & \wedge L \\
\cdots \ \text{(using } \wedge L \text{ until)} & \\
\hline & \wedge L \\
\Sigma; P, A_1 \wedge \cdots \wedge A_n; A_1 \wedge \cdots \wedge A_n \supset A \xrightarrow{A_1 \wedge \cdots \wedge A_n} A & \\
\hline & \textsc{decide}\langle\rangle \\
\Sigma; P, A_1 \wedge \cdots \wedge A_n; A_1 \wedge \cdots \wedge A_n \supset A \longrightarrow \langle A \rangle & \\
\hline & \supset R'\langle\rangle \\
\Sigma; P; A_1 \wedge \cdots \wedge A_n \supset A \longrightarrow \langle A_1 \wedge \cdots \wedge A_n \supset A \rangle & \\
\hline & \textsc{co-fix} \\
\Sigma; P \looparrowright A_1 \wedge \cdots \wedge A_n \supset A &
\end{array}$$

We can infer from the proof steps that $A_k =_{\mathit{fix}\beta} A$, then $A_k^T = A^T$, implying that the coinductive goal is a tautology.

There are further separate observations.

1. The above proof pays no regard as to whether A has a predicate in or *not* in P, so in both cases the proof holds. However, if the latter is the case [12], and the goal is a tautology, and we use $\supset R\langle\rangle$, then we cannot make a proof. An example is to prove $\Sigma; \emptyset \looparrowright p \supset p$ with $\supset R'\langle\rangle$ (success) then with $\supset R\langle\rangle$ (failure).

2. In the above proof, if the $\textsc{decide}\langle\rangle$ happens to select from P, then the proof proceeds as in the theory with $\supset R\langle\rangle$.

A.2.3 Conclusion

We decided to adopt $\supset R'\langle\rangle$ for the following reasons:

- The theory with $\supset R'\langle\rangle$ properly[1] subsumes the theory with $\supset R\langle\rangle$.

- $\supset R'\langle\rangle$ does not affect soundness.

- The symmetry between $\supset R'\langle\rangle$ and $\supset R$ simplifies the theory, compared with the alternative theory that involves the difference between $\supset R\langle\rangle$ and $\supset R$.

[1] "Proper" is as in "proper super set".