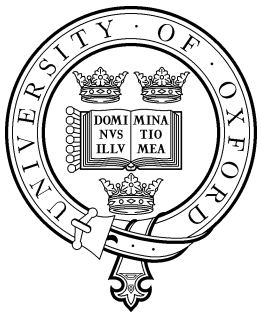
Higher-order Constrained Horn Clauses for Higher-order Program Verification



Jerome Jochems

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2020

Acknowledgements

Sincere thanks go to my supervisor, Luke Ong, for introducing me to higher-order model checking and higher-order constrained Horn clauses, and for his unwavering encouragement and support throughout the project.

I would like to thank Steven Ramsay for his helpful and detailed feedback on parts of the thesis, as well as his collaboration in HoCHC project meetings along with Toby Cathcart Burn, Long Pham, and Dominik Wagner.

Finally, I gratefully acknowledge the financial support from the Scatcherd European Scholarship and the Engineering and Physical Sciences Research Council that facilitated this work.

Abstract

Higher-order constrained Horn clauses (HoCHC) are a fragment of higher-order logic modulo theories recently introduced by Cathcart Burn et al. (2018). This thesis explores the adequacy of HoCHC as a unifying framework for the algorithmic verification of higher-order programs, through links with existing approaches to program verification as well as a sound and refutationally complete solution method for HoCHC.

We establish a continuous interpretation for HoCHC in which the semantic domains are restricted to Scott-continuous functions. We show that the continuous HoCHC decision problem is equivalent to the original problem. This continuous interpretation serves as a foundation for a sound and refutation-complete resolution proof system—based on SLD-resolution—for solving unsatisfiable instances of the HoCHC problem over a semi-decidable background theory.

To address the relation between HoCHC and higher-order model checking, we introduce a coinductive version of HoCHC that is characterised by a greatest fixpoint construction. We define an encoding of higher-order recursion schemes into a HoCHC logic program, which can be used to reduce the open higher-order recursion scheme equivalence problem—and thus the $\lambda\mathbf{Y}$-calculus Böhm tree equivalence problem—to semi-decidability of coinductive HoCHC over Maher's complete and decidable background theory of trees (Maher, 1988; Djelloul et al., 2008).

Finally, we develop an axiomatic basis for relations as higher-order program invariants in the form of a Hoare logic. Our approach to Hoare logic for higher-order programs distinguishes itself from the literature through the combination of a higher-order assertion logic and a relatively complete proof system.

Table of contents

Chapter 1

Introduction

In recent years, first-order logic has emerged as a framework in which to understand automatic verification of first-order imperative programs. Automatic safety verification algorithms can be understood in terms of the logical character of the invariants they construct. Thus, first-order logic provides a universal setting in which even disparate verification algorithms can be compared. In a sense, first-order logic acts as a neutral intermediate language for automatic first-order verification, independent of application or programming language.

However, such a principled approach to understanding the breadth of automatic verification appears to be lacking for functional programs. Although first-order logic plays an important role in many approaches to higher-order program verification, it fails to provide a unifying paradigm, as many invariants synthesized by established approaches to higher-order program verification do not have a natural description in first-order logic (e.g. intersection types and higher-order pushdown automata, see Ramsay, 2014, and Hague et al., 2008, resp.).

Cathcart Burn, Ong, and Ramsay (2018) have proposed a promising logical framework in which to understand higher-order program verification, following advances in the first-order verification community (Bjørner et al., 2013a, 2015): the higher-order constrained Horn clause (HoCHC) fragment of higher-order logic.

The HoCHC fragment has been shown to express certain invariants of higher-order programs quite directly, yet retains many of the excellent algorithmic properties to which first-order constrained Horn clauses owe their suitability for first-order model checking. In fact, HoCHC seems to be a remarkably well-behaved fragment of higher-order logic, which is otherwise renowned for its intractability.

This thesis explores two questions pertaining to the adequacy of HoCHC as a setting for higher-order program verification:

(i) Is the fragment sufficiently expressive to capture the problems typically studied by the higher-order program verification community?

(ii) Are there (relatively) complete solution methods (e.g. algorithms that are sound and complete in the presence of an oracle for first-order constraint solving)?

We expect these questions to establish HoCHC as a robust fragment of higher-order logic, algorithmically and semantically, paving the way for HoCHC as a comprehensive verification framework to rival existing approaches to higher-order program verification.

1.1 First-order program verification

Because thorough testing of software is excessively expensive and does not provide strong guarantees about correctness, the field of verification has gained a lot of traction. In this area, we employ mathematical techniques to prove whether a given program has a specified property. Properties of interest include *safety* properties (can a 'bad' event ever happen?) and *liveness* properties (is a 'good' event guaranteed to happen?).

Using (first-order) constrained Horn clauses for expressing first-order verification problems was advocated by Bjørner et al. (2013a); these are Horn clauses of first-order logic containing constraints from some suitable background theory. Bjørner et al. argue that Horn clauses are a suitable standard format for verification problems, because many first-order verification problems can be framed as satisfiability problems for systems of constrained Horn clauses (see Beyene et al., 2013; Bjørner et al., 2013b, 2015).

The (purely logical) 'unifying framework' for first-order verification provided by constrained Horn clauses has two main benefits. On the one hand, it facilitates the development of a vast collection of benchmarks to understand and compare different approaches to verification. On the other hand, it enables the reuse of tools; highly efficient SMT and constrained Horn clause solvers (e.g. Z3[1] by de Moura and Bjørner, 2008, and SeaHorn[2] by Gurfinkel et al., 2015) can be exploited by a large number

[1] https://github.com/Z3Prover/z3
[2] https://seahorn.github.io/

of program verification tools that offload some complex task (invariant finding is a typical example) to a solver by framing it in terms of constrained Horn clauses.

1.1.1 Coinductive logic programming

Some programs do not have natural inductive interpretations, notably programs over infinite data types like infinite lists, infinite trees, and streams. Such data types typically arise as the observable behaviour of a program that runs forever and can be represented in functional programming languages with lazy evaluation—as in Haskell—and in logical programming languages (Gupta et al., 2007; Simon et al., 2007).

Coinduction allows us to reason about unfounded sets and unbounded data types in a way that induction does not. Coinductive interpretations correspond to greatest fixpoint semantics (cf. the least fixpoint semantics of induction).

Let us consider the following logic program that defines a list.

```
list([n|l]) := integer(n), list(l)
```

Under the usual least fixpoint semantics, `list` is assigned the empty relation, because there is no 'base case' without self-references. If we interpret the program coinductively, however, then `list` holds of all infinite lists of integers. Consider the extension of this program with a base case:

```
list([]).
list([n|l]) := integer(n), list(l)
```

Then, of course, the inductive interpretation is the set of all finite lists. The coinductive interpretation of this program is the set of all finite and infinite lists, since the least fixpoint is included in the greatest fixpoint by virtue of the base case.

Induction and coinduction have their roots in category theory. An inductive (least fixpoint) interpretation corresponds to an initial algebra, an initial object in the category of F-algebras for an endofunctor F. This initiality provides a general framework for induction and recursion. A coinductive (greatest fixpoint) interpretation is precisely its dual: a final coalgebra in the category of F-coalgebras.

Intuitively, coinduction 'observes' the structure of an object (destructs it), instead of building up an object over base cases (constructing it), as induction does. Induction describes terminating computations, while coinduction describes observable behaviour

of (possibly non-terminating) processes. Like inductive definitions use recursion, so do we refer to coinductive self-referential definitions as *corecursion*.

The following simple program has vastly different inductive and coinductive interpretations.

```
P(X) := P(X)
```

In the inductive setting, the least model is built up from the empty relation. Because there is no base case, the empty relation is also the least model. In a coinductive interpretation, the greatest model is derived from and is, in fact, the universal relation.

Co-logic programming (CoLP) extends the traditional declarative and operational semantics of logic programming to allow reasoning over infinite structures and properties (Gupta et al., 2007; Simon et al., 2007), thus unifying logic programming and coinductive logic programming. Key to the operational semantics of coinductive logic programming—which may not be captured by finite proofs—is the introduction of the *coinductive hypothesis rule*, reminiscent of the bisimulation proof method (Sangiorgi, 1998). According to this rule, if the current resolvent contains a call C that unifies with an earlier call C', then the call succeeds.

CoLP can only handle those infinite proofs that are *regular*, i.e. have a finite number of distinct subproofs (see e.g. Komendantskaya et al., 2016; Komendantskaya and Li, 2018). Although Komendantskaya and Johann (2015) and Basold et al. (2019) have since extended this proof search to include some irregular proofs, proof search for coinductive programs remains incomplete.

As Gupta et al. (2007) point out, verifying liveness properties in CoLP is natural and elegant. CoLP has been implemented in Prolog and Logtalk (Ancona, 2013; Moura, 2013).

1.2 Higher-order program verification

Nowadays higher-order constructs such as lambdas and streams are standard features of some of the most frequently used programming languages, e.g. C++, Java, C#, and Python. On top of that, functional languages are often favoured in parallel and distributed programming because of the lack of side-effects. With functional programming moving into the mainstream from niche applications in academia and the financial sector, the need for methods to verify higher-order programs has never been greater.

Functions are the building blocks in functional programming languages like Haskell, Standard ML, OCaml, and F#. Functional programming becomes truly higher-order when functions take other functions as input and output, rather than mere objects. An archetypical example is the Haskell `map` function that applies a function element-wise to a list and returns the resulting list:

```
map :: (a -> b) -> [a] -> [b]
map _ []     = []
map f (x:xs) = f x : map f xs
```

To derive properties of such a function, higher-order program verification must reason about all functions of the appropriate type, while simultaneously accounting for recursion and infinite data types. First-order imperative programming has a clear distinction between data (objects) and control (states), whereas this line blurs in higher-order programming. As a consequence, higher-order program verification faces unique challenges.

The two leading approaches to fully automatic verification of higher-order programs are *higher-order model checking* (HoMC, see Ong, 2006) and *refinement type inference* (RTI, see Vazou et al., 2014). HoMC can capture complex higher-order control flow accurately, but only in the context of data types inhabited by finitely many objects; given such finite base types, it is sound, complete, and automatic for a range of verification problems such as reachability and flow analysis. RTI, on the other hand, supports infinite data types through first-order theory reasoning, but not without a cost; even some finite data type programs are not verifiable using standard techniques of RTI.

Thus, both HoMC and RTI lack the combined reasoning power to reliably verify higher-order programs that arise in practice, but in disparate manners. This gap seems symptomatic of the deeper lack of common foundation for automated higher-order verification in general. Consequently, research into higher-order program verification is fragmented, with no unifying concepts through which to understand the work of different communities.

Higher-order verification has an important empirical component. Given how time-consuming the development of prototype verifiers is, it is unfortunate that there is little opportunity to reuse components from one project to another. For example, verifiers based on type inference are typically tailored to a particular programming language. HoMC, on the other hand, is programming language independent but

awkward to reuse: regular properties of an infinite tree and the usual restrictions to simply typed rules and call-by-name evaluation are at odds with common applications.

We hope that HoCHC, as a shared framework for understanding higher-order program verification, will lead to greater modularity through a separation of concerns: a program can be modelled as a HoCHC instance that can be solved using an SMT-solver, independent of the source language.

1.2.1 Higher-order model checking

Higher-order model checking refers to higher-order extensions of ordinary finite-state model checking. Since a model-checking problem consists of a model $\mathcal{L}$ (a system to be verified) and a property φ, one can conceive higher-order extensions in either argument: the model and the logic.

Typically, higher-order model checking refers to *HoRS model checking*, in which models are extended to *higher-order recursion schemes* (HoRS), building on work by Knapik et al. (2002). However, *HFL model checking* has also gained traction in recent years. In this form of model checking, the property logic is replaced with higher-order modal fixpoint logic (HFL, see Section 1.2.2).

HoRS model checking, hereafter HoMC, concerns the model checking of (potentially infinite) trees generated by higher-order recursion schemes or, equivalently, $\lambda\mathbf{Y}$-calculus. A HoRS of order n is essentially a closed ground-type term definable in the nth-order fragment of simply typed $\lambda\mathbf{Y}$-calculus with recursion and first-order constant symbols as tree constructors. Collapsible pushdown automata of order n are another equi-expressive formalism (Hague et al., 2008).

Recursive schemes are an expressive family of tree generators. The trees generated at orders 0, 1 and 2 correspond to regular trees, algebraic trees (i.e. those generated by context-free tree grammars), and hyperalgebraic trees, respectively (Courcelle, 1978). The trees generated by HoRS correspond to (abstractions of) computation trees of higher-order functional programs.

Ong (2006) has proved that model checking of HoRS is decidable with respect to monadic second-order (MSO) and modal μ-calculus properties. However, the worst-case time complexity is n-EXPTIME in the order n of the HoRS.

Despite a hyperexponential worst-case complexity, significant advances have been

made in (automated) HoMC. The state-of-the-art model checker MoCHi[3] developed by Kobayashi et al. (2011) is a fully automated verification tool for a subset of OCaml that combines HoRS model checking with predicate abstraction (Clarke et al., 2003). The underlying HoRS model checker HorSat2[4] readily scales to HoRS consisting of thousands of rewrite rules (Kobayashi, 2019). However, due to HoMC's inherent lack of support for data types with infinitely many inhabitants, this does not generally scale to large real-life programs. Programs beyond a few hundred lines are already infeasible for tools like MoCHi and its extensions that use refinement type inference to aid predicate abstraction (Sato et al., 2019).

The *HoRS equivalence problem* asks whether two given HoRS generate the same tree. Decidability of this problem is perhaps the best known and most challenging open problem in higher-order model checking. The problem is recursively equivalent to the λY-calculus Böhm tree equivalence problem (Clairambault and Murawski, 2013), which asks whether the Böhm trees of two given λY-terms are equal.

1.2.2 Higher-order fixpoint logic

Higher-order fixpoint logic (HFL) is a higher-order extension of the modal μ-calculus. HFL model checking checks whether a given finite state system satisfies the property described by a given HFL formula. HFL model checking was introduced by Viswanathan and Viswanathan (2004). Although it has received less attention than HoMC, it was picked up by Kobayashi et al. (2017), who have found mutual translations between HoRS and HFL model checking. In fact, various classes of program verification problems can be reduced to HFL model checking (Kobayashi et al., 2018; Watanabe et al., 2019) even more naturally than to HoRS model checking, according to Kobayashi.

This recent interest in HFL model checking is partly due to awareness of the limitations of HoMC. HFL model checking provides a more uniform approach to verification of programs with infinite data types and naturally extends other popular approaches to automated program verification, such as program verification based on constrained Horn clauses (Bjørner et al., 2013a, 2015).

Thus, the motivations for HFL resemble the motivations for HoCHC. HoCHC roughly corresponds to a modality-free, alternation-free fragment of HFL.

[3] https://github.com/hopv/MoCHi
[4] https://github.com/hopv/horsat2

1.2.3 Higher-order constrained Horn clauses

Earlier approaches to the verification of functional programs that were based on constrained Horn clauses required intelligent intervention by the program verifier to circumvent the inherent mismatch between the higher-order nature of the program and the first-order logic in which the Horn clauses are expressed.

Cathcart Burn et al. (2018) have developed a notion of *higher-order constrained Horn clauses* (HoCHC) to bridge this discrepancy between constrained Horn clauses and higher-order program verification. These clauses form a fragment of higher-order logic, a generalisation of first-order Horn clauses with constraints from a first-order background theory.

Consider the following higher-order OCaml program, adapted from Cathcart Burn et al. (2018).

```
let add x y = x + y;;
let rec iter f s k =
  if k <= 0 then
    s
  else
    f k (iter f s (k - 1))
in
    fun (m : int) -> assert (m <= iter add 0 m);;
```

The term `iter add 0 m` in the assert statement denotes the sum of integers from 1 to `m` (for positive `m`; 0 otherwise). This program with assertion can be considered an instance of the safety verification problem; it is *safe* if the assertion is never violated.

Safety verification can be understood in terms of finding an invariant that implies the required property. We can think of a program invariant as an overapproximation of the input-output graphs of the functions defined in the program. If there exists an overapproximation of the graph of `iter add 0` that does not contain a pair `(m,n)` such that `m <= n`, then the assertion is never violated and the program is safe.

Analogous to the first-order case, we characterise this problem of finding invariants logically, by framing it as a satisfiability problem for the following set of higher-order

constrained Horn clauses:

$$\begin{aligned}
\forall x\, y\, z.\, z = x + y &\Rightarrow \mathit{Add}\, x\, y\, z \\
\forall f\, s\, n\, m.\, n \leq 0 \wedge m = s &\Rightarrow \mathit{Iter}\, f\, s\, n\, m \\
\forall f\, s\, n\, m.\, n > 0 \wedge \exists p.\, \mathit{Iter}\, f\, s\, (n-1)\, p \wedge f\, n\, p\, m &\Rightarrow \mathit{Iter}\, f\, s\, n\, m \\
\forall n\, m.\, \mathit{Iter}\, \mathit{Add}\, 0\, n\, m \wedge n > m &\Rightarrow \mathsf{false}
\end{aligned}$$

We have effectively adopted a continuation-passing style (i.e. where functions are relations between inputs and outputs). Constraints, such as $n \leq 0$, are formulas from a first-order background theory. Here, the background theory could be Linear Integer Arithmetic.

A model of this set of clauses is an assignment of relations to the variables *Add* and *Iter* such that the above clauses are satisfied. Note that models are precisely overapproximations of the input-output graphs. For example, the universal relation that takes three natural numbers as arguments is a model of *Add* that strictly overapproximates the function `add` in the original program. The final clause, however, qualifies that an overapproximation must be precise enough to exclude the possibility of relating an input n of $\mathit{Iter}\, \mathit{Add}\, 0$ to a greater output m. We conclude that a model of these clauses is, in fact, an invariant that witnesses the safety of the program. The larger the model, the coarser the invariant.

Existence of canonical solutions. Consider the following model of the above set of clauses, expressed in higher-order logic.

$$\begin{aligned}
&\mathit{Add} \mapsto \lambda x\, y\, z.\, z = x + y \\
&\mathit{Iter} \mapsto \lambda f\, s\, n\, m.\, ((\forall x\, y\, z.\, f\, x\, y\, z \Rightarrow (0 < x \Rightarrow y < z)) \wedge 0 \leq s) \Rightarrow n \leq m
\end{aligned}$$

This model is rather coarse. In particular, under this model the relation described by $\mathit{Iter}\, \mathit{Add}\, (-1)$ relates every pair of integers n, m.

In first-order constrained Horn clauses over the theory of Linear Integer Arithmetic, if a set of clauses has a model, then it has a least model. Many applications and algorithmic properties of first-order constrained Horn clauses are a direct result of this least model property.

Higher-order (constrained or not) Horn clauses do not generally enjoy a least model property, as pointed out by Charalambidis et al. (2013). However, if we restrict the semantic domain to hereditarily monotone elements, such a property does arise for

HoCHC. In fact, a set of HoCHC clauses is satisfiable under the standard interpretation if and only if it is satisfiable under the monotone interpretation (Cathcart Burn et al., 2018).

Compositionality. Whole-program verification is possible in the higher-order setting, because a program usually has a first-order type like $int \to int$. However, it is natural to want to express and verify properties of higher-order functions, say of type $(int \to int) \to int$, in a compositional approach to higher-order program verification; large programs can be broken down into smaller components that can be verified independently, even if the ultimate goal is still a whole-program analysis. Cathcart Burn et al. (2018) prove that higher-order properties definable using refinement types can also be expressed in higher-order constrained Horn clauses.

Solution methods. Deciding whether a given set of higher-order constrained Horn clauses is satisfiable is known as *solving* a HoCHC problem. The solution method for HoCHC proposed by Cathcart Burn et al. (2018) uses a refinement type system to transform a HoCHC problem to a first-order constrained Horn problem. This method is sound but incomplete.

The first sound and refutation-complete solution method for HoCHC relies on a Reynolds-style defunctionalisation to reduce sets of higher-order Horn constraints to first-order Horn constraints (Pham et al., 2018). This allows for the exploitation of the efficiency of SMT solvers for first-order Horn clauses.

1.3 Hoare logic

The field of Hoare logic (Hoare, 1969; Floyd, 1967) is concerned with the annotation of programs with logical formulas and the extraction of logical formulas from annotated programs, for the purpose of verifying program correctness, either manually or mechanically. Hoare logic has become one of the most prolific subfields of program verification due to its compositional, syntax-driven nature that allows for straightforward adaptation to different program languages. An exhaustive survey of the field is available in Apt and Olderog (2019).

Hoare logic was first employed for a simple imperative 'while language' that includes an iteration construct and a fixed number of mutable global variables (a *state*). It has

since been extended to include more complicated constructs such as pointers (Jianhua and Xuandong, 2013), nondeterminism (Dijkstra, 1975; de Bakker, 1980), parallelism (Hoare and Perrott, 1972; Owicki, 1975), and objects (de Boer and Pierik, 2004; Apt et al., 2010).

Typically, the underlying assertion logic is a form of first-order logic. This standard first-order notion of Hoare logic allows for two different correctness statements, with formulas φ, ψ drawn from the assertion language:

- Partial correctness: Do all terminating executions of program Π in a state φ result in a state ψ?
- Total correctness: Do all executions of program Π in a state φ terminate in a state ψ?

Note that $\varphi = \mathsf{true}$ and $\psi = \mathsf{false}$ in the former statement yields the (undecidable) halting problem for the simple 'while language'.

First-order Hoare logic is sound for partial correctness. That is, every *Hoare triple* $\{\varphi\}\,\Pi\,\{\psi\}$ that can be derived in the proof system is valid. Completeness is a different matter. This is due to the nature of the consequence rule, depicted in Figure 1.1.

$$\frac{\vDash \varphi_1 \to \varphi_2 \qquad \{\varphi_2\}\,\Pi\,\{\psi_1\} \qquad \varphi_2 \vDash \psi_1 \to \psi_2}{\{\varphi_1\}\,\Pi\,\{\psi_2\}}$$

Figure 1.1: Consequence rule for first-order Hoare logic.

The first and the third premise of the consequence rules are not Hoare triples but statements of first-order logic. Since first-order logic is undecidable, the validity of these two premises cannot generally be decided. Therefore, any notion of completeness of first-order Hoare logic is relative to an 'oracle' as a source for valid first-order implications; this result is due to Cook (1978).

As a consequence, automated proof generation is notoriously hard for Hoare logic, but this does not preclude the existence of procedures for proving many useful theorems. Automated proof verification is feasible as well.

1.3.1 Hoare logics for higher-order programs

To our knowledge, there have been two adaptations of Hoare logic to a call-by-value functional programming language.

Régis-Gianas and Pottier (2008) have used a higher-order Hoare logic for a language that contains higher-order functions, algebraic data types, and a polymorphic type system in the style of Hindley and Milner, but no state.

Preconditions and postconditions are encoded as a pair in the type of a program. However, due to polymorphism, the logic is not simply typed. Instead, they extend higher-order logic with a form of explicit quantification over types that is embedded within the Calculus of Inductive Constructions (Paulin-Mohring, 1993).

Régis-Gianas and Pottier focus on partial correctness. Because they lift only values—not expressions—to the logical level, they can ignore non-termination issues, and their semantic domains do not contain diverging (non-terminating) elements. This approach is sound for call-by-value semantics but is incomplete.

Another higher-order approach to Hoare logic was taken by Honda et al. (2006), which has since been followed up by Berger et al. (2007); Yoshida et al. (2008); Honda et al. (2014). They annotate call-by-value PCF and an imperative extension thereof with assertions from an extended first-order logic. Their approach is descriptively complete for both partial and total correctness, which implies relative completeness and observational completeness.

Finally, we mention Hoare Type Theory (Nanevski et al., 2006, 2008). It is not strictly a Hoare logic, but it offers type-checking and generation of proof obligations through embedding Hoare-style specifications in types; a Hoare triple $\{P\}\, x : A\, \{Q\}$ is a type that denotes computations with a precondition P and postcondition Q that return a value of type A, where A can be an arbitrarily complex expression. However, Hoare Type Theory is far more ambitious than the higher-order approaches to Hoare logic we mentioned earlier, as Hoare types are a dependently typed version of monads used in Haskell.

1.3.2 Relational Hoare Logic

Benton (2004) has adapted Hoare logic from correctness proofs of programs to correctness proofs of program transformations. His *Relational Hoare Logic* reasons about pairs of imperative programs whose variables are related by some precondition and postcondition.

The role of Hoare triples is subsumed by judgements of the form

$$S_1 \sim S_2 : \Psi \Rightarrow \Phi$$

where S_1, S_2 are programs and Ψ, Φ relations on program states.

Intuitively, such a judgement states that if the relation Ψ holds between the variables of programs S_1 and S_2, then Φ holds between these variables after the (independent) execution of S_1 and S_2. Up to variable renaming—to ensure disjointness of the respective program variables—the above judgement is equivalent to the traditional Hoare triple $\{\Psi\}\, S_1; S_2\, \{\Phi\}$.

Although judgements in Relational Hoare Logic have 'relations' as precondition and postcondition, these relations are assertions that relate the variables of S_1 and S_2, rather than terms of relational type as in the context of higher-order logic (see Chapter 2.2). Thus, Relational Hoare Logic introduces little new from a semantic perspective. Its main result is that the correctness of (first-order imperative) program transformations can be verified in a way analogous to Hoare logic.

1.4 Contributions

Our aim is to explore the adequacy of higher-order constrained Horn clauses (HoCHC) as a unifying framework for higher-order program verification. As mentioned before, we are interested in the following two questions:

(i) Is the fragment sufficiently expressive to capture the problems typically studied by the higher-order program verification community?

(ii) Are there (relatively) complete solution methods (e.g. algorithms that are sound and complete in the presence of an oracle for first-order constraint solving)?

1.4.1 Continuous interpretation for HoCHC

Logic programming is inherently monotone, because models are built up incrementally in the least fixpoint interpretation of a logic program. However, as soon as we start looking at resolution proof methods, monotonicity turns out too liberal for HoCHC; Charalambidis et al. (2013) show that a refutation-complete proof system for a HoCHC-like fragment requires a continuous interpretation, because continuity makes the difference between a countably and uncountably infinite search space, and thus semi-decidability and undecidability.

To this end, we establish a *continuous interpretation* for HoCHC in Chapter 3, in which the semantic domains are restricted to only those functions that are Scott-continuous, i.e. which preserve least upper bounds of directed sets. We prove that

this interpretation of HoCHC is well-defined and well-behaved; it enjoys a least model property, and the associated continuous HoCHC decision problem is equivalent to the monotone decision problem (and hence to the standard problem). We provide a direct translation between the continuous problem and the monotone problem. Finally, we show that there exist HoCHC problem instances where the monotone one-step consequence operator is not continuous, which demonstrates that the continuous interpretation of HoCHC is, in fact, new.

1.4.2 Higher-order model checking via HoCHC

Regarding (i), we choose higher-order model checking (HoMC) as a representative of the class of problems studied in higher-order program verification. As a widely studied approach with extensive literature, HoMC is the basis for state-of-the-art model checker MoCHi (Kobayashi et al., 2011) and is often chosen for comparison with other approaches (see e.g. Kobayashi et al., 2017). On top of that, the meaning of a HoRS is a least fixpoint construction, which aligns it—at least in theory—close to the construction of a least model in HoCHC.

Although the meaning of HoRS and HoCHC are both given by least fixpoints over posets of trees, there is a fundamental disparity due to the orderings on the respective posets. The deterministic HoMC safety problem can be encoded in a decidable fragment of HoCHC (Wagner, 2019, private communication) solely by virtue of only having to inspect finite prefixes of trees. Wider classes of HoMC problems require inspection of the infinitary behaviour of HoRS, including liveness, which is at odds with HoCHC.

To address this disparity, we define a *coinductive* HoCHC framework in Chapter 4 that enjoys a greatest model property under the monotone interpretation. This new framework allows us to reduce decidability of the (open) HoRS equivalence problem (i.e. whether $[\![\mathcal{G}_1]\!] = [\![\mathcal{G}_2]\!]$ for deterministic HoRS $\mathcal{G}_1, \mathcal{G}_2$) to semi-decidability of *coinductive* HoCHC over a decidable background theory.

In particular, we encode a HoRS into a HoCHC logic program over a complete and decidable background theory of trees first introduced by Maher (1988). We anticipate that this HoRS-to-HoCHC encoding can be employed for a wider range of HoMC problems, including checking HoRS against Büchi tree automata and possibly liveness.

Our result on the HoRS equivalence problem has implications for the $\lambda\mathbf{Y}$-calculus Böhm tree equivalence problem (Clairambault and Murawski, 2013), which asks

whether the Böhm trees of two given $\lambda\mathbf{Y}$-terms are equal. Since this problem is recursively equivalent to the HoRS equivalence problem, it can also be reduced to semi-decidability of coinductive HoCHC over Maher's theory.

1.4.3 Solving HoCHC through SLD-resolution

With the logical foundations of HoCHC in place, the next step is looking at sound and refutation-complete solution methods for HoCHC, as per (ii), to bridge the gap between theory and practice.

In Chapter 5, we introduce a sound and refutation-complete resolution proof system for semi-deciding (unsolvable) instances of the HoCHC problem—the first such system at the time of writing, but another has been published since (Ong and Wagner, 2019).

Our system is based on Selective Linear Definite clause (SLD) resolution. This system is known to be sound and refutation-complete for first-order Horn clauses (Kowalski, 1974). We build on work by Charalambidis et al. (2013) on a sound and refutation-complete SLD-resolution proof system for a fragment of higher-order logic that is closely related to HoCHC, namely a positive, existential fragment that corresponds to higher-order Horn clauses without constraints.

Because we are interested in (un)solvability rather than solution sets, we can eliminate higher-order existential quantification from goal terms thanks to our continuous interpretation of HoCHC. This allows us to simplify our proof system compared to Charalambidis et al.

1.4.4 Axiomatic basis for relations as higher-order program invariants

HoCHC is an application of higher-order logic to higher-order program verification that defers to SMT solvers for the first-order background theory. Higher-order Hoare logic is another, related application that annotates programs with formulas of a higher-order logic or extracts such formulas from annotated programs.

Typically, Hoare logic—the line of work initiated by Hoare (1969) and Floyd (1967)—annotates programs with formulas from a first-order logic. However, with HoCHC as a robust logical framework for automated higher-order program verification, we look at the potential of HoCHC, and particularly the higher-order relations that are key to HoCHC, in a higher-order Hoare logic.

Chapter 6 introduces a sound and relatively complete Hoare logic for call-by-value PCF where preconditions are formulas of higher-order logic and postconditions higher-order relations. Useful program logics can be obtained from this general system through restrictions on the syntax. For example, we can restrict ourselves to HoCHC.

Our approach to a Hoare logic for higher-order functions is novel in the sense that our assertion logic is strictly higher-order (cf. Honda et al., 2006) and our proof system relatively complete (cf. Régis-Gianas and Pottier, 2008).

1.4.5 Outline

The definitions of domain theory and higher-order logic used throughout this thesis are presented in the first two sections of Chapter 2. Higher-order constrained Horn clauses (HoCHC) are introduced in Chapter 3, including the new continuous interpretation that we prove to be equivalent to the monotone interpretation. Chapter 4 establishes a connection between HoCHC and higher-order model checking, building on preliminary definitions of higher-order model checking provided in Chapter 2.3. We introduce a coinductive version of HoCHC and an encoding of higher-order recursion schemes into HoCHC logic programs. This allows us to reduce decidability of the (open) higher-order recursion scheme equivalence problem to semi-decidability of coinductive HoCHC over the complete and decidable background theory of trees by Maher (1988). Chapter 5 provides a sound and refutation-complete resolution proof system—based on SLD-resolution—for HoCHC over a semi-decidable background theory. Finally, Chapter 6 takes a related but different approach to higher-order program verification: via higher-order Hoare logic. We introduce a sound and relatively complete proof system for annotating call-by-value PCF programs in which higher-order relations take a prominent position, as they do in HoCHC.

Chapter 2

Preliminaries

2.1 Domain theory

Many of the formalisms in this thesis are rooted in domain theory, which uses *partially ordered sets* (posets) to model domains of computation. Elements of a poset are thought of as information or (partial) results of a computation, such that larger elements—with respect to the partial order—extend the information of smaller elements. Plotkin's Pisa Notes (1983) are among the most authoritative references on domain theory and are referenced throughout this section.

Partially ordered set. A *partially ordered set* or *poset* is a set P equipped with an order—written $\sqsubseteq \subseteq P \times P$ unless we are comparing natural numbers and write $\leq$ instead—that is reflexive, antisymmetric, and transitive. If the order is only reflexive and transitive, we call it a *preorder*.

If $a \sqsubseteq b$ or $b \sqsubseteq a$, then a, b are *comparable*; otherwise a, b are *incomparable*. A poset P such that $a \sqsubseteq b$ implies $a = b$, for all $a, b \in P$, is *discretely* or *trivially* ordered.

Directed (sub)sets. In a poset P, a (nonempty) subset $D \subseteq P$ is *directed* if for every $x, y \in D$ there exists $z \in D$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$. That is, if D contains an upper bound for every pair of elements.

One might think of directed sets as consistent computations, because the information in every pair of elements in D is consistently extended by a larger element in D. In particular, we might be interested in the limit of a directed set, because that is the most general piece of information that describes the information contained in the directed set. Such a limit coincides with the *least upper bound* (lub).

The least upper bound of a directed set does not always exist within the underlying poset. However, if we restrict ourselves to *directed-complete* posets (dcpos), then these lubs do exist by definition. We use $\bigsqcup S$ to denote the lub of subset S. Note that dcpos are not assumed to have a least element, written $\bot$; those that do are called *pointed*.

A poset in which all subsets have both a least upper bound and a *greatest lower bound* (glb), written $\bigsqcap S$ for subset S, is called a *complete lattice*.

Chain. A *chain* C is a totally ordered subset of a poset (i.e. in which $a \sqsubseteq b$ or $b \sqsubseteq a$, for all $a, b \in C$). We distinguish *non-decreasing* (or *ascending*) chains that have a least element from *non-increasing* chains that have a greatest element.

Finite and infinite elements. In a poset A, x *approximates* y (written $x \ll y$, for $x, y \in A$) if there exists $d \in D$ such that $x \sqsubseteq d$ for all directed $D \subseteq A$ with $y \sqsubseteq \bigsqcup D$. We may also say that x is an *approximant* of y. In a poset A, an element $x \in A$ is *finite* (also known as *compact*) if $x \ll x$. Complementarily, x is *infinite* if $x \not\ll x$. We write $\mathit{fin}(A)$ for the set of finite elements of A.

The intuition is that a finite element cannot be obtained as a limit of a directed set in which it does not already occur.

A relevant question is whether we can guarantee that all elements of our domain can be obtained as a limit of approximants. This guarantee would give us a method of approximating infinite elements, which cannot be computed directly.

Continuous functions. For posets A and B, a function $f : A \to B$ is *monotone* if it preserves the order, i.e. $a_1 \sqsubseteq a_2$ implies $f(a_1) \sqsubseteq f(a_2)$ for all $a_1, a_2 \in A$. The function $f : A \to B$ is *(Scott-)continuous* if it additionally preserves least upper bounds of directed subsets $D \subseteq A$ for which this least upper bound exists in A, i.e. $f(\bigsqcup D) = \bigsqcup_{d \in D} f(d)$ for all $D \subseteq A$ such that $\bigsqcup D \in A$.

Continuous functions do not 'invent' things at infinity. Their behaviour on infinite elements—those occurring as the limit of directed sets—is determined by the function values at the smaller finite elements. This makes continuous functions exceedingly suited for modelling the semantics of computer programs, where infinitary behaviour is dictated by finite approximations.

Basis of a poset. For a poset P, the *basis* $B \subseteq P$ is a subset such that, for each $x \in P$, the set of elements in B that approximate x contains a directed set with lub x, or equivalently a non-decreasing chain with lub x.

A poset with a basis consisting of solely finite elements is called *algebraic*. If this basis is countable, we say the poset is *ω-algebraic*.

Given an ω-algebraic dcpo $\langle D, \sqsubseteq \rangle$, we write $\langle \mathsf{B}_D, \sqsubseteq \rangle$ for the poset of finite elements of D, which is precisely the basis of D.

Ideals. An *ideal* of poset P is a subset $X \subseteq P$ that is downward closed (i.e. $x \sqsubseteq y$ and $y \in X$ implies $x \in X$, for all $x, y \in P$) and directed (i.e. for every $x, y \in X$ there exists $z \in X$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$).

Given a countable pointed preorder $\langle P, \sqsubseteq \rangle$, its *ideal completion* $\langle \mathsf{Idl}(P), \subseteq \rangle$ is the set of ideals ordered by inclusion. Let $\downarrow_P : P \to \mathsf{Idl}(P)$ denote the (continuous) function that maps $x \in P$ to the principal ideal $\downarrow x := \{x' \in P \mid x' \sqsubseteq x\}$.

Theorem 2.1 (Plotkin's Pisa Notes, 1983)**.**

(i) For any countable pointed preorder $\langle P, \sqsubseteq \rangle$, $\langle \mathsf{Idl}(P), \subseteq \rangle$ is an ω-algebraic dcpo where P, B_P, and $\downarrow P := \{\downarrow p \mid p \in P\}$ are all isomorphic. Further, for any other ω-algebraic dcpo D and monotone map $f : P \to D$ there exists a unique continuous map $\widehat{f} : \mathsf{Idl}(P) \to D$ such that $f = \widehat{f} \circ \downarrow_P$.

(ii) For any ω-algebraic D, D is isomorphic to $\mathsf{Idl}(\mathsf{B}_D)$. Further, for any countable pointed preorder $\langle P, \sqsubseteq \rangle$, $\mathsf{B}_{\mathsf{Idl}(P)} = P \,/\, \approx$, where $\approx$ is the equivalence associated with the preorder $\sqsubseteq$.

Fixpoint. A *fixpoint* of a function $f : A \to A$ is an element $a \in A$ such that $a = f(a)$. Furthermore, a *prefixed point* of f is an element $a \in A$ such that $f(a) \sqsubseteq a$ (cf. a *postfixed point* $a \in A$ such that $a \sqsubseteq f(a)$).

Theorem 2.2 (Knaster-Tarski Theorem, Tarski, 1955)**.** *Let L be a complete lattice and $f : L \to L$ a monotone function. Then, the set of fixpoints of f in L is also a complete lattice, guaranteeing the existence of a least and a greatest fixpoint of f in L. Furthermore, the least fixpoint coincides with the least prefixed point, and the greatest fixpoint with the greatest postfixed point.*

Theorem 2.3 (Kleene's Fixed-Point Theorem)**.** *For a pointed dcpo L (with least element $\bot$), a continuous function $f : L \to L$ has a least fixpoint, written $\mathsf{lpf}(f)$, that is the least upper bound of the ascending Kleene chain of f:*

$$\mathsf{lpf}(f) = \bigsqcup \{f^n(\bot) \mid n \in \mathbb{N}\}$$

Proposition 2.4. *For every monotone function $f : A \to B$ and directed subset $D \subseteq A$, the set $\{f(d) \mid d \in D\}$ is directed.*

Proof. Because D is directed, for all $x, y \in D$ there exists $z \in D$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$. By monotonicity of f, for all $x, y \in D$ there exists $z \in D$ such that $f(x) \sqsubseteq f(z)$ and $f(y) \sqsubseteq f(z)$. Thus, $\{f(d) \mid d \in D\}$ is a directed subset of B. □

Notation 2.5. While we write $f : A \to B$ for a function f with domain A and codomain B, we use $A \Rightarrow B$ or $[A \Rightarrow B]$ to denote the entire space of functions with domain A and codomain B, so that e.g. $f \in [A \Rightarrow B]$.

Proposition 2.6. *For every directed subset $F \subseteq [A \Rightarrow B]$ and all $a \in A$, the set $\{f(a) \mid f \in F\}$ is directed.*

Proof. Because F is directed, for all $f, g \in F$ there exists $h \in F$ such that $f \sqsubseteq h$ and $g \sqsubseteq h$. By the pointwise order, $f(a) \sqsubseteq h(a)$ and $g(a) \sqsubseteq h(a)$ for all $a \in A$. Thus, $\{f(a) \mid f \in F\}$ is a directed set for each $a \in A$. □

Proposition 2.7. *Let A be a poset and B a complete lattice. For all $F \subseteq [A \Rightarrow B]$, $(\bigsqcap F)(a) = \bigsqcap_{f \in F} f(a)$ for all $a \in A$.*

Proof. Let $h : A \to B$ be defined by $h(a) = \bigsqcap_{f \in F} f(a)$ for all $a \in A$. This greatest lower bound is well-defined because B is a complete lattice. Clearly, h is a lower bound of F.

Now let $g : A \to B$ be an arbitrary lower bound of F. For each $a \in A$, it holds that $g(a)$ is an lower bound of $\{f(a) \mid f \in F\}$, thus $\bigsqcap_{f \in F} f(a) \sqsubseteq g(a)$, which means that $h \sqsubseteq g$. Therefore, $h = \bigsqcap F$. □

We are interested in two cases of the following proposition. First, the proposition holds for directed sets $F \subseteq [A \Rightarrow B]$, because $\{f(a) \mid f \in F\}$ is a directed set for all $a \in A$ by Proposition 2.6, which implies that $\bigsqcup_{f \in F} f(a)$ is an element of B

for all $a \in A$. Second, if B is a complete lattice, then the proposition holds for all $F \subseteq [A \Rightarrow B]$.

Proposition 2.8. *Let A be a poset and B a dcpo. If $\bigsqcup_{f \in F} f(a)$ is defined in B for $F \subseteq [A \Rightarrow B]$ and for all $a \in A$, then $(\bigsqcup F)(a) = \bigsqcup_{f \in F} f(a)$.*

Proof. Let $h : A \to B$ be defined by $h(a) = \bigsqcup_{f \in F} f(a)$ for all $a \in A$. Note that h is an upper bound of F. Now let $g : A \to B$ be an arbitrary upper bound of F. For each $a \in A$, it holds that $g(a)$ is an upper bound of $\{f(a) \mid f \in F\}$, thus $\bigsqcup_{f \in F} f(a) \sqsubseteq g(a)$, which means that $h \sqsubseteq g$. Therefore, $h = \bigsqcup F$. □

Lemma 2.9 (Tennent, 1991)**.** *If C is a dcpo, and $(a_i)_{i \in \mathbb{N}}$ and $(b_j)_{j \in \mathbb{N}}$ are non-decreasing chains in C such that for all $i \in \mathbb{N}$ there exists $j \in \mathbb{N}$ such that $a_i \sqsubseteq b_j$, then $\bigsqcup_{i \in \mathbb{N}} a_i \sqsubseteq \bigsqcup_{j \in \mathbb{N}} b_j$.*

Proposition 2.10. *Let P be a poset and L a dcpo. Let $f : P \times P \to C$ be a monotone function, and let $D \subseteq P$ be a directed set. Then:*

$$\bigsqcup_{x \in D} \bigsqcup_{y \in D} f(x, y) = \bigsqcup_{y \in D} \bigsqcup_{x \in D} f(x, y) = \bigsqcup_{x \in D} f(x, x).$$

Proof. It suffices to show the above for all non-decreasing chains $D = (i)_{i \in \mathbb{N}}$ in P, so let $(i)_{i \in \mathbb{N}}$ be such a chain. By monotonicity, $f(i, 0) \sqsubseteq f(i, 1) \sqsubseteq f(i, 2) \sqsubseteq \ldots$ and $f(0, i) \sqsubseteq f(1, i) \sqsubseteq f(2, i) \sqsubseteq \ldots$ are non-decreasing chains in C for all $i \in \mathbb{N}$. Thus, their respective lubs exist in dcpo C. Let $d_{i\infty}$ and $d_{\infty i}$, resp., be these lubs. This gives us the following to prove:

$$\bigsqcup_{i \in \mathbb{N}} d_{i\infty} = \bigsqcup_{j \in \mathbb{N}} d_{\infty j} = \bigsqcup_{k \in \mathbb{N}} f(k, k) \tag{2.1}$$

First, we need to prove that these lubs exist as well. For the RHS, by monotonicity, $f(k, k) \sqsubseteq f(k + 1, k) \sqsubseteq f(k + 1, k + 1)$ for all $k \in \mathbb{N}$, and thus $f(0, 0) \sqsubseteq f(1, 1) \sqsubseteq \ldots$ is a non-decreasing chain in dcpo C, so that $\bigsqcup_{k \in \mathbb{N}} f(k, k)$ is well-defined.

Pertaining to the first and second lub, resp., Lemma 2.9 gives us $d_{i\infty} \sqsubseteq d_{(i+1)\infty}$ and $d_{\infty j} \sqsubseteq d_{\infty(j+1)}$ for all $i, j \in \mathbb{N}$. This means that $d_{0\infty} \sqsubseteq d_{1\infty} \sqsubseteq \ldots$ and $d_{\infty 0} \sqsubseteq d_{\infty 1} \sqsubseteq \ldots$ are non-decreasing chains whose lubs exist in dcpo C.

The two equalities in (2.1) remain to be proved. Let us consider the first equality. For all $i, j \in \mathbb{N}$, $f(i, j) \sqsubseteq d_{i\infty}$. By Lemma 2.9, $d_{\infty j} = \bigsqcup_{i \in \mathbb{N}} f(i, j) \sqsubseteq \bigsqcup_{i \in \mathbb{N}} d_{i\infty}$. This

proves $\sqsupseteq$. Similarly, we can prove the other direction: for all $i, j \in \mathbb{N}$, $f(i,j) \sqsubseteq d_{\infty j}$. By Lemma 2.9, $d_{i\infty} = \bigsqcup_{j\in\mathbb{N}} f(i,j) \sqsubseteq \bigsqcup_{j\in\mathbb{N}} d_{\infty j}$.

Finally, consider the second equality in (2.1). The direction $\sqsupseteq$ is trivial. For $\sqsubseteq$, it suffices to show that $d_{\infty j} \sqsubseteq \bigsqcup_{k\in\mathbb{N}} f(k,k)$ for all $j \in \mathbb{N}$. It is clear that $f(i,j) \sqsubseteq f(k,k)$ where $k = \max\{i,j\}$, for all $j \in \mathbb{N}$. Thus, by Lemma 2.9, $d_{\infty j} = \bigsqcup_{i\in\mathbb{N}} f(i,j) \sqsubseteq \bigsqcup_{k\in\mathbb{N}} f(k,k)$, which proves the claim. □

The remaining proofs in this chapter concern the properties of continuous function spaces. These results are due to or inspired by Charalambidis et al. (2013) and attributed appropriately.

Proposition 2.11. *For a poset A and a complete lattice L, the continuous function space $A \Rightarrow_c L$ is a complete lattice.*

Proof. Let $F \subseteq [A \Rightarrow_c L]$. We prove that $\bigsqcup F \in [A \Rightarrow_c L]$ and omit the proof of $\bigsqcap F \in [A \Rightarrow_c L]$ because it is symmetrical. Let $h : A \to L$ be defined by $h(a) = \bigsqcup_{f\in F} f(a)$ for all $a \in A$. By Proposition 2.8, $h = \bigsqcup F$.

It remains to show that h is continuous. Let us first show that it is monotone. Consider $x, y \in A$ such that $x \sqsubseteq y$. For all $f \in F$, we have $f(x) \sqsubseteq f(y)$ due to monotonicity of f. Since $\bigsqcup_{f\in F} f(y)$ is an upper bound of $\{f(y) \mid f \in F\}$, it is also an upper bound of $\{f(x) \mid f \in F\}$. Therefore, $\bigsqcup_{f\in F} f(x) \sqsubseteq \bigsqcup_{f\in F} f(y)$ and h is monotone.

Now, for continuity of h, let $D \subseteq A$ be a directed subset. We rely on Proposition 2.10 and continuity of all $f \in F$ for:

$$h\left(\bigsqcup D\right) = \bigsqcup_{f\in F} f\left(\bigsqcup D\right) = \bigsqcup_{f\in F}\bigsqcup_{d\in D} f(d) = \bigsqcup_{d\in D}\bigsqcup_{f\in F} f(d) = \bigsqcup_{d\in D} h(d)$$

Thus, h is continuous, and F has a lub in $A \Rightarrow_c L$. Combined with the symmetrical argument for $\bigsqcap$, this proves that the continuous function space $A \Rightarrow_c L$ is a complete lattice. □

Definition 2.12 (Step function, Plotkin's Pisa Notes, 1983)**.** *Let A be a poset and P a pointed poset with least element $\bot_P$. For all $a \in A$ and $p \in P$, we define a* step function *$(a \searrow p) : A \to P$ by, for all $x \in A$:*

$$(a \searrow p)(x) := \left\{ \begin{array}{ll} p & \text{if } a \sqsubseteq x \\ \bot_P & \text{otherwise} \end{array} \right\}$$

Notation 2.13. Let A be a poset. Given $S \subseteq A$ and $a \in A$, we write

$$S_{[a]} := \{b \in S \mid b \sqsubseteq a\}$$

for the set of elements in S below a.

Lemma 2.14 (Charalambidis et al., 2013)**.** *Let A be a poset and P a pointed poset. Then, for each step function $(a \searrow p)$ and for every monotone $f : A \to P$ it holds that $(a \searrow p) \sqsubseteq f$ if and only if $p \sqsubseteq f(a)$.*

Proposition 2.15 (Charalambidis et al., 2013)**.** *Let L be a complete lattice and assume there exists $S \subseteq fin(L)$ such that for every $x \in L$ it holds that $x = \bigsqcup S_{[x]}$. Then L is an algebraic lattice (ω-algebraic if S is countable) with basis $fin(L) = \{\bigsqcup M \mid M$ is a finite subset of $S\}$.*

The following proposition was adapted from Charalambidis et al. (2013)'s monotone setting to a continuous setting. Their approach to show that $f(a) \sqsubseteq g(a)$ breaks down for infinite $a \in A$ because $(a \searrow c)$ is not necessarily continuous. As a result, the set $T_a = \{(a \searrow c) \mid c \in fin(L)_{[f(a)]}\}$ may not be included in $S_{[f]}$. We can work around this by requiring the poset A to be algebraic.

Proposition 2.16. *For an algebraic poset A and an algebraic lattice L, the continuous function space $A \Rightarrow_c L$ is an algebraic lattice with basis*

$$B = \left\{ \bigsqcup M \;\middle|\; M \text{ is a finite subset of } S \right\},$$

where

$$S = \{(a \searrow c) \mid a \in fin(A), c \in fin(L)\}.$$

Additionally, $A \Rightarrow_c L$ is ω-algebraic if L is ω-algebraic and A is countable.

Proof. Let A be an algebraic poset, and L an algebraic lattice. By Proposition 2.11, the continuous function space $A \Rightarrow_c L$ is a complete lattice. It remains to show that $A \Rightarrow_c L$ has a basis consisting of only finite elements, and that this basis coincides with B.

We show that every continuous function $f \in [A \Rightarrow_c L]$ is the lub of $S_{[f]} = \{s \in S \mid s \sqsubseteq f\}$. Since f is clearly an upper bound of this set, we let g be an arbitrary upper bound of $S_{[f]}$ and show that $f \sqsubseteq g$ by showing that $f(a) \sqsubseteq g(a)$ for all $a \in A$.

Let $a \in A$. Because A is algebraic, there exists a non-decreasing chain $(x_i)_{i\in\mathbb{N}}$ of finite elements in A such that $a = \bigsqcup(x_i)_{i\in\mathbb{N}}$. Let x_i be one of those elements. Let

$$T_{x_i} = \left\{(x_i \searrow c) \;\middle|\; c \in \mathit{fin}(L)_{[f(x_i)]}\right\}.$$

By Lemma 2.14 and definition of $\mathit{fin}(L)_{[f(x_i)]}$, we have for all $h_c \in T_{x_i}$ that $h_c \sqsubseteq f$, and thus T_{x_i} is a subset of $S_{[f]}$. Since g is an upper bound of $S_{[f]}$, it must also be an upper bound of T_{x_i}. Therefore, $h_c \sqsubseteq g$ for each $h_c \in T_{x_i}$. Applying this inequality for x_i we get that $c \sqsubseteq g(x_i)$ for each $c \in \mathit{fin}(L)_{[f(x_i)]}$, and thus $\bigsqcup \mathit{fin}(L)_{[f(x_i)]} \sqsubseteq g(x_i)$. Since L is an algebraic lattice, $f(x_i)$ is the least upper bound of $\mathit{fin}(L)_{[f(x_i)]}$. Thus, $f(x_i) \sqsubseteq g(x_i) \sqsubseteq g(a)$. Because f is continuous, $f(a) = f(\bigsqcup(x_i)_{i\in\mathbb{N}}) = \bigsqcup_{i\in\mathbb{N}} f(x_i)$. Since each $f(x_i) \sqsubseteq g(a)$, this then gives us $f(a) \sqsubseteq g(a)$, as required.

We know from Plotkin's Pisa Notes (1983) that all step functions in S are finite elements of $A \Rightarrow_c L$, so for each $f \in [A \Rightarrow_c L]$ we have $S_{[f]} \subseteq S \subseteq \mathit{fin}(A \Rightarrow_c L)$, which implies that $A \Rightarrow_c L$ is an algebraic lattice. Note that if A is countable and L ω-algebraic, then S is countable and $A \Rightarrow_c L$ is ω-algebraic. By Proposition 2.15, the basis $fin(A \Rightarrow_c L)$ of $A \Rightarrow_c L$ coincides with B, as required. □

Proposition 2.17. *For poset A and algebraic dcpo C, the continuous function space $A \Rightarrow_c C$ is a dcpo.*

Proof. Following the proof of Proposition 2.11, where $F \subseteq [A \Rightarrow_c C]$ is a directed set, and each lub $\bigsqcup_{f\in F} f(a)$ exists due to Proposition 2.6 and C being a dcpo. □

Proposition 2.18. *For algebraic poset A and algebraic dcpo C, the continuous function space $A \Rightarrow_c C$ is an algebraic dcpo with as basis*

$$B = \left\{\bigsqcup M \;\middle|\; M \text{ is a finite subset of } S\right\},$$

where

$$S = \{(a \searrow c) \mid a \in A, c \in \mathit{fin}(L)\}.$$

Additionally, $A \Rightarrow_c L$ is ω-algebraic if L is ω-algebraic and A is countable.

Proof. Following the proof of Proposition 2.16, where $\mathit{fin}(L)_{[f(x_i)]}$ is a directed set for finite element $x_i \in A$, and each lub exists due to Proposition 2.6 and C being a dcpo. □

2.2 Higher-order logic

Higher-order logic is a predicate logic that subsumes first-order logic. It is distinguished from its first-order counterpart through the existence of higher-order quantifiers, which allow quantification over higher-order variables such as sets (second-order variables), sets of sets (third-order variables), etc.

We present higher-order logic as a typed lambda calculus in line with Cathcart Burn et al. (2018). The syntax of higher-order logic is commonly described through types (Church, 1940) that we call *sorts* to distinguish them from refinement types, dependent types, etc.

2.2.1 Syntax

Sorts. Given a sort ι of individuals (e.g. integers or lists), and a boolean sort o of propositions, *sorts* are the simple types that can be generated from:

$$\sigma ::= \iota \mid o \mid \sigma_1 \to \sigma_2$$

The *order* of a sort σ, written $\mathsf{order}(\sigma)$, is defined as:

$$\mathsf{order}(\iota) := 0 \qquad \mathsf{order}(o) := 1 \qquad \mathsf{order}(\sigma_1 \to \sigma_2) := \max\{\mathsf{order}(\sigma_1) + 1, \mathsf{order}(\sigma_2)\}$$

We call a sort or an element inhabiting it *first order* if it has order 0 or 1 (cf. *higher order*).

We usually write sorts as σ or τ, unless we are specifically talking about a *relational sort*, written ρ, which has sort o in tail position and whose higher-order subsorts are also relational:

$$\rho ::= o \mid \iota \to \rho \mid \rho \to \rho$$

Terms. We consider terms of an applied lambda calculus, i.e. those generated by

$$M, N ::= x \mid c \mid M\ N \mid \lambda x : \sigma.\, M$$

for variable x and constant c, where $M : \sigma \to \tau$ and $N : \sigma$. We assume that application associates to the left and the scope of the abstraction extends as far to the right as possible. We identify terms up to α-equivalence.

Sometimes we write sorts as superscripts, particularly if they occur in an application, e.g. $M^{\sigma\to\tau}\ N^{\sigma}$ for $M\ N$. The sort of a bound variable may be omitted if it is irrelevant or clear from the context, i.e. $\lambda x : \sigma.\, M$ becomes $\lambda x.\, M$.

The order of a term is the largest order of any of the variables occurring in it.

Sorting. A *sort environment*, denoted Δ, is a finite sequence of distinct (in the first argument) pairs $x : \sigma$. A term $t : \sigma$ is well-sorted under Δ if the judgement $\Delta \vdash t : \sigma$ can be derived from the following proof rules:

$$\textsf{(SCst)}\ \frac{}{\Delta \vdash c : \sigma_c} \qquad \textsf{(SApp)}\ \frac{\Delta \vdash s : \sigma_1 \to \sigma_2 \qquad \Delta \vdash t : \sigma_1}{\Delta \vdash s\ t : \sigma_2}$$

$$\textsf{(SVar)}\ \frac{}{\Delta_1, x : \sigma, \Delta_2 \vdash x : \sigma} \qquad \textsf{(SAbs)}\ \frac{\Delta_1, x : \sigma_1, \Delta_2 \vdash s : \sigma_2}{\Delta \vdash \lambda x.\, s : \sigma_1 \to \sigma_2}\ x \notin \mathsf{dom}(\Delta)$$

We may write $\Delta(x)$ for the sort of $x \in \mathsf{dom}(\Delta)$, where $\mathsf{dom}(\Delta)$ denotes the domain of Δ.

Formulas. Let Σ be a first-order signature specifying a collection $\mathbb{G}$ of ground sorts (including o) and sorted constants. We consider higher-order *formulas* over Σ by considering well-sorted terms of sort o generated from Σ and the following set LSym of logical constant symbols:

$$\begin{array}{rcl@{\qquad\qquad}rcl} \mathsf{true}, \mathsf{false} & : & o & \neg & : & o \to o \\ \wedge, \vee, \Rightarrow & : & o \to o \to o & \forall_\sigma, \exists_\sigma & : & (\sigma \to o) \to o \end{array}$$

We abbreviate $\exists_\sigma(\lambda x : \sigma.\, M)$ to $\exists x : \sigma.\, M$ or simply $\exists x.\, M$.

Formulas are terms, so the notion of order carries over without modification.

Substitutions. Our substitutions coincide with the usual notion of substitution in lambda calculus. A *substitution* θ is a finite set, written as $[M_1/x_1, \ldots, M_n/x_n]$ or $[x_1 \mapsto M_1, \ldots, x_n \mapsto M_n]$, where each $x_i : \sigma_i$ is a variable and each $M_i : \sigma_i$ a term. We define a term N under substitution θ, written $\theta(N)$, by structural induction:

$$\begin{aligned} \theta(x) &= \begin{cases} M_i & \text{if } x = x_i \in \mathsf{dom}(\theta) \\ x & \text{otherwise} \end{cases} \\ \theta(c) &= c \\ \theta(M\ M') &= \theta(M)\,\theta(M') \\ \theta(\lambda x.\, M) &= \lambda x.\, \theta(M) \end{aligned}$$

The substitution corresponding to the empty set is called the *identity substitution* and will be denoted by ϵ. Substitutions are closed under composition, which is associative. Thus, the set of substitutions with composition forms a monoid. Finally, note that $(M\theta)\gamma = M(\theta\gamma)$, for all terms M and substitutions θ and γ.

2.2.2 Standard semantics

Let A be a Σ-structure, i.e. an interpretation of the sorts and the constants in Σ. We assume that A assigns a nonempty set A_ι to each ground sort $\iota \in \mathbb{G}$ and the complete lattice $\mathbb{B} = \{0 \leq 1\}$ to the boolean sort o of propositions. Furthermore, A assigns an element $A(c : \sigma) \in \mathcal{S}[\![\sigma]\!]$ to every constant $c : \sigma \in \Sigma$, where $\mathcal{S}[\![\sigma]\!]$ is the interpretation of sort σ, as follows.

Interpretation of sorts. Regarding each A_ι as a discrete poset, we define the *sort frame* by induction:

$$\mathcal{S}[\![\iota]\!] := A_\iota \qquad \mathcal{S}[\![o]\!] := \mathbb{B} \qquad \mathcal{S}[\![\sigma_1 \to \sigma_2]\!] := \mathcal{S}[\![\sigma_1]\!] \Rightarrow \mathcal{S}[\![\sigma_2]\!]$$

where $X \Rightarrow Y$ is the full set-theoretic function space between posets X and Y. The lattice $\mathcal{S}[\![o]\!]$ supports the following functions:

$$\begin{array}{rclrcl} \mathsf{or}(b_1)(b_2) & = & \max\{b_1, b_2\} & \mathsf{not}(b) & = & 1 - b \\ \mathsf{and}(b_1)(b_2) & = & \min\{b_1, b_2\} & \mathsf{implies}(b_1)(b_2) & = & \mathsf{or}(\mathsf{not}(b_1))(b_2) \\ \mathsf{exists}_\sigma(f) & = & \max\{f(v) \mid v \in \mathcal{S}[\![\sigma]\!]\} & \mathsf{forall}_\sigma(f) & = & \mathsf{not}(\mathsf{exists}_\sigma(\mathsf{not} \circ f)) \end{array}$$

We extend the order on $\mathcal{S}[\![o]\!]$ to an order on the set $\mathcal{S}[\![\rho]\!]$, for every relational sort ρ. The order $\sqsubseteq_\rho$ is defined pointwise by induction on the structure of ρ. Let σ be either ι or a relational sort ρ'.

- For all $b_1, b_2 \in \mathcal{S}[\![o]\!]$: if $b_1 \leq b_2$ then $b_1 \sqsubseteq_o b_2$
- For all $r_1, r_2 \in \mathcal{S}[\![\sigma \to \rho]\!]$: if $r_1(s) \sqsubseteq_\rho r_2(s)$ for all $s \in \mathcal{S}[\![\sigma]\!]$, then $r_1 \sqsubseteq_{\sigma\to\rho} r_2$.

This ordering determines a complete lattice structure on each $\mathcal{S}[\![\rho]\!]$. We denote the least upper bound (lub) and greatest lower bound by $\bigsqcup_\rho$ and $\bigsqcap_\rho$, respectively, where we omit the superscript for readability.

Interpretation of terms. Sort environment Δ is interpreted by indexed product

$$\mathcal{S}[\![\Delta]\!] := \Pi x \in \mathsf{dom}(\Delta).\, \mathcal{S}[\![\Delta(x)]\!],$$

which is the set of functions that map each $x \in \mathsf{dom}(\Delta)$ to an element of $\mathcal{S}[\![\Delta(x)]\!]$. We refer to the elements of $\mathcal{S}[\![\Delta]\!]$, typically α or β, as *valuations*. The order on $\mathcal{S}[\![\rho]\!]$ extends pointwise to an order on $\mathcal{S}[\![\Delta]\!]$, with $f_1 \sqsubseteq_\Delta f_2$ just if $f_1(x) \sqsubseteq_\sigma f_2(x)$ for all $x : \sigma \in \Delta$.

Note that if Δ contains an element $x : \iota$, then $\alpha \sqsubseteq \beta$ implies $\alpha(x) = \beta(x)$ due to the trivial order on individuals in $\mathcal{S}[\![\iota]\!]$.

The logical symbols from LSym are interpreted according to the functions given earlier. The interpretation of a term $\Delta \vdash M : \sigma$ is a function $\mathcal{S}[\![\Delta \vdash M : \sigma]\!]$, the Σ-structure A left implicit, that belongs to the set $\mathcal{S}[\![\Delta]\!] \Rightarrow \mathcal{S}[\![\sigma]\!]$ and is defined by the following equations.

$$\begin{aligned}
\mathcal{S}[\![\Delta \vdash x : \sigma]\!](\alpha) &= \alpha(x) \\
\mathcal{S}[\![\Delta \vdash c : \sigma]\!](\alpha) &= A(c) \\
\mathcal{S}[\![\Delta \vdash M\ N : \sigma_2]\!](\alpha) &= \mathcal{S}[\![\Delta \vdash M : \sigma_1 \to \sigma_2]\!](\alpha)\big(\mathcal{S}[\![\Delta \vdash N : \sigma_1]\!](\alpha)\big) \\
\mathcal{S}[\![\Delta \vdash \lambda x : \sigma_1.\, M : \sigma_1 \to \sigma_2]\!](\alpha) &= \lambda v \in \mathcal{S}[\![\sigma_1]\!].\, \mathcal{S}[\![\Delta, x : \sigma_1 \vdash M : \sigma_2]\!](\alpha[x \mapsto v])
\end{aligned}$$

We may write $\mathcal{S}[\![M]\!]$ for $\mathcal{S}[\![\Delta \vdash M : \sigma]\!]$ when the judgement $\Delta \vdash M : \sigma$ is clear from the context.

Satisfaction For a Σ-structure A, a formula $\Delta \vdash M : o$ and a valuation $\alpha \in \mathcal{S}[\![\Delta]\!]$, we say that $\langle A, \alpha \rangle$ *satisfies* M and write $A, \alpha \vDash M$ just if $\mathcal{S}[\![\Delta \vdash M : o]\!](\alpha) = 1$. We define entailment $M \vDash N$ between two formulas M and N in terms of satisfaction as usual. That is, for formulas M and N, $M \vDash N$ if $A, \alpha \vDash M : o$ implies $A, \alpha \vDash N : o$ for all $\alpha \in \mathcal{S}[\![\Delta]\!]$.

2.2.3 Henkin semantics

In addition to the *standard semantics* of higher-order logic and fragments thereof, the well-established *Henkin semantics* are frequently employed (Henkin, 1950). Rather than interpreting higher-order sorts as full function spaces, Henkin semantics fixes a subspace for each higher-order sort.

Henkin semantics are less rich but enjoy better algorithmic properties than the standard semantics of HoL for some choices of the semantic domains. Note that the standard semantics are an instance of Henkin semantics.

Henkin frame. A *Henkin frame* $\mathcal{F}$ is a family of mutually disjoint domains, one for each sort, that satisfies the following criteria:

(i) $\mathcal{F}[\![\iota]\!]$ is an arbitrary set of individuals

(ii) $\mathcal{F}[\![o]\!] = \mathbb{B}$

(iii) $\mathcal{F}[\![\sigma_1 \to \sigma_2]\!] \subseteq [\mathcal{F}[\![\sigma_1]\!] \Rightarrow \mathcal{F}[\![\sigma_2]\!]]$, for all sorts σ_1, σ_2

(iv) $\mathsf{and}, \mathsf{or} \in \mathcal{F}[\![o \to o \to o]\!]$

(v) $\mathsf{fexists}_\sigma \in \mathcal{F}[\![(\sigma \to o) \to o]\!]$ for all relational or individual sorts σ

where $\mathsf{fexists}_\sigma(f) := \max\{f(v) \mid v \in \mathcal{F}[\![\sigma]\!]\}$ for all $f \in \mathcal{F}[\![\sigma \to o]\!]$, and other logical constants are defined as in the previous subsection.

Let A be a Σ-structure. Analogous to the standard semantics, a Henkin frame $\mathcal{F}$ interprets sort environment by indexed product

$$\mathcal{F}[\![\Delta]\!] := \Pi x \in \mathsf{dom}(\Delta).\, \mathcal{F}[\![\Delta(x)]\!],$$

which is the set of ($\mathcal{F}$-)valuations of the variables in Δ. Similar to the standard semantics, the order on each $\mathcal{F}[\![\rho]\!]$ extends pointwise to an order on $\mathcal{F}[\![\Delta]\!]$, with $f_1 \sqsubseteq_\Delta f_2$ just if $f_1(x) \sqsubseteq_\sigma f_2(x)$ for all $x : \sigma \in \Delta$, where the subscripts are often omitted.

We require the frame $\mathcal{F}$ to contain enough elements so that any well-formed formula can be evaluated, i.e. the domains of function types are rich enough to satisfy *comprehension*: for each signature Σ, sort environment Δ, Σ-structure A over $\mathcal{F}$, valuation $\alpha \in \mathcal{F}[\![\Delta]\!]$, positive existential Σ-term $\Delta \vdash \lambda x.\, M$, and $r \in \mathcal{F}[\![\Delta(x)]\!]$,

$$\mathcal{F}[\![\lambda x.\, M]\!](\alpha)(r) = \mathcal{F}[\![M]\!](\alpha[x \mapsto r]).$$

Normally, a Henkin frame requires this equality to hold for all Σ-terms. However, positive existential terms suffice for our purpose, which are those terms that do not contain negation, implication, or universal quantification. This nonstandard notion of a Henkin frame is also used by Ong and Wagner (2019).

Interpretation of terms. The interpretation of a term $\Delta \vdash M : \sigma$ is a function $\mathcal{F}[\![\Delta \vdash M : \sigma]\!]$, Σ-structure A left implicit, that belongs to the set $\mathcal{F}[\![\Delta]\!] \Rightarrow \mathcal{F}[\![\sigma]\!]$ and is defined by the following equations.

$$\begin{aligned}
\mathcal{F}[\![\Delta \vdash x : \sigma]\!](\alpha) &= \alpha(x) \\
\mathcal{F}[\![\Delta \vdash c : \sigma]\!](\alpha) &= A(c) \\
\mathcal{F}[\![\Delta \vdash M\ N : \sigma_2]\!](\alpha) &= \mathcal{F}[\![\Delta \vdash M : \sigma_1 \to \sigma_2]\!](\alpha)\big(\mathcal{F}[\![\Delta \vdash N : \sigma_1]\!](\alpha)\big) \\
\mathcal{F}[\![\Delta \vdash \lambda x : \sigma_1.\, M : \sigma_1 \to \sigma_2]\!](\alpha) &= \lambda v \in \mathcal{F}[\![\sigma_1]\!].\, \mathcal{F}[\![\Delta, x : \sigma_1 \vdash M : \sigma_2]\!](\alpha[x \mapsto v])
\end{aligned}$$

We will write $\mathcal{F}[\![M]\!]$ for $\mathcal{F}[\![\Delta \vdash M : \sigma]\!]$ whenever the judgement $\Delta \vdash M : \sigma$ is clear from the context.

The monotone and continuous Henkin frames. In this thesis we use two Henkin frames, namely monotone frame $\mathcal{M}$ and continuous frame $\mathcal{C}$, defined by:

$$\mathcal{M}[\![\sigma_1 \to \sigma_2]\!] := \mathcal{M}[\![\sigma_1]\!] \Rightarrow_m \mathcal{M}[\![\sigma_2]\!] \qquad \mathcal{C}[\![\sigma_1 \to \sigma_2]\!] := \mathcal{C}[\![\sigma_1]\!] \Rightarrow_c \mathcal{C}[\![\sigma_2]\!]$$

These are the monotone and continuous function spaces, respectively. It is easy to show that $\mathcal{M}[\![\rho]\!]$ and $\mathcal{C}[\![\rho]\!]$ are complete lattices—like $\mathcal{S}[\![\rho]\!]$—and that their respective orderings extend to complete lattices $\mathcal{M}[\![\Delta]\!]$ and $\mathcal{C}[\![\Delta]\!]$ of *valuations*. Cathcart Burn et al. (2018) prove this for the monotone frame. For the continuous frame, it follows from Proposition 2.16 and a straightforward inductive argument.

2.3 Higher-order model checking

Terms over ι. Like in higher-order logic (Section 2.2), we consider terms of an applied lambda calculus, but we restrict ourselves to sorts generated by

$$\sigma ::= \iota \mid \sigma_1 \to \sigma_2,$$

where the sort ι of individuals corresponds to trees. We may refer to a sort σ of this form as a *sort over* ι and to a term with subsorts over ι as a *term over* ι. Whenever we consider higher-order recursion schemes, we assume sorts and terms over ι.

An *applicative term* over a set Θ is a term is constructed from Θ using only the application rule; it does not contain lambda-abstractions and is generated by:

$$t ::= s \in \Theta \mid t_1\, t_2$$

Higher-order recursion scheme. A *higher-order recursion scheme* (HoRS) is a simply sorted (typed) generator of possibly infinite ranked trees over a finite alphabet.

A HoRS is defined as a quadruple $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$ where:

- Σ is a finite set of ranked *terminal symbols*. That is, each $f \in \Sigma$ has an arity $\mathrm{ar}(f) \geq 0$ such that $f : \underbrace{\iota \to \cdots \to \iota \to}_{\mathrm{ar}(f)} \iota$.
- $\mathcal{N}$ is a finite set of sorted *nonterminal symbols*.
- $\mathcal{R}$ is a finite set of well-formed *rewrite rules*, each of the form

  $$F = \lambda x_1 \ldots x_m.\, t$$

  where $F : \sigma_1 \to \cdots \to \sigma_m \to \iota$ is a nonterminal symbol in $\mathcal{N}$, each $x_i : \sigma_i$ is distinct, and t is an applicative term over $\Sigma \cup \mathcal{N} \cup \{x_1, \ldots, x_m\}$.

- $S : \iota \in \mathcal{N}$ is the distinguished *start symbol*.

We assume that each x_i is drawn from a suitably large but finite set V_{RS} of *recursion scheme variables*. We often refer to (non)terminal symbols as *(non)terminals*, and may write $\Sigma(f)$ to denote the sort σ of $f : \sigma \in \Sigma$, and $\mathcal{N}(F)$ for the sort σ of $F : \sigma \in \mathcal{N}$.

The nonterminal symbols in $\mathcal{N}$ are the productive symbols in the grammar; they may be rewritten in a reduction step (see below), as opposed to the terminal symbols in Σ that are tree constructors that remain unchanged once produced.

A HoRS is *deterministic* if for every nonterminal $F : \sigma \in \mathcal{N}$ there exists exactly one rewrite rule in $\mathcal{R}$. In this case, $\mathcal{R}$ is a function, so that we can write $\mathcal{R}(F : \sigma)$ or just $\mathcal{R}(F)$ for the RHS of the unique rewrite rule of $F : \sigma \in \mathcal{N}$. Henceforth, we assume that HoRS are deterministic.

Order of a HoRS. As usual, the *order* of a term is defined as the type-theoretic order of its sort. We choose the order of ground sort ι to be 0, as is customary in higher-order model checking:

$$\mathsf{order}(\iota) := 0 \qquad \mathsf{order}(\sigma_1 \to \sigma_2) := \max\{\mathsf{order}(\sigma_1) + 1, \mathsf{order}(\sigma_2)\}$$

The order of a HoRS $\mathcal{G}$, written $\mathsf{order}(\mathcal{G})$, is given by the highest order of its nonterminals. Note that this definition of order is consistent with the one we have seen for higher-order logic.

Meaning of a (deterministic) HoRS. The *meaning* (or *value tree*) of a HoRS is a possibly infinite applicative term over $\Sigma \cup \{\bot\}$ or, equivalently, a $(\Sigma \cup \{\bot\})$-labelled tree. It is obtained by unfolding the rewrite rules of $\mathcal{G}$ ad infinitum, replacing formal by actual parameters each time, beginning with start symbol S.

For a HoRS $\mathcal{G}$, we formally define the *reduction relation* $\to_{\mathcal{G}}$ by induction over the following rules:

$$\frac{\mathcal{R}(F) = \lambda x_1 \ldots x_m.\, t}{F\, t_1 \ldots t_m \to_{\mathcal{G}} t\,[t_1/x_1, \ldots, t_m/x_m]} \qquad \frac{t \to_{\mathcal{G}} t'}{s\, t \to_{\mathcal{G}} s\, t'} \qquad \frac{t \to_{\mathcal{G}} t'}{t\, s \to_{\mathcal{G}} t'\, s}$$

The reflexive transitive closure of $\to_{\mathcal{G}}$ is denoted by $\to_{\mathcal{G}}^*$, using the Kleene star * that stands for 0 or more (but a finite number of) iterations. The subscript $\mathcal{G}$ is often omitted when the scheme is clear from the context.

A Σ-labelled tree is a partial function s from $\{1,\ldots,M\}^*$ to Σ, where M is the maximal arity of any $f \in \Sigma$, such that $\mathsf{dom}(s)$ is prefix-closed; we assume that s is ranked: if $s(w) = a$ and $\mathrm{ar}(a) = r$, then $\{i \mid wi \in \mathsf{dom}(s)\} = \{1,\ldots,r\}$.

Given a term t, we define a $(\Sigma \cup \{\bot\})$-labelled tree $t^\bot$ by:

$$t^\bot := \begin{cases} f & \text{if } t = f \in \Sigma \\ t_1^\bot\, t_2^\bot & \text{if } t = t_1\, t_2 \text{ and } t_1^\bot \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

This operation substitutes $\bot$ for terms headed by nonterminals, which are essentially 'unfinished' computations. For example, $(f\,(F\,a)\,b)^\bot = f\,\bot\,b$.

Let $\sqsubseteq$ be the least partial order on $\Sigma \cup \{\bot\}$, written $\Sigma_\bot$, such that $\forall a \in \Sigma.\, \bot \sqsubseteq a$. We extend $\sqsubseteq$ to a partial order on trees by:

$$s \sqsubseteq t := \forall w \in \mathsf{dom}(s).\, (w \in \mathsf{dom}(t) \wedge s(w) \sqsubseteq t(w)).$$

E.g. $\bot \sqsubseteq f\,\bot\,\bot \sqsubseteq f\,\bot\,b \sqsubseteq f\,a\,b$. Finally, we define the meaning of $\mathcal{G}$, the value tree of $\mathcal{G}$, by

$$[\![\mathcal{G}]\!] := \bigsqcup \{t^\bot \mid S \to_\mathcal{G}^* t\}.$$

By construction, $[\![\mathcal{G}]\!]$ is a possibly infinite $(\Sigma \cup \{\bot\})$-labelled tree.

Note that this least upper bound exists due to $t_1 \to_\mathcal{G} t_2$ implying $t_1^\bot \sqsubseteq t_2^\bot$ and the set of trees ordered with respect to $\sqsubseteq$ being a dcpo, as a consequence of Theorem 2.1. We formalise this in Lemma 4.8.

Set of trees. Let $\Sigma_\bot = \{\Sigma \cup \bot\}$. The set of all finite and infinite $\Sigma_\bot$-labelled trees is denoted by $\mathcal{T}_{\Sigma_\bot}$. Clearly, $[\![\mathcal{G}]\!] \in \mathcal{T}_{\Sigma_\bot}$ for every HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$.

$\bot$-freeness of a HoRS. We say that a HoRS $\mathcal{G}$ (or its value tree $[\![\mathcal{G}]\!]$) is $\bot$-free if $\bot$ does not occur in $[\![\mathcal{G}]\!]$. To be precise, this happens when there does not exist $w \in \mathsf{dom}([\![\mathcal{G}]\!])$ such that $[\![\mathcal{G}]\!](w) = \bot$. For example, the HoRS given by $S = f\,(F\,a)$, $F = \lambda x.\, G\,x$, and $G = \lambda x.\, F\,x$ is not $\bot$-free, because the reduction sequence

$$S \to f\,(F\,a) \to f\,(G\,a) \to f\,(F\,a) \to \ldots$$

repeats ad infinitum, so that $[\![\mathcal{G}]\!] = f\,\bot$.

Chapter 3

Higher-order constrained Horn clauses

Higher-order constrained Horn clauses (HoCHC) are a fragment of higher-order logic that extends the first-order constrained Horn clauses known from logic programming, which are Horn clauses of first-order logic containing constraints expressed in some suitable language (and interpreted in a suitable background theory).

We present the HoCHC fragment as a sorted (typed) lambda calculus. We largely follow the notations of Cathcart Burn et al. (2018), building on the definitions of higher-order logic introduced in Chapter 2.2. The definitions in the first three sections of this chapter are largely due to Cathcart Burn et al. (2018).

Background theory. Fix a first-order language over a first-order signature Σ, consisting of distinguished subsets of first-order terms Tm and first-order formulas $\varphi \in Fm$, and a first-order theory Th in which to interpret those formulas. We refer to this first-order language as the *constraint language*, and Th as the *background theory*.

Atoms and constraints. An *atom* is an applicative formula $X\ M_1 \cdots M_k$ in which X is a relational variable and each M_i a term. A *constraint*, φ, is just a formula from the constraint language. For technical convenience, we assume that atoms do not contain any constants (including logical constants), and constraints do not contain any relational variables.

3.1 Constrained definite clauses

Constrained Horn clauses. Let Δ be a sorting of relational variables. The *constrained goal formulas* over Δ, typically G, and the *constrained definite formulas* over Δ, typically D, are defined by induction over

$$G ::= M \mid \varphi \mid G \wedge G \mid G \vee G \mid \exists x : \sigma.\, G$$
$$D ::= \mathsf{true} \mid \forall x : \sigma.\, D \mid D \wedge D \mid G \Rightarrow X\ \overline{x}$$

where σ is an individual or a relational sort, M is an atom—its head variable not required to occur in $\mathsf{dom}(\Delta)$, φ a constraint, X a relational symbol inside $\mathsf{dom}(\Delta)$, and $\overline{x} = x_1 \cdots x_n$ a sequence of pairwise distinct variables.

It will often be convenient to view a constrained definite formula equivalently as a conjunction of (constrained) *definite clauses*, which are those definite formulas with shape: $\forall \overline{x}.\, G \Rightarrow X\ \overline{x}$.

Problem. A *(higher-order) constrained Horn clause problem* is given by a tuple $\langle \Delta, D, G \rangle$ in which:

- Δ is a sorting of relational variables.
- $\Delta \vdash D : o$ is a constrained definite formula over Δ.
- $\Delta \vdash G : o$ is a constrained goal formula over Δ.

We say that such a problem is *solvable* just if, for all models A of the background theory Th, there exists a valuation α of the variables in Δ such that $A, \alpha \vDash D$, and yet $A, \alpha \nvDash G$.

In practice, we may restrict ourselves to a single (standard) model of the background theory, e.g. the standard model of Linear Integer Arithmetic.

3.2 Constrained logic programs

The HoCHC fragment and the associated decision problem can be equivalently defined in terms of logic programs. The logic program definition brings us closer to related work on the extensional semantics of higher-order logic program by e.g. Wadge (1991) and Charalambidis et al. (2013). Furthermore, it is a natural choice of definition if we restrict the semantics to a Henkin frame (defined in Chapter 2.2.3) consisting of,

say, monotone relations, as it eliminates the need for the (non-monotone) implication operator.

Goal terms. The class of well-sorted *goal terms* $\Delta \vdash G : \rho$ is given by the sorting judgements defined by the rules below, where σ stands for an individual or a relational sort. The constrained goal formulas, defined in the previous section, are a propositional sorted subset of the goal terms.

$$\textsf{(GCst)}\ \frac{}{\Delta \vdash c : \rho_c}\, c \in \{\wedge, \vee, \exists_\iota\} \cup \{\exists_\rho \mid \rho\} \qquad \textsf{(GVar)}\ \frac{}{\Delta_1, x : \rho, \Delta_2 \vdash x : \rho}$$

$$\textsf{(GConstr)}\ \frac{}{\Delta \vdash \varphi : o}\, \Delta \vdash \varphi : o \in Fm \quad \textsf{(GAppR)}\ \frac{\Delta \vdash G : \rho_1 \to \rho_2 \qquad \Delta \vdash H : \rho_1}{\Delta \vdash G\, H : \rho_2}$$

$$\textsf{(GAppl)}\ \frac{\Delta \vdash G : \iota \to \rho}{\Delta \vdash G\, N : \rho}\, \Delta \vdash N : \iota \in Tm \quad \textsf{(GAbs)}\ \frac{\Delta, x : \sigma \vdash G : \rho}{\Delta \vdash \lambda x.\, G : \sigma \to \rho}\, x : \sigma \notin \Delta$$

Logic programs. A higher-order constrained *logic program*, P, over a relational sort environment $\Delta = x_1 : \rho_1, \ldots, x_m : \rho_m$ is just a finite system of (mutual) recursive definitions of shape:

$$x_1 : \rho_1 = G_1, \quad \ldots, \quad x_m : \rho_m = G_m$$

Such a program is well sorted when, for each $1 \leq i \leq m$, $\Delta \vdash G_i : \rho_i$. Since each x_i is distinct, we may regard a program P as a finite map from variables to terms such that $P(x_i) = G_i$. We will write $\vdash P : \Delta$ to abbreviate that P is a well-sorted program over Δ.

The logic program of a definite formula. Every definite formula D gives rise to a logic program if we collapse clauses that share the same head $X\,\overline{x}$, take the disjunction of their bodies, and view the resulting expression as a recursive definition of X.

The formulation as logic program is a natural object to which to assign a monotone interpretation, since we have eliminated implication—which does not act monotonically in its first argument—in favour of definitional equality.

Consider a definite formula $\Delta \vdash D : o$. We assume, without loss of generality, that D has the shape

$$\forall \overline{x_{r_1}}.\, G_1 \Rightarrow X_{r_1}\, \overline{x_{r_1}} \quad \wedge \quad \cdots \quad \wedge \quad \forall \overline{x_{r_\ell}}.\, G_\ell \Rightarrow X_{r_\ell}\, \overline{x_{r_\ell}}$$

over a sort environment $\Delta = \{X_1 : \rho_1, \ldots, X_k : \rho_k\}$, i.e. $\{1, \ldots, k\} = \{r_1, \ldots, r_\ell\}$. We construct a program over Δ, called the *logic program of* D and denoted P_D, as

$$X_1 = \lambda \overline{x_1}.\, G'_1, \quad \ldots, \quad X_k = \lambda \overline{x_k}.\, G'_k$$

where $G'_j = \bigvee\{G_i \mid r_i = j\}$ such that $\{G_i \mid r_i = j\}$ are exactly the bodies of all the definite clauses in D whose heads are X_j. The fact that $\vdash P_D : \Delta$ follows immediately from the well-sortedness of D.

Problem. A *(higher-order) constrained Horn clause problem* is given by a tuple $\langle \Delta, P_D, G \rangle$ in which:

- Δ is a sorting of relational variables.
- $\vdash P_D : \Delta$ is a logic program of a constrained definite formula D over Δ.
- $\Delta \vdash G : o$ is a constrained goal formula over Δ.

We say that such a problem is *solvable* just if, for all models A of the background theory Th, there exists a valuation α of the variables in Δ such that $A, \alpha \vDash P_D$, and yet $A, \alpha \nvDash G$. Again, we may restrict ourselves to a single (standard) model of the background theory.

3.3 Interpretation

A crucial part of the HoCHC decision problem is still missing: the interpretation of terms and formulas. As Cathcart Burn et al. (2018) point out, higher-order constrained definite formulas do not necessarily possess least models under the *standard semantics* (written $\mathcal{S}$, see Chapter 2.2.2). However, Cathcart Burn et al. also show that a monotone interpretation does yield a least model property. In such an interpretation, higher-order sorts are interpreted as the monotone function space, rather than the full function space. Ong and Wagner (2019) recently showed that the monotone interpretation for HoCHC is a Henkin frame (Chapter 2.2.3). Furthermore, we know from both Cathcart Burn et al. and Ong and Wagner that the HoCHC decision problem under the standard semantics is equivalent to the HoCHC decision problem under the monotone semantics, written $\mathcal{M}$.

In Section 3.4, we introduce a *continuous semantics*, written $\mathcal{C}$, in which the semantic domain is restricted even further to only those functions that are (Scott-)continuous. As the reader will see, this interpretation of HoCHC enjoys a least model property

as well and is, in fact, equivalent—though not identical—to the monotone interpretation (and hence to the standard interpretation). This result has since been subsumed by Ong and Wagner. However, our proofs of equivalence also provide a direct translation between models in one interpretation and models in another. On top of this, we show that the continuous interpretation is indeed a restriction compared to the monotone interpretation; there exist HoCHC problem instances where the monotone one-step consequence operator is not continuous, unlike in first-order Horn clauses.

In the remainder of this section we fix a Henkin frame $\mathcal{F}$, as per Chapter 2.2.3. In particular, let $\mathcal{F} \in \{\mathcal{S}, \mathcal{M}, \mathcal{C}\}$.

$\mathcal{F}$-sort frame. We define the *$\mathcal{F}$-sort frame* by induction over the background theory A; we inherit this definition from HoL. Recall that A_ι is a set, regarded as a discretely ordered poset, so that

$$\mathcal{F}[\![\iota]\!] := A_\iota \qquad \mathcal{F}[\![o]\!] := \mathbb{B} \qquad \mathcal{F}[\![\sigma_1 \to \sigma_2]\!] := \mathcal{F}[\![\sigma_1]\!] \Rightarrow_f \mathcal{F}[\![\sigma_2]\!],$$

where $X \Rightarrow_f Y$ is a subset of the function space between posets X and Y as dictated by the Henkin frame:

$$X \Rightarrow_f Y := \begin{cases} [X \Rightarrow Y] & \text{if } \mathcal{F} = \mathcal{S} \\ [X \Rightarrow_m Y] & \text{if } \mathcal{F} = \mathcal{M} \\ [X \Rightarrow_c Y] & \text{if } \mathcal{F} = \mathcal{C} \end{cases}$$

I.e. for $\mathcal{F} = \mathcal{S}$ this is the full function space, for $\mathcal{F} = \mathcal{M}$ the set of all functions $g \in [X \Rightarrow Y]$ that have the property $x_1 \sqsubseteq x_2$ implies $g(x_1) \sqsubseteq g(x_2)$ for all $x_1, x_2 \in X$, and for $\mathcal{F} = \mathcal{C}$ the set of all functions $g \in [X \Rightarrow Y]$ such that directed set $D \subseteq X$ implies $g(\bigsqcup D) = \bigsqcup_{d \in D} g(d)$.

As we have seen before in HoL, the lattice structure on $\mathbb{B}$ extends to a pointwise ordering $\sqsubseteq_\rho$, or just $\sqsubseteq$, on each relational semantic domain $\mathcal{F}[\![\rho]\!]$. It is worth noting that $X \Rightarrow_f Y$ coincides with the full function space if X is a discrete poset A_ι.

Let us consider what it means for a relation, which is a propositional function, to be monotone. A relation r is an element of $X_1 \Rightarrow_m \cdots \Rightarrow_m X_k \Rightarrow_m \mathbb{B}$ just if it is *upward closed*: whenever r is true of $x_1, \ldots, x_k$ and $x_i \sqsubseteq x'_i$ for all $i \in [k]$, then r must also be true of $x'_1, \ldots, x'_k$. In particular, when $r \in [X \Rightarrow_m \mathbb{B}]$, then r can be thought of as an upward closed set of elements of X.

Interpretation of goal terms. Recall that HoCHC is a fragment of HoL. As such, its interpretation is inherited from HoL. Nonetheless, we provide the explicit denotational semantics of goal terms here.

$$\begin{aligned}
\mathcal{F}[\![\Delta \vdash x : \rho]\!](\alpha) &= \alpha(x) \\
\mathcal{F}[\![\Delta \vdash \varphi : o]\!](\alpha) &= \mathcal{S}[\![\Delta \vdash \varphi : o]\!](\alpha) \\
\mathcal{F}[\![\Delta \vdash G\ H : \rho_2]\!](\alpha) &= \mathcal{F}[\![\Delta \vdash G : \rho_1 \to \rho_2]\!](\alpha)(\mathcal{F}[\![\Delta \vdash H : \sigma_1]\!](\alpha)) \\
\mathcal{F}[\![\Delta \vdash G\ N : \rho]\!](\alpha) &= \mathcal{F}[\![\Delta \vdash G : \iota \to \rho]\!](\alpha)(\mathcal{S}[\![\Delta \vdash N : \iota]\!](\alpha)) \\
\mathcal{F}[\![\Delta \vdash \lambda x : \sigma.\, G : \rho]\!](\alpha) &= \lambda x' \in \mathcal{F}[\![\sigma]\!].\, \mathcal{F}[\![\Delta, x : \sigma \vdash G : \rho]\!](\alpha[x \mapsto x']) \\
\mathcal{F}[\![\Delta \vdash \wedge : o \to o \to o]\!](\alpha) &= \mathsf{and} \\
\mathcal{F}[\![\Delta \vdash \vee : o \to o \to o]\!](\alpha) &= \mathsf{or} \\
\mathcal{F}[\![\Delta \vdash \exists_\sigma : (\sigma \to o) \to o]\!](\alpha) &= \mathsf{fexists}_\sigma
\end{aligned}$$

Quantification is relativised to the sort frame: $\mathsf{fexists}_\sigma(r) = \max\{r(d) \mid d \in \mathcal{F}[\![\sigma]\!]\}$, where σ is either ι or a relational sort. As usual, the fixed interpretation A of the background theory is left implicit; we fall back on this interpretation for constraint formulas and terms of sort ι, as demonstrated by the invocation of $\mathcal{S}$ in those cases.

We define a function $T^{\mathcal{F}}_{P:\Delta} : \mathcal{F}[\![\Delta]\!] \to \mathcal{F}[\![\Delta]\!]$ on semantic environments, also called the *one-step consequence operator* in logic programming literature, by interpreting the RHSs of the HoCHC logic program: $T^{\mathcal{F}}_{P:\Delta}(\alpha)(x) = \mathcal{F}[\![\Delta \vdash P(x) : \Delta(x)]\!](\alpha)$. A prefixed point of $T^{\mathcal{F}}_{P:\Delta}$ is called a *model* of the program P.

Cathcart Burn et al. (2018) establish a standard and a monotone interpretation for HoCHC. Pertaining to the latter, they show that the semantics are well-defined, i.e. $\mathcal{M}[\![\Delta \vdash G : \rho]\!] \in \mathcal{M}[\![\Delta]\!] \Rightarrow_m \mathcal{M}[\![\rho]\!]$ for all $\Delta \vdash G : \rho$, and that the one-step consequence operator $T^{\mathcal{M}}_{P:\Delta}$ is monotone. Furthermore, it follows from the Knaster-Tarski Theorem 2.2 that, unlike the one-step consequence operator arising from the standard interpretation, $T^{\mathcal{M}}_{P:\Delta}$ has a least prefixed point—or least fixpoint. Consequently, logic programs $\vdash P : \Delta$ have a canonical least model under the monotone interpretation.

In the sequel, we introduce the *continuous interpretation* and establish similar results.

3.4 Continuous interpretation

We define the *continuous sort frame* by induction over the background theory A:

$$\mathcal{C}[\![\iota]\!] := A_\iota \qquad \mathcal{C}[\![o]\!] := \mathbb{B} \qquad \mathcal{C}[\![\rho_1 \to \rho_2]\!] := \mathcal{C}[\![\rho_1]\!] \Rightarrow_c \mathcal{C}[\![\rho_2]\!]$$

where A_ι is a discretely ordered poset and $X \Rightarrow_c Y$ is the (Scott-)continuous function space between X and Y, which is the set of all functions $g \in [X \Rightarrow Y]$ such that directed set $D \subseteq X$ implies $g(\bigsqcup D) = \bigsqcup_{d \in D} g(d)$.

This coincides with the definition of the $\mathcal{F}$-sort frame in Section 3.3, where $\mathcal{F} = \mathcal{C}$. The interpretation of goal terms is given in that section as well.

As mentioned in Chapter 2.2, it follows from Proposition 2.16 and a straightforward inductive argument that each $\langle \mathcal{C}[\![\rho]\!], \sqsubseteq_\rho \rangle$ is an algebraic lattice, ω-algebraic if each A_ι is countable. The same holds for $\mathcal{C}[\![\Delta]\!]$ for any relational sorting Δ:

$$\mathcal{C}[\![\Delta]\!] := \prod x \in \mathsf{dom}(\Delta).\, \mathcal{C}[\![\Delta(x)]\!],$$

with valuations $\alpha, \beta \in \mathcal{C}[\![\Delta]\!]$ ordered pointwise over their elements, i.e. $\alpha \sqsubseteq \beta$ for $\alpha, \beta \in \mathcal{C}[\![\Delta]\!]$ if and only if $\alpha(x) \sqsubseteq \beta(x)$ for all $x : \rho \in \Delta$.

Under the continuous interpretation $\mathcal{C}$, an instance $\langle \Delta, P, G \rangle$ of the HoCHC problem, with a logic program P,

$$P := \{F_1 : \Delta(F_1) = P(F_1), \quad \ldots, \quad F_m : \Delta(F_m) = P(F_m)\},$$

where each $P(F_i)$ is a goal term, gives rise to a *one-step consequence operator*:

$$T^{\mathcal{C}}_{P:\Delta} : \mathcal{C}[\![\Delta]\!] \to \mathcal{C}[\![\Delta]\!], \qquad T^{\mathcal{C}}_{P:\Delta}(\alpha)(F_i) := \mathcal{C}[\![\Delta \vdash P(F_i) : \Delta(F_i)]\!](\alpha).$$

A prefixed point of $T^{\mathcal{C}}_{P:\Delta}$ is a *model* of P. Showing that the continuous interpretation has the least model property thus boils down to whether there exists a least prefixed point of $T^{\mathcal{C}}_{P:\Delta}$.

By Kleene's Fixed-Point Theorem 2.3, if $\mathcal{C}[\![\Delta]\!]$ is a pointed dcpo (which it is, as above) and $T^{\mathcal{C}}_{P:\Delta}$ is continuous, then $T^{\mathcal{C}}_{P:\Delta}$ has a least prefixed point—least fixpoint—that is the supremum of the ascending Kleene chain of $T^{\mathcal{C}}_{P:\Delta}$. Thus, to establish the least model property it suffices to prove that $T^{\mathcal{C}}_{P:\Delta}$ is continuous, as we do in the following subsection.

Similar to the monotone HoCHC problem, a HoCHC problem $\langle \Delta, P, G \rangle$ is solvable under the continuous interpretation just if, for all models of the background theory, there is a prefixed point α of $T^{\mathcal{C}}_{P:\Delta}$ such that $\mathcal{C}[\![G]\!](\alpha) = 0$.

Comparison to first order. In general, the least fixpoint (and the greatest fixpoint) of a monotone function f on a complete lattice, like of $T^{\mathcal{M}}_{P:\Delta}$ on $\mathcal{M}[\![\Delta]\!]$, is computed through transfinite induction on the ordinal powers of f (Lloyd, 1987); the usual induction on natural numbers does not suffice, because

$$\bigsqcup\{f^n(\bot) \mid n \in \mathbb{N}\}$$

is not necessarily a fixpoint. The least fixpoint can be attained by induction beyond the natural numbers (i.e. transfinite induction). However, if f is continuous, then the above is—in fact—the least fixpoint (by Kleene's Fixed-Point Theorem 2.3) and induction on natural numbers suffices.

In first-order logic programming, the monotone one-step consequence operator is always continuous. However, as Section 3.4.2 shows, continuity does not come 'for free' for higher-order Horn clauses, so that the monotone and continuous interpretations are distinct for HoCHC. If we were to restrict ourselves to first-order variables and quantification, we would again obtain continuity 'for free' in the monotone interpretation, as in first-order logic.

3.4.1 Least model property

When we say that the monotone interpretation has a *least model property*, we mean that there exists a model M_P of logic program P, for every monotone HoCHC problem $\langle \Delta, P, G\rangle$, such that $M_P \sqsubseteq \alpha$ for all models α of P. Due to monotonicity, this means that a HoCHC problem $\langle \Delta, P, G\rangle$ is solvable under the monotone interpretation if and only if $\mathcal{M}[\![G]\!](M_P) = 0$.

By Kleene's Fixed-Point Theorem (Theorem 2.3), if we show that the $\mathcal{C}$ semantics of a goal term are, in fact, continuous, and thus that the one-step consequence operator is continuous, then we have proved that the HoCHC problem under the continuous interpretation also enjoys a least model property.

The bigger chunk of the work in this section is done in Lemma 3.4, with Theorem 3.6 establishing the least model property.

Proposition 3.1. *For all terms $\Delta \vdash T : \sigma$ that do not contain relational variables, $\mathcal{S}[\![\Delta \vdash T : \sigma]\!]$ is continuous.*

Proof. Recall that the ordering on $\mathcal{S}[\![\Delta]\!]$ is pointwise with respect to relational variables, and trivial on individuals. That means that if T does not contain relational

variables then $\mathcal{S}[\![\Delta \vdash T : \sigma]\!]$ is trivially monotone, as $\alpha \sqsubseteq \beta$ for $\alpha, \beta \in \mathcal{S}[\![\Delta]\!]$ means that α and β map individuals in Δ to the same element in $\mathcal{S}[\![\iota]\!]$, and thus:

$$\mathcal{S}[\![\Delta \vdash T : \sigma]\!](\alpha) = \mathcal{S}[\![\Delta \vdash T : \sigma]\!](\beta).$$

By extension, every I in a directed set $\mathcal{I}$ of valuations maps an individual in Δ to the same element in $\mathcal{S}[\![\iota]\!]$, and so does $\bigsqcup \mathcal{I}$. It follows that $\mathcal{S}[\![\Delta \vdash T : \sigma]\!]$ is trivially continuous. □

Lemma 3.2. *The codomain of $\alpha \in \mathcal{C}[\![\Delta]\!]$ is a dcpo.*

Proof. The codomain of $\alpha \in \mathcal{C}[\![\Delta]\!]$ is $\prod_{(x:\sigma)\in\Delta} \mathcal{C}[\![\sigma]\!]$, as α maps each $x : \sigma \in \Delta$ to an element of $\mathcal{C}[\![\sigma]\!]$, where σ is either ι or a relational sort ρ.

The discrete poset $\mathcal{C}[\![\iota]\!]$ is a dcpo. Each $\mathcal{C}[\![\rho]\!]$ is a complete lattice and therefore a dcpo. As the finite product of a number of dcpos is itself a dcpo, $\prod_{x:\sigma\in\Delta} \mathcal{C}[\![\sigma]\!]$ is a dcpo. □

Lemma 3.3. *For all goal terms $\Delta \vdash G : \rho$, (1) $\mathcal{C}[\![\Delta \vdash G : \rho]\!] : \mathcal{C}[\![\Delta]\!] \to \mathcal{C}[\![\rho]\!]$ is monotone, and (2) $\mathcal{C}[\![\Delta \vdash G : \rho]\!](\alpha)$ is monotone for all $\alpha \in \mathcal{C}[\![\Delta]\!]$ (i.e. if $\rho = \sigma \to \rho'$ then $\mathcal{C}[\![\Delta \vdash G : \rho]\!](\alpha) : \mathcal{C}[\![\sigma]\!] \to \mathcal{C}[\![\rho']\!]$ is monotone).*

Proof. We proceed by induction on the structure of G.

For (1), let $\alpha, \beta \in \mathcal{C}[\![\Delta]\!]$ such that $\alpha \sqsubseteq \beta$. The base cases are:

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash x : \rho]\!](\alpha) &= \alpha(x) \\
&\sqsubseteq \beta(x) \\
&= \mathcal{C}[\![\Delta \vdash x : \rho]\!](\beta) \\
\mathcal{C}[\![\Delta \vdash \varphi : o]\!](\alpha) &= \mathcal{S}[\![\Delta \vdash \varphi : o]\!](\alpha) \\
&= \mathcal{S}[\![\Delta \vdash \varphi : o]\!](\beta) \qquad \text{Prop 3.1} \\
&= \mathcal{C}[\![\Delta \vdash \varphi : o]\!](\beta)
\end{aligned}$$

For (2), the cases where $G : o$ are trivial since $\mathcal{C}[\![o]\!] = \mathcal{M}[\![o]\!] = \mathcal{S}[\![o]\!]$, so suppose that $G : \sigma \to \rho$. The relevant base case is:

$$\mathcal{C}[\![\Delta \vdash x : \sigma \to \rho]\!](\alpha) = \alpha(x) \in \mathcal{C}[\![\sigma \to \rho]\!]$$

Now suppose that (1) and (2) hold for all simpler goal terms. For (1), let $\alpha, \beta \in \mathcal{C}[\![\Delta]\!]$ such that $\alpha \sqsubseteq \beta$. It is clear that the claim holds for monotone operators and, or, and $\mathsf{cexists}_\sigma$. The remaining inductive cases are:

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash H_1\, H_2 : \rho_2]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!](\alpha)(\mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!](\alpha)) \\
&\sqsubseteq \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!](\beta)(\mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!](\beta)) && \text{IH} \\
&= \mathcal{C}[\![\Delta \vdash H_1\, H_2 : \rho_2]\!](\beta) \\
\mathcal{C}[\![\Delta \vdash H\, N : \rho]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](\alpha)(\mathcal{S}[\![\Delta \vdash N : \iota]\!](\alpha)) \\
&\sqsubseteq \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](\beta)(\mathcal{S}[\![\Delta \vdash N : \iota]\!](\alpha)) && \text{IH} \\
&= \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](\beta)(\mathcal{S}[\![\Delta \vdash N : \iota]\!](\beta)) && \text{Prop 3.1} \\
&= \mathcal{C}[\![\Delta \vdash H\, N : \rho]\!](\beta) \\
\mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](\alpha) &= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](\alpha[x \mapsto v]) \\
&\sqsubseteq \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](\beta[x \mapsto v]) && \text{IH} \\
&= \mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](\beta)
\end{aligned}$$

This proves that $\mathcal{C}[\![\Delta \vdash G : \rho]\!]$ is monotone for all goal terms $\Delta \vdash G : \rho$, which is (1).

Finally, for (2), again the cases where G has sort o are trivial, so let $G : \sigma \to \rho$. Then, the inductive cases are as follows.

For $\mathcal{C}[\![\Delta \vdash H\, N : \sigma \to \rho]\!](\alpha) = \mathcal{C}[\![\Delta \vdash H : \iota \to \sigma \to \rho]\!](\alpha)(\mathcal{S}[\![\Delta \vdash N]\!](\alpha))$, we know by the IH that $\mathcal{C}[\![\Delta \vdash H : \iota \to \sigma \to \rho]\!](\alpha)$ is a monotone function from $\mathcal{C}[\![\iota]\!]$ to $\mathcal{C}[\![\sigma \to \rho]\!]$. It follows that $\mathcal{C}[\![\Delta \vdash H\, N : \sigma \to \rho]\!](\alpha)$ is an element of $\mathcal{C}[\![\sigma \to \rho]\!]$, and thus that $\mathcal{C}[\![\Delta \vdash H\, N : \sigma \to \rho]\!](\alpha) : \mathcal{C}[\![\sigma]\!] \to \mathcal{C}[\![\rho]\!]$ is monotone, as required.

For $\mathcal{C}[\![\Delta \vdash H_1\, H_2 : \sigma \to \rho]\!](\alpha)$ we follow a similar reasoning using the IH twice.

And finally, for $\Delta \vdash \lambda x : \sigma.\, H : \rho$, let $f_1, f_2 \in \mathcal{C}[\![\sigma]\!]$ such that $f_1 \sqsubseteq f_2$. Then:

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](\alpha)(f_1) &= \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](\alpha[x \mapsto f_1]) \\
&\sqsubseteq \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](\alpha[x \mapsto f_2]) && \text{IH} \\
&= \mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](\alpha)(f_2)
\end{aligned}$$

This proves that $\mathcal{C}[\![\Delta \vdash G]\!](\alpha)$ is monotone for all goal terms $\Delta \vdash G$, which is (2). □

Lemma 3.4. *For all goal terms $\Delta \vdash G : \rho$, (1) $\mathcal{C}[\![\Delta \vdash G : \rho]\!] : \mathcal{C}[\![\Delta]\!] \to \mathcal{C}[\![\rho]\!]$ is continuous, and (2) $\mathcal{C}[\![\Delta \vdash G : \rho]\!](\alpha) \in \mathcal{C}[\![\rho]\!]$ for all $\alpha \in \mathcal{C}[\![\Delta]\!]$.*

Proof. We proceed by induction on the structure of G. By Lemma 3.3 we already have that $\mathcal{C}[\![\Delta \vdash G : \rho]\!]$ is monotone, and that $\mathcal{C}[\![\Delta \vdash G : \rho]\!](\alpha)$ is monotone for all $\alpha \in \mathcal{C}[\![\Delta]\!]$. Let $\mathcal{I} \subseteq \mathcal{C}[\![\Delta]\!]$ be a directed subset of valuations. Proving continuity of $\mathcal{C}[\![\Delta \vdash G : \rho]\!]$, which is (1), then comes down to showing that

$$\mathcal{C}[\![\Delta \vdash G : \rho]\!]\left(\bigsqcup \mathcal{I}\right) = \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash G : \rho]\!](I).$$

For (1), the base cases of the induction are follows:

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash x : \rho]\!]\left(\bigsqcup \mathcal{I}\right) &= \left(\bigsqcup \mathcal{I}\right)(x) \\
&= \bigsqcup_{I \in \mathcal{I}} I(x) \qquad \text{Prop 2.8, Lem 3.2} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash x : \rho]\!](I) \\
\mathcal{C}[\![\Delta \vdash \varphi : o]\!]\left(\bigsqcup \mathcal{I}\right) &= \mathcal{S}[\![\Delta \vdash \varphi : o]\!]\left(\bigsqcup \mathcal{I}\right) \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{S}[\![\Delta \vdash \varphi : o]\!](I) \qquad \text{Prop 3.1} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash \varphi : o]\!](I)
\end{aligned}$$

For (2), the cases of sort o are trivial since $\mathcal{C}[\![o]\!] = \mathcal{M}[\![o]\!] = \mathcal{S}[\![o]\!]$, so let G be of sort $\sigma \to \rho$. The relevant base case is:

$$\mathcal{C}[\![\Delta \vdash x : \sigma \to \rho]\!](\alpha) = \alpha(x) \in \mathcal{C}[\![\sigma \to \rho]\!]$$

Now suppose that (1) and (2) hold for simpler goal terms. It is clear that and, or, and $\mathsf{cexists}_\sigma$ are all continuous operators. For (1), let $\mathcal{I} \subseteq \mathcal{C}[\![\Delta]\!]$ be a directed set of valuations.

The remaining inductive cases are:

$$
\begin{aligned}
&\mathcal{C}[\![\Delta \vdash H_1\, H_2 : \rho_2]\!] \left(\bigsqcup \mathcal{I}\right) \\
&= \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!] \left(\bigsqcup \mathcal{I}\right) \left(\mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!] \left(\bigsqcup \mathcal{I}\right)\right) \\
&= \left(\bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!](I)\right) \left(\bigsqcup_{J \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!](J)\right) && \text{IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!](I) \left(\bigsqcup_{J \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!](J)\right) && \text{Prop 2.8} \\
&= \bigsqcup_{I,J \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!](I) \left(\mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!](J)\right) && \text{Prop 2.4, IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_1 : \rho_1 \to \rho_2]\!](I) \left(\mathcal{C}[\![\Delta \vdash H_2 : \rho_1]\!](I)\right) && \text{Prop 2.10} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H_1\, H_2 : \rho_2]\!](I)
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{C}[\![\Delta \vdash H\, N : \rho]\!] \left(\bigsqcup \mathcal{I}\right) \\
&= \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!] \left(\bigsqcup \mathcal{I}\right) \left(\mathcal{S}[\![\Delta \vdash N : \iota]\!] \left(\bigsqcup \mathcal{I}\right)\right) \\
&= \left(\bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](I)\right) \left(\mathcal{S}[\![\Delta \vdash N : \iota]\!] \left(\bigsqcup \mathcal{I}\right)\right) && \text{IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](I) \left(\mathcal{S}[\![\Delta \vdash N : \iota]\!] \left(\bigsqcup \mathcal{I}\right)\right) && \text{Prop 2.8} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](I) \left(\bigsqcup_{J \in \mathcal{I}} \mathcal{S}[\![\Delta \vdash N : \iota]\!](J)\right) && \text{Prop 3.1} \\
&= \bigsqcup_{I,J \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](I) \left(\mathcal{S}[\![\Delta \vdash N : \iota]\!](J)\right) && \text{Prop 2.4, IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H : \iota \to \rho]\!](I) \left(\mathcal{S}[\![\Delta \vdash N : \iota]\!](I)\right) && \text{Prop 2.10} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash H\, N : \rho]\!](I)
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!] \left(\bigsqcup \mathcal{I}\right) \\
&= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!] \left(\left(\bigsqcup \mathcal{I}\right)[x \mapsto v]\right) \\
&= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!] \left(\bigsqcup_{I \in \mathcal{I}} I[x \mapsto v]\right) \qquad x \notin \mathsf{dom}(\Delta) \\
&= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](I[x \mapsto v]) \qquad \text{IH} \\
&= \bigsqcup_{I \in \mathcal{I}} (\lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](I[x \mapsto v])) \qquad \text{Prop 2.8} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](I)
\end{aligned}
$$

This proves that $\mathcal{C}[\![\Delta \vdash G]\!]$ is continuous for all goal terms $\Delta \vdash G$, which is (1).

Finally, for (2), again the cases where G is of sort o are trivial, so let $G : \sigma \to \rho$. Then, the inductive cases are as follows.

For $\mathcal{C}[\![\Delta \vdash H\, N : \sigma \to \rho]\!](\alpha) = \mathcal{C}[\![\Delta \vdash H : \iota \to \sigma \to \rho]\!](\alpha)(\mathcal{S}[\![\Delta \vdash N]\!](\alpha))$, we know by the IH that $\mathcal{C}[\![\Delta \vdash H : \iota \to \sigma \to \rho]\!](\alpha) \in \mathcal{C}[\![\iota \to \sigma \to \rho]\!]$. It follows directly that $\mathcal{C}[\![\Delta \vdash H\, N : \sigma \to \rho]\!](\alpha)$ is an element of $\mathcal{C}[\![\sigma \to \rho]\!]$, as required.

For $\mathcal{C}[\![\Delta \vdash H_1\, H_2 : \sigma \to \rho]\!](\alpha)$ we follow a similar reasoning using the IH twice.

And finally, for $\Delta \vdash \lambda.x : \sigma.\, H : \rho$, let $F \subseteq \mathcal{C}[\![\sigma]\!]$ be a directed subset. Then:

$$
\begin{aligned}
\mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](\alpha) \left(\bigsqcup F\right) &= \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!] \left(\alpha \left[x \mapsto \bigsqcup F\right]\right) \\
&= \bigsqcup_{f \in F} \mathcal{C}[\![\Delta, x : \sigma \vdash H : \rho]\!](\alpha[x \mapsto f]) \qquad \text{IH} \\
&= \bigsqcup_{f \in F} \mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H : \rho]\!](\alpha)(f)
\end{aligned}
$$

This proves that $\mathcal{C}[\![\Delta \vdash G : \rho]\!](\alpha) \in \mathcal{C}[\![\rho]\!]$ for all goal terms $\Delta \vdash G : \rho$, i.e. (2). □

Corollary 3.5. *$T^{\mathcal{C}}_{P:\Delta}$ is continuous for all programs $\vdash P : \Delta$.*

Theorem 3.6 (Least model property)**.** *The continuous interpretation of the HoCHC problem has the least model property.*

Proof. By Corollary 3.5 and Kleene's Fixed-Point Theorem (see Theorem 2.3). □

3.4.2 The monotone one-step consequence operator is not continuous

Unlike in first-order Horn clauses, the monotone one-step consequence operator is not always continuous for HoCHC. We demonstrate this through an example HoCHC problem for which the one-step consequence operator in the monotone interpretation is not continuous.

Let us define a goal term $\Delta \vdash G : \rho$ and directed set of monotone valuations $\mathcal{I} \subseteq \mathcal{M}[\![\Delta]\!]$ such that:

$$\mathcal{M}[\![\Delta \vdash G : \rho]\!] \left(\bigsqcup \mathcal{I}\right) \neq \bigsqcup_{I \in \mathcal{I}} \mathcal{M}[\![\Delta \vdash G : \rho]\!](I)$$

Let $\Delta = \{Foo : ((\iota \to o) \to o) \to o, Bar : (\iota \to o) \to o\}$ be a sorting of relational variables, interpreted over the set of individuals $\mathcal{M}[\![\iota]\!] = \mathbb{N} \cup \{\infty\}$.

For all $i \in \mathbb{N} \cup \{\infty\}$, we define the (monotone) function

$$f_i : \iota \to o,\ x \mapsto \begin{cases} 0 & \text{if } x < i \\ 1 & \text{if } x \geq i. \end{cases}$$

Let us also define $f_{\infty+1}$ that maps all inputs to 0. This gives us the following ordering:

$$f_{\infty+1} \sqsubset f_\infty \sqsubset \cdots \sqsubset f_1 \sqsubset f_0$$

Note that while we regard $\mathbb{N} \cup \{\infty\}$ as a discretely ordered poset (consistent with a HoCHC sort frame), we can still use the ordering $0 < 1 < 2 < \cdots < \infty$ in our definitions. Intuitively, each f_i is the upward-closed characteristic function of $i \in \mathbb{N} \cup \{\infty\}$.

	0	1	…	∞
$f_{\infty+1}$	0	0	…	0
f_∞	0	0	…]	1
⋮	⋮	⋮	⋱	⋮
f_1	0	1	…	1
f_0	1	1	…	1

Figure 3.1: The result of $f : \iota \to o$ (left) applied to $i : \iota$ (top)

We define a family of (monotone) functions of sort $(\iota \to o) \to o$ by

$$g_i : (\iota \to o) \to o,\ f \mapsto \begin{cases} 0 & \text{if } f = f_j \sqsubset f_i \\ 1 & \text{if } f = f_j \sqsupseteq f_i \\ g_i(f_j) & \text{if } f_j = \bigsqcap\{f_k \mid k \in \mathbb{N} \text{ and } f \sqsubset f_k\} \end{cases}$$

where $j \in \mathbb{N} \cup \{\infty, \infty + 1\}$, for all $i \in \mathbb{N} \cup \{\infty\}$. Additionally, we define

$$g_{\infty-1} : (\iota \to o) \to o,\ f \mapsto \begin{cases} 0 & \text{if } f \sqsubseteq f_\infty \\ 1 & \text{otherwise} \end{cases}$$

It is clear that $g_{\infty-1}$ is monotone, as well as each g_i. This gives us the following lemma.

Lemma 3.7.

$$g_0 \sqsubset g_1 \sqsubset \cdots \sqsubset g_{\infty-1}$$

Proof. First we show that $g_i \sqsubset g_{\infty-1}$ for all $i \in \mathbb{N}$, and then that $g_i \sqsubseteq g_j$ for all $i, j \in \mathbb{N}$ such that $i \leq j$.

Let $i \in \mathbb{N}$. Suppose that $g_{\infty-1}(x) = 0$, then either $x = f_\infty$ or $x = f_{\infty+1}$. In both cases, $x \sqsubset f_i$ and thus $g_i(x) = 0$. We conclude that $g_i \sqsubset g_{\infty-1}$ for all $i \in \mathbb{N}$, where f_{i+1} is a witness to the strictness of this inclusion.

Let $i, j \in \mathbb{N}$ such that $i \leq j$. Note that $f_j \sqsubseteq f_i$. We proceed by case analysis on $x \in \mathcal{M}[\![\iota \to o]\!]$:

(i) If $x = f_k \sqsubset f_j$, then by transitivity $x \sqsubset f_i$ and $g_i(x) = 0 = g_j(x)$.

(ii) If $x = f_k \sqsupseteq f_j$, then $g_j(x) = 1$ and trivially $g_i(x) \sqsubseteq g_j(x)$.

(iii) Otherwise, i.e. there does not exist $k \in \mathbb{N} \cup \{\infty, \infty + 1\}$ such that $x = f_k$, let $f_\ell = \bigsqcap\{f_k \mid k \in \mathbb{N} \text{ and } x \sqsubset f_k\}$, so that $g_i(x) = g_i(f_\ell)$ and $g_j(x) = g_j(f_\ell)$. We can now apply either (i) or (ii) to f_ℓ.

We conclude that $g_i \sqsubseteq g_j$ for all $i, j \in \mathbb{N}$ such that $i \leq j$. □

Lemma 3.8 ($g_{\infty-1}$ is an infinite element)**.**

$$g_{\infty-1} = \bigsqcup_{i \in \mathbb{N}} g_i$$

Proof. Let $x \in \mathcal{M}[\![\iota \to o]\!]$. We aim to show that $g_{\infty-1}(x) = (\bigsqcup_{i \in \mathbb{N}} g_i)(x) = \bigsqcup_{i \in \mathbb{N}} g_i(x)$.

(i) If $x = f_j \sqsubseteq f_\infty$, then $g_{\infty-1}(x) = 0 = g_i(x)$ for all $i \in \mathbb{N}$.

	$f_{\infty+1}$	f_∞	$\dots$	f_2	f_1	f_0
g_0	0	0	$\dots$	0	0	1
g_1	0	0	$\dots$	0	1	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
$g_{\infty-1}$	0	0	$\lceil\dots$	1	1	1
g_∞	0	1	$\dots$	1	1	1

Figure 3.2: The result of $g : (\iota \to o) \to o$ (left) applied to $f : \iota \to o$ (top)

(ii) If $x = f_j \sqsupset f_\infty$, then $g_{\infty-1}(x) = 1 = g_j(x) = \bigsqcup_{i\in\mathbb{N}} g_i(x)$.

(iii) Otherwise, let $f_\ell = \bigsqcap\{f_k \mid k \in \mathbb{N} \text{ and } x \sqsubset f_k\}$, so that $g_{\infty-1}(x) = g_{\infty-1}(f_\ell) = 1$. Clearly, $g_\ell(x) = g_\ell(f_\ell) = 1 = \bigsqcup_{i\in\mathbb{N}} g_i(f_\ell) = \bigsqcup_{i\in\mathbb{N}} g_i(x)$.

We conclude that $g_{\infty-1} = \bigsqcup_{i\in\mathbb{N}} g_i$. Furthermore, $g_{\infty-1}$ is an infinite element, because we have obtained $g_{\infty-1}$ as the limit of a directed set in which it did not already occur. □

Theorem 3.9. *The one-step consequence operator in the monotone interpretation of the HoCHC problem is not generally continuous.*

Proof. Recall our sorting of relational variables:

$$\Delta = \{Foo : ((\iota \to o) \to o) \to o, Bar : (\iota \to o) \to o\}$$

Now consider the function $h : ((\iota \to o) \to o) \to o$ that maps all elements $g_{\infty-1}$ and larger to 1 and all other/smaller elements to 0. It is clear that h is an element of $\mathcal{M}[\![\Delta(Foo)]\!]$, as it respects the ordering of the input elements in $\mathcal{M}[\![(\iota \to o) \to o]\!]$.

Recall that continuity of a function is determined by its behaviour on infinite elements. In particular, consider $g_{\infty-1} = \bigsqcup_{i\in\mathbb{N}} g_i$, an infinite element by Lemma 3.8. This function witnesses that h is not continuous:

$$h(g_{\infty-1}) = 1 \neq 0 = \bigsqcup_{i\in\mathbb{N}} h(g_i)$$

Consider a directed set $\mathcal{I} = \{I_0, I_1, \dots\}$ of valuations (pointwise) determined by $I_i(Bar) = g_i$ and $I_i(Foo) = h$. Thus, $(\bigsqcup\mathcal{I})(Bar) = g_{\infty-1}$. This gives us the following semantics for the term $Foo\,Bar$

$$\mathcal{M}[\![Foo\,Bar]\!]\left(\bigsqcup\mathcal{I}\right) = \mathcal{M}[\![Foo]\!]\left(\bigsqcup\mathcal{I}\right)\left(\mathcal{M}[\![Bar]\!]\left(\bigsqcup\mathcal{I}\right)\right) = h(g_{\infty-1}) = 1$$

but

$$\bigsqcup_{I \in \mathcal{I}} \mathcal{M}[\![Foo\, Bar]\!](I) = \bigsqcup_{I \in \mathcal{I}} \mathcal{M}[\![Foo]\!](I)(\mathcal{M}[\![Bar]\!](I)) = \bigsqcup_{i \in \mathbb{N}} h(g_i) = 0.$$

□

Thus, we have proved the existence of HoCHC problem instances where the monotone one-step consequence operator is not continuous. This demonstrates that the continuous interpretation of HoCHC is, in fact, distinct from the monotone interpretation.

3.4.3 Equivalence with monotone interpretation

Cathcart Burn et al. (2018) have proved that every solution to a HoCHC problem instance $\langle \Delta, P, G \rangle$ under the standard interpretation determines a solution to $\langle \Delta, P, G \rangle$ under the monotone interpretation and vice versa. In this subsection, we show interreducibility between the monotone interpretation and the continuous interpretation (culminating in Theorem 3.18), which yields an equivalence between the standard and the continuous interpretation by transitivity.

Our approach is remarkably similar to Cathcart Burn et al.'s. We too rely on two Galois connections, pairs of adjoints, between complete lattices of monotone relations and—in our case—those of continuous relations. In fact, our approach is all but identical. This is due to Cathcart Burn et al.'s (monotone) Galois connections being continuous in our setting, which is precisely what we require. This subsection heavily relies on their work, albeit with small modifications.

We start by proving an auxiliary result left to the reader in Cathcart Burn et al. (2018), our Proposition 3.10. This is also where we first deviate from them, by correcting a minor error in the statement of their version of this proposition: for f and g to be well-defined we need $f : [P_2 \Rightarrow_m P_1] \to [Q_2 \Rightarrow_m Q_1]$ rather than $f : [P_1 \Rightarrow_m P_2] \to [Q_1 \Rightarrow_m Q_2]$, and similarly for g. That is, we swap P_1, P_2 and Q_1, Q_2 in the domains and codomains of f and g.

Of course, the second and crucial modification is that now we work with continuous function spaces rather than monotone ones, which brings us to a third difference: to ensure that f and g are well-defined (that they map continuous functions to continuous functions), we assume that f_1, f_2, g_1, g_2 are all continuous.

Proposition 3.10. *If $f_1 : P_1 \to Q_1$, $g_1 : Q_1 \to P_1$, $f_2 : P_2 \to Q_2$ and $g_2 : Q_2 \to P_2$ with $f_1 \dashv g_1$ and $f_2 \dashv g_2$ are continuous functions, then it follows that the pair of*

functions $f : [P_2 \Rightarrow_c P_1] \to [Q_2 \Rightarrow_c Q_1]$ *and* $g : [Q_2 \Rightarrow_c Q_1] \to [P_2 \Rightarrow_c P_1]$ *defined by* $f(h) = f_1 \circ h \circ g_2$ *and* $g(k) = g_1 \circ k \circ f_2$ *forms a Galois connection* $f \dashv g$ *between the corresponding continuous function spaces (ordered pointwise).*

Proof. Let f_1, f_2, f_1, g_2 and f, g as stated. Then, $f(h)$ and $g(k)$ are well-defined for all $h \in [P_2 \Rightarrow_c P_1]$ and $k \in [Q_2 \Rightarrow_c Q_1]$, since they are a composition of continuous functions. Thus, f and g are well-defined.

It can easily be verified that $f \dashv g$ is indeed a Galois connection:

- $f(g(k)) = f(g_1 \circ k \circ f_2) = f_1 \circ g_1 \circ k \circ f_2 \circ g_2 \leq k$
- $h \leq g_1 \circ f_1 \circ h \circ g_2 \circ f_2 = g(f_1 \circ h \circ g_2) = g(f(h))$

for all $h \in [P_2 \Rightarrow_c P_1]$ and $k \in [Q_2 \Rightarrow_c Q_1]$. □

Definition 3.11 (Embedding the continuous relations)**.** *Every complete lattice of continuous relations* $\mathcal{C}[\![\rho]\!]$ *can be embedded into the complete lattice of monotone relations* $\mathcal{M}[\![\rho]\!]$ *in the following two ways:*

$$\mathcal{M}[\![\rho]\!] \underset{L_\rho}{\overset{I_\rho}{\leftarrow\!\!\!\twoheadrightarrow}} \mathcal{C}[\![\rho]\!] \underset{J_\rho}{\overset{U_\rho}{\twoheadleftarrow\!\!\!\rightarrow}} \mathcal{M}[\![\rho]\!]$$

We define a family of right adjoints I_ρ *and a family of left adjoints* J_ρ *by induction on sort* ρ*. In the definition,* L_ρ *is the uniquely determined left adjoint of* I_ρ *and* U_ρ*.*

$$\begin{aligned} I_o(b) &:= b & J_o(b) &:= b \\ I_{\iota\to\rho}(r) &:= I_\rho \circ r & J_{\iota\to\rho}(r) &:= J_\rho \circ r \\ I_{\rho_1\to\rho_2}(r) &:= I_{\rho_2} \circ r \circ L_{\rho_1} & J_{\rho_1\to\rho_2}(r) &:= J_{\rho_2} \circ r \circ U_{\rho_1} \end{aligned}$$

Lemma 3.12. *For all relational sorts* ρ*, (i)* $L_\rho \dashv I_\rho$*, (ii)* $J_\rho \dashv U_\rho$*, and (iii) all four adjoints are continuous.*

Proof. We show only $L_\rho \dashv I_\rho$ and continuity of L_ρ, I_ρ because the proof for J_ρ, U_ρ is analogous. We show that $I_\rho : \mathcal{C}[\![\rho]\!] \to \mathcal{M}[\![\rho]\!]$ is a well-defined continuous right adjoint by induction on ρ.

- When $\rho = o$, $\mathcal{C}[\![o]\!] = \mathcal{M}[\![o]\!]$ and I_o is the identity, which is continuous and is its own left and right adjoint.
- When ρ is of shape $\iota \to \rho_2$, it follows from the induction hypothesis that $I_{\rho_2} : \mathcal{C}[\![\rho_2]\!] \to \mathcal{M}[\![\rho_2]\!]$ is a well-defined continuous right adjoint. By Proposition 3.10,

the mapping

$$r \mapsto I_{\rho_2} \circ r \circ \mathrm{id}_{\mathcal{C}[\![\iota]\!]} = I_{\rho_2} \circ r = I_{\iota \to \rho_2}(r)$$

is a well-defined right adjoint, using that $\mathcal{C}[\![\iota]\!] \Rightarrow_c A$ coincides with the full function space for all posets A due to the trivial ordering on the individuals in $\mathcal{C}[\![\iota]\!] = \mathcal{M}[\![\iota]\!]$. It remains to show that $I_{\iota \to \rho_2}$ is continuous, so let $D \subseteq \mathcal{C}[\![\iota \to \rho_2]\!]$ be a directed set. For all $x \in \mathcal{C}[\![\iota]\!]$:

$$\begin{aligned}
I_{\iota \to \rho_2}\left(\bigsqcup D\right)(x) &= I_\rho\left(\left(\bigsqcup D\right)(x)\right) \\
&= I_\rho\left(\bigsqcup_{d \in D} d(x)\right) \\
&= \bigsqcup_{d \in D} I_{\rho_2}(d(x)) \qquad \text{IH} \\
&= \bigsqcup_{d \in D} I_{\iota \to \rho_2}(d)(x) \\
&= \left(\bigsqcup_{d \in D} I_{\iota \to \rho_2}(d)\right)(x)
\end{aligned}$$

Thus, $I_{\iota \to \rho_2}$ well-defined continuous right adjoint. Similarly, $L_{\iota \to \rho_2}$ can be shown to be continuous.

- When ρ is of shape $\rho_1 \to \rho_2$, we decompose the definition as follows:

$$\mathcal{C}[\![\rho_1]\!] \Rightarrow_c \mathcal{C}[\![\rho_2]\!] \xrightarrow{r \mapsto I_{\rho_2} \circ r \circ L_{\rho_1}} \mathcal{M}[\![\rho_1]\!] \Rightarrow_c \mathcal{M}[\![\rho_2]\!] \xrightarrow{s \mapsto s} \mathcal{M}[\![\rho_1]\!] \Rightarrow_m \mathcal{M}[\![\rho_2]\!]$$

  By the induction hypothesis, $I_{\rho_1} : \mathcal{C}[\![\rho_1]\!] \to \mathcal{M}[\![\rho_1]\!]$ and $I_{\rho_2} : \mathcal{C}[\![\rho_2]\!] \to \mathcal{M}[\![\rho_2]\!]$ are well-defined continuous right adjoints, from which we may infer the existence of a continuous left adjoint $L_{\rho_1} : \mathcal{M}[\![\rho_1]\!] \to \mathcal{C}[\![\rho_1]\!]$.

  By Proposition 3.10, $r \mapsto I_{\rho_2} \circ r \circ L_{\rho_1} : [\mathcal{C}[\![\rho_1]\!] \Rightarrow_c \mathcal{C}[\![\rho_2]\!]] \to [\mathcal{M}[\![\rho_1]\!] \Rightarrow_c \mathcal{M}[\![\rho_2]\!]]$ is a well-defined right adjoint. Observe that there is a canonical inclusion between the continuous and monotone function spaces which, since it trivially preserves meets, is an right adjoint. The result follows because right adjoints compose.

  Finally, we show that $I_{\rho_1 \to \rho_2}$ is continuous. To this end, let $D \subseteq \mathcal{C}[\![\rho_1 \to \rho_2]\!]$ be

a directed set. For all $x \in \mathcal{C}[\![\rho_1]\!]$:

$$\begin{aligned} I_{\rho_1\to\rho_2}\left(\bigsqcup D\right)(x) &= I_{\rho_2}\left(\left(\bigsqcup D\right)(L_{\rho_1}(x))\right) \\ &= I_{\rho_2}\left(\bigsqcup_{d\in D} d(L_{\rho_1}(x))\right) \\ &= \bigsqcup_{d\in D} I_{\rho_2}(d(L_{\rho_1}(x))) \qquad \text{IH} \\ &= \bigsqcup_{d\in D} I_{\rho_1\to\rho_2}(d)(x) \\ &= \left(\bigsqcup_{d\in D} I_{\rho_1\to\rho_2}(d)\right)(x) \end{aligned}$$

Thus, $I_{\rho_1\to\rho_2}$ is continuous, and similarly $L_{\rho_1\to\rho_2}$ can be shown to be continuous.

□

These Galois connections provide a canonical way of moving between the universes of continuous and monotone relations. We extend these connections to mappings on valuations.

Definition 3.13 (Embedding the continuous valuations)**.** *Let $\alpha \in \mathcal{C}[\![\Delta]\!]$ and let $\beta \in \mathcal{M}[\![\Delta]\!]$. Analogous to Definition 3.11, we define $I_\Delta, J_\Delta : \mathcal{C}[\![\Delta]\!] \to \mathcal{M}[\![\Delta]\!]$ and $L_\Delta, U_\Delta : \mathcal{M}[\![\Delta]\!] \to \mathcal{C}[\![\Delta]\!]$. Let $K \in \{I, J\}$ and $H \in \{L, U\}$, so that:*

$$K_\Delta(\alpha)(x) := \begin{cases} \alpha(x) & \text{if } \Delta(x) = \iota \\ K_{\Delta(x)}(\alpha(x)) & \text{otherwise} \end{cases}$$

$$H_\Delta(\beta)(x) := \begin{cases} \beta(x) & \text{if } \Delta(x) = \iota \\ H_{\Delta(x)}(\beta(x)) & \text{otherwise} \end{cases}$$

Corollary 3.14. *For each sorting Δ, $L_\Delta \dashv I_\Delta$ and $J_\Delta \dashv U_\Delta$ are well-defined Galois connections.*

Lemma 3.15. *For sorts $\sigma ::= \iota \mid \rho$, $\mathsf{cexists}_\sigma \circ U_{\sigma\to o} \sqsubseteq \mathsf{mexists}_\sigma \sqsubseteq \mathsf{cexists}_\sigma \circ L_{\sigma\to o}$.*

Proof. For $\sigma = \iota$, $\mathsf{cexists}_\iota = \mathsf{mexists}_\iota$ by definition. Furthermore, $U_{\iota\to o}(s) = s = L_{\iota\to o}(s)$, so the result is clear. Otherwise, σ is some relational sort ρ and we observe that both of the following are true for all $s \in \mathcal{M}[\![\rho \to o]\!]$:

(i) for all $r \in \mathcal{C}[\![\rho]\!]$: $U_{\rho\to o}(s)(r) = 1$ implies $s(J_\rho(r)) = 1$

(ii) for all $t \in \mathcal{M}[\![\rho]\!]$: $s(t) = 1$ implies $L_{\rho\to o}(s)(L_\rho(t)) = 1$

To see that (i) is true, we observe that $U_{\rho\to o}(s)(r) = 1$ can be rewritten to $s(J_\rho(r)) = 1$. We reason as follows to show that (ii) holds: if $s(t) = 1$, then since $I_{\rho\to o} \circ L_{\rho\to o}$ is inflationary, also $I_{\rho\to o}(L_{\rho\to o}(s))(t) = 1$. Unfolding the definition of $I_{\rho\to o}$ and $I_o = \mathrm{id}_{\mathcal{C}[\![o]\!]}$ then yield $L_{\rho\to o}(s)(L_\rho(t)) = 1$.

Hence, by (i), witnesses r to the satisfiability of $U_{\rho\to o}(s)$ can be mapped to witnesses $J_\rho(r)$ to the satisfiability of s, and by (ii) witnesses t to the satisfiability of s can be mapped to witnesses $L_\rho(t)$ to the satisfiability of $L_{\rho\to o}(s)$, thus proving the lemma. □

Lemma 3.16. *For all goal terms* $\Delta \vdash G : \rho$, $J_\rho \circ \mathcal{C}[\![G]\!] \circ U_\Delta \sqsubseteq \mathcal{M}[\![G]\!] \sqsubseteq I_\rho \circ \mathcal{C}[\![G]\!] \circ L_\Delta$.

Proof. We prove the inclusion $J_\rho \circ \mathcal{C}[\![G]\!] \circ U_\Delta \sqsubseteq \mathcal{M}[\![G]\!]$ by induction on the structure of goal terms; showing the inclusion $\mathcal{M}[\![G]\!] \sqsubseteq I_\rho \circ \mathcal{C}[\![G]\!] \circ L_\Delta$ is dual. Let $\beta \in \mathcal{M}[\![\Delta]\!]$.

- For $\Delta \vdash x : \rho$,
$$J_\rho(\mathcal{C}[\![x]\!](U_\Delta(\beta))) = J_\rho(U_\rho(\beta(x)) \sqsubseteq \beta(x) = \mathcal{M}[\![x]\!](\beta),$$
using the definition of U_Δ and the fact that $J_\rho \circ U_\rho$ is deflationary.

- For $\Delta \vdash \varphi : o$,
$$J_o(\mathcal{C}[\![\varphi]\!](U_\Delta(\beta))) = \mathcal{S}[\![\varphi]\!](U_\Delta(\beta)) = \mathcal{S}[\![\varphi]\!](\beta) = \mathcal{M}[\![\varphi]\!](\beta),$$
using that $J_o = \mathrm{id}_{\mathcal{C}[\![o]\!]}$ and the variables of φ are assumed to be first-order.

- For $\Delta \vdash GH : \rho_2$ with $\Delta \vdash G : \rho_1 \to \rho_2$ and $\Delta \vdash H : \rho_1$, we reason as follows. First observe that
$$\begin{aligned} J_{\rho_2}(\mathcal{C}[\![GH]\!](U_\Delta(\beta))) &= J_{\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta)))(\mathcal{C}[\![H]\!](U_\Delta(\beta))) \\ &\sqsubseteq J_{\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta)))(U_{\rho_1}(J_{\rho_1}(\mathcal{C}[\![H]\!](U_\Delta(\beta))))), \end{aligned}$$
because $J_{\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta)))$ is monotone and $U_{\rho_1} \circ J_{\rho_1}$ is inflationary. Then,
$$\begin{aligned} &J_{\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta))(U_{\rho_1}(J_{\rho_1}(\mathcal{C}[\![H]\!](U_\Delta(\beta)))))) \\ &= J_{\rho_1\to\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta))(J_{\rho_1}(\mathcal{C}[\![H]\!](U_\Delta(\beta))))) \\ &\sqsubseteq J_{\rho_1\to\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta))(\mathcal{M}[\![H]\!](\beta)) \end{aligned}$$
using $J_{\rho_1\to\rho_2}(r) = J_{\rho_2} \circ r \circ U_{\rho_1}$, the induction hypothesis, and $J_{\rho_1\to\rho_2} \circ \mathcal{C}[\![G]\!](U_\Delta(\beta))$ being monotone. Finally,
$$J_{\rho_1\to\rho_2}(\mathcal{C}[\![G]\!](U_\Delta(\beta))(\mathcal{M}[\![H]\!](\beta)) \sqsubseteq \mathcal{M}[\![G]\!](\beta)(\mathcal{M}[\![H]\!](\beta)) = \mathcal{M}[\![GH]\!](\beta)$$
by the induction hypothesis.

- For $\Delta \vdash GN : \rho$ with $\Delta \vdash G : \iota \to \rho_2$ and $\Delta \vdash N : \iota$, we reason in a similar fashion to previous case but now using the identity $J_{\iota\to\rho}(r) = J_\rho \circ r$ to obtain:

$$\begin{aligned} J_\rho(\mathcal{C}[\![GN]\!](U_\Delta(\beta))) &= J_\rho(\mathcal{C}[\![G]\!](U_\Delta(\beta)))(\mathcal{S}[\![N]\!](U_\Delta(\beta))) \\ &= J_{\iota\to\rho}(\mathcal{C}[\![G]\!](U_\Delta(\beta)))(\mathcal{S}[\![N]\!](U_\Delta(\beta))) \\ &= J_{\iota\to\rho}(\mathcal{C}[\![G]\!](U_\Delta(\beta)))(\mathcal{S}[\![N]\!](\beta)) \\ &\sqsubseteq \mathcal{M}[\![G]\!](\beta)(\mathcal{S}[\![N]\!](\beta)) \\ &= \mathcal{M}[\![GN]\!](\beta) \end{aligned}$$

- For $\Delta \vdash \lambda x.\, H : \rho_1 \to \rho_2$ with $\Delta, x : \rho_1 \vdash H : \rho_2$, we first observe that for all $s \in \mathcal{M}[\![\rho_1]\!]$:

$$\begin{aligned} J_{\rho_1\to\rho_2}(\mathcal{C}[\![\lambda x.\, H]\!](U_\Delta(\beta)))(s) &= J_{\rho_2}(\mathcal{C}[\![\lambda x.\, H]\!](U_\Delta(\beta))(U_{\rho_1}(s))) \\ &= J_{\rho_2}(\mathcal{C}[\![H]\!](U_\Delta(\beta[x \mapsto s]))) \end{aligned}$$

  This is due to the identity $J_{\rho_1\to\rho_2}(r) = J_{\rho_2} \circ r \circ U_{\rho_1}$ and the definition of U_Δ. Then, we note that

$$J_{\rho_2}(\mathcal{C}[\![H]\!](U_\Delta(\beta[x \mapsto s]))) \sqsubseteq \mathcal{M}[\![H]\!](\beta[x \mapsto s])) = \mathcal{M}[\![\lambda x.\, H]\!](\beta)(s)$$

  by the induction hypothesis.

- For $\Delta \vdash \lambda x.\, H : \iota \to \rho$ with $\Delta, x : \iota \vdash H : \rho$, we reason in a similar way as in the previous case but now use the identity $J_{\iota\to\rho}(r) = J_\rho \circ r$ to obtain, for all $s \in \mathcal{M}[\![\iota]\!]$:

$$\begin{aligned} J_{\iota\to\rho}(\mathcal{C}[\![\lambda x.\, H]\!](U_\Delta(\beta)))(s) &= J_\rho(\mathcal{C}[\![\lambda x.\, H]\!](U_\Delta(\beta)))(s) \\ &= J_\rho(\mathcal{C}[\![H]\!](U_\Delta(\beta[x \mapsto s]))) \\ &\sqsubseteq \mathcal{M}[\![H]\!](\beta[x \mapsto s]) \\ &= \mathcal{M}[\![\lambda x.\, H]\!](\beta)(s) \end{aligned}$$

- If $\Delta \vdash \vee : o \to o \to o$ or $\Delta \vdash \wedge : o \to o \to o$, the claim holds by definition.

- For $\Delta \vdash \exists_\sigma : (\sigma \to o) \to o$, the result follows from Lemma 3.15:

$$\begin{aligned} J_{(\sigma\to o)\to o}(\mathcal{C}[\![\exists_\sigma]\!](U_\Delta(\beta))) &= J_{(\sigma\to o)\to o}(\mathsf{cexists}_\sigma) \\ &= \mathsf{cexists}_\sigma \circ U_{\sigma\to o} \\ &\sqsubseteq \mathsf{mexists}_\sigma \\ &= \mathcal{M}[\![\exists_\sigma]\!](\beta) \end{aligned}$$

□

Lemma 3.17 (Model translation)**.** *Fix a program* $\vdash P : \Delta$.

(i) If β *is a prefixed point of* $T^{\mathcal{M}}_{P:\Delta}$*, then* $U_\Delta(\beta)$ *is a prefixed point of* $T^{\mathcal{C}}_{P:\Delta}$.

(ii) If α *is a prefixed point of* $T^{\mathcal{C}}_{P:\Delta}$*, then* $I_\Delta(\alpha)$ *is a prefixed point of* $T^{\mathcal{M}}_{P:\Delta}$.

Proof. Let β be a prefixed point (model) of $T^{\mathcal{M}}_{P:\Delta}$, and α of $T^{\mathcal{C}}_{P:\Delta}$. It follows from Lemma 3.12 and Lemma 3.16 that for any goal term $G : \rho$:

$$\mathcal{C}[\![G]\!](U_\Delta(\beta)) \sqsubseteq U_\rho(\mathcal{M}[\![G]\!](\beta)) \qquad \mathcal{M}[\![G]\!](I_\Delta(\alpha)) \sqsubseteq I_\rho(\mathcal{C}[\![G]\!](\alpha))$$

The former follows from $J_\rho \circ \mathcal{C}[\![G]\!] \circ U_\Delta \sqsubseteq \mathcal{M}[\![G]\!]$, composing with U_ρ on the left and using that $\mathrm{id}_{\mathcal{C}[\![\rho]\!]} \sqsubseteq U_\rho \circ J_\rho$. The latter follows from $\mathcal{M}[\![G]\!] \sqsubseteq I_\rho \circ \mathcal{C}[\![G]\!] \circ L_\Delta$, composing with I_Δ on the right and using that $L_\Delta \circ I_\Delta \sqsubseteq \mathrm{id}_{\mathcal{C}[\![\Delta]\!]}$.

By definition, $T^{\mathcal{C}}_{P:\Delta}(\alpha)(x) = \mathcal{C}[\![P(x)]\!](\alpha)$ and $T^{\mathcal{M}}_{P:\Delta}(\beta)(x) = \mathcal{M}[\![P(x)]\!](\beta)$ for goal term $P(x)$, so that we can deduce the following from the above inclusions:

$$T^{\mathcal{C}}_{P:\Delta}\left(U_\Delta(\beta)\right) \sqsubseteq U_\Delta\left(T^{\mathcal{M}}_{P:\Delta}(\beta)\right) \qquad T^{\mathcal{M}}_{P:\Delta}\left(I_\Delta(\alpha)\right) \sqsubseteq I_\Delta\left(T^{\mathcal{C}}_{P:\Delta}(\alpha)\right)$$

For part (i), observe that $T^{\mathcal{M}}_{P:\Delta}(\beta) \sqsubseteq \beta$ (which holds because β is a prefixed point of $T^{\mathcal{M}}_{P:\Delta}$) implies $U_\Delta(T^{\mathcal{M}}_{P:\Delta}(\beta)) \sqsubseteq U_\Delta(\beta)$ by monotonicity. By the above inclusions, we then have $T^{\mathcal{C}}_{P:\Delta}(U_\Delta(\beta)) \sqsubseteq U_\Delta(\beta)$. Thus, $U_\Delta(\beta)$ is indeed a prefixed point of $T^{\mathcal{C}}_{P:\Delta}$. Part (ii) is analogous. □

Theorem 3.18. *The HoCHC problem* $\langle \Delta, P, G\rangle$ *is solvable under the monotone interpretation if and only if it is solvable under the continuous interpretation.*

Proof. Suppose that $\langle \Delta, P, G\rangle$ is solvable under the continuous interpretation. Then, for all models of the background theory $\mathcal{C}[\![G]\!](\mathcal{C}[\![P]\!]) = 0$ for least model $\mathcal{C}[\![P]\!]$ of $\vdash P : \Delta$. Let us fix a model of the background theory.

We compose L_o on the left and I_Δ on the right of Lemma 3.16 to obtain $L_o \circ \mathcal{M}[\![G : o]\!] \circ I_\Delta \sqsubseteq \mathcal{C}[\![G : o]\!]$. This means that $L_o(\mathcal{M}[\![G]\!](I_\Delta(\mathcal{C}[\![P]\!]))) = 0$. Since L_o is the identity on $\mathcal{C}[\![o]\!] = \mathcal{M}[\![o]\!]$, this gives us $\mathcal{M}[\![G]\!](I_\Delta(\mathcal{C}[\![P]\!])) = 0$. By the same lemma (composing I_Δ on the right), this gives us

$$\mathcal{M}[\![G]\!](I_\Delta(\mathcal{C}[\![P]\!])) \sqsubseteq I_o(\mathcal{C}[\![G]\!](\mathcal{C}[\![P]\!])) = \mathcal{C}[\![G]\!](\mathcal{C}[\![P]\!]) = 0$$

and thus $\mathcal{M}[\![G]\!](I_\Delta(\mathcal{C}[\![P]\!])) = 0$. By Lemma 3.17, $I_\Delta(\mathcal{C}[\![P]\!])$ is a model of $T^{\mathcal{M}}_{P:\Delta}$. We conclude that $\langle \Delta, P, G\rangle$ is solvable under the monotone interpretation.

The proof in the other direction is analogous, using the first parts of Lemma 3.16 and Lemma 3.17, respectively. □

3.5 Effectively continuous interpretation

Charalambidis et al. (2013) have defined a semantics for a positive existential fragment of higher-order logic programs—essentially higher-order Horn clauses without constraints—that they show to be continuous, even though it is not explicitly so. One would expect their semantics to coincide with our continuous semantics for HoCHC. This is indeed the case.

However, their sort frame is only explicitly monotone, not continuous. Hence, we call their semantics *effectively continuous*:

$$\mathcal{EC}[\![\iota]\!] := A_\iota \qquad \mathcal{EC}[\![o]\!] := \mathbb{B} \qquad \mathcal{EC}[\![\sigma \to \rho]\!] := \mathit{fin}(\mathcal{EC}[\![\sigma]\!]) \Rightarrow_m \mathcal{EC}[\![\rho]\!].$$

The difference with our continuous semantics lies in the definition of function spaces. The intuition is that continuity ensures that 'nothing is invented at infinity'. Thus, for any function $f \in \mathcal{C}[\![\rho_1 \to \rho_2]\!]$, the value of $f(x)$ for an infinite element $x \in \mathcal{C}[\![\rho_1]\!]$ is uniquely determined by $f(y)$ of its smaller finite/compact elements $y \in \mathcal{C}[\![\rho_1]\!]$.

Formally, for an infinite element $x \in \mathcal{C}[\![\rho_1]\!]$, there exists a non-decreasing chain $(x_i)_{i\in\omega}$ of elements in $\mathit{fin}(\mathcal{C}[\![\rho_1]\!])$ such that $x = \bigsqcup(x_i)_{i\in\omega}$. Now, if we consider a continuous function $f : \mathcal{C}[\![\rho_1]\!] \to \mathcal{C}[\![\rho_2]\!]$, then it must be that $f(x) = \bigsqcup_{i\in\omega} f(x_i)$. Thus, $f(x)$ is uniquely determined by all $f(x_i)$.

Our proof of interreducibility of the HoCHC problem under the continuous interpretation $\mathcal{C}$ and the HoCHC problem under the effectively continuous interpretation $\mathcal{EC}$ consists of two parts. First, we establish a bijection between $\mathcal{C}[\![\rho]\!]$ and $\mathcal{EC}[\![\rho]\!]$ for each relational sort ρ, which establishes a stronger equivalence than e.g. between the monotone and continuous interpretation (see Section 3.4.3). Second, we show that there exists a mapping from models of $\mathcal{C}$ to models of $\mathcal{EC}$ and vice versa.

Let us define the $\mathcal{EC}$ semantics of goal terms. The meaning $\mathcal{EC}[\![\Delta \vdash G : \rho]\!]$ of a goal

term $\Delta \vdash G : \rho$ is defined as:

$$\begin{aligned}
\mathcal{EC}[\![\Delta \vdash x : \rho]\!](\alpha) &:= \alpha(x) \\
\mathcal{EC}[\![\Delta \vdash \varphi : o]\!](\alpha) &:= \mathcal{S}[\![\Delta \vdash \varphi : o]\!](\alpha) \\
\mathcal{EC}[\![\Delta \vdash G\,H : \rho_2]\!](\alpha) &:= \bigsqcup \left\{ \mathcal{EC}[\![\Delta \vdash G : \rho_1 \to \rho_2]\!](\alpha)(b) \,\middle|\, \begin{array}{l} b \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!]), \\ b \sqsubseteq \mathcal{EC}[\![\Delta \vdash H]\!](\alpha) \end{array} \right\} \\
\mathcal{EC}[\![\Delta \vdash G\,N : \rho]\!](\alpha) &:= \mathcal{EC}[\![\Delta \vdash G : \iota \to \rho]\!](\alpha)(\mathcal{S}[\![\Delta \vdash N : \iota]\!](\alpha)) \\
\mathcal{EC}[\![\Delta \vdash \lambda x : \sigma.\, G : \rho]\!](\alpha) &:= \lambda x' \in \mathit{fin}(\mathcal{EC}[\![\sigma]\!]).\, \mathcal{EC}[\![\Delta, x : \sigma \vdash G : \rho]\!](\alpha[x \mapsto x']) \\
\mathcal{EC}[\![\Delta \vdash \wedge : o \to o \to o]\!](\alpha) &:= \mathsf{and} \\
\mathcal{EC}[\![\Delta \vdash \vee : o \to o \to o]\!](\alpha) &:= \mathsf{or} \\
\mathcal{EC}[\![\Delta \vdash \exists_\sigma : (\sigma \to o) \to o]\!](\alpha) &:= \mathsf{ecexists}_\sigma
\end{aligned}$$

Quantification is modelled by $\mathsf{ecexists}_\sigma(r) := \max\{r(d) \mid d \in \mathit{fin}(\mathcal{EC}[\![\sigma]\!])\}$, for $\sigma = \iota$ and $\sigma = \rho$. The two application cases could be merged into a single case but are worth spelling out, because the subcase for $\sigma = \iota$ is simpler and identical to the corresponding case in the continuous semantics.

The effectively continuous interpretation $\mathcal{EC}$ is strictly speaking not a Henkin frame, due to its interpretation of the arrow sorts. A Henkin frame $\mathcal{F}$ typically requires that $\mathcal{F}[\![\sigma_1 \to \sigma_2]\!]$ is interpreted as a subspace of $\mathcal{F}[\![\sigma_1]\!] \Rightarrow \mathcal{F}[\![\sigma_2]\!]$. However, the elements of $\mathcal{EC}[\![\sigma_1 \to \sigma_2]\!]$ are only partial functions on the domain $\mathcal{EC}[\![\sigma_1]\!]$, rather than total functions. This then requires a nonstandard semantics of application.

The effectively continuous interpretation may not be a Henkin frame by definition, but it is one in spirit; there exists a family of bijections between the continuous and effectively continuous sort frames. Furthermore, these bijections—rather than mere Galois connections—make the equivalence between the continuous and effectively continuous interpretations stronger than those between the other HoCHC interpretations: standard, monotone, and continuous.

Under the effectively continuous interpretation, an instance $\langle \Delta, P, G \rangle$ of the HoCHC problem, with a logic program P,

$$P := \{F_1 : \Delta(F_1) = P(F_1), \quad \ldots, \quad F_m : \Delta(F_m) = P(F_m)\},$$

where each $P(F_i)$ is a goal term, gives rise to a *one-step consequence operator*:

$$T^{\mathcal{EC}}_{P:\Delta} : \mathcal{EC}[\![\Delta]\!] \to \mathcal{EC}[\![\Delta]\!], \qquad T^{\mathcal{EC}}_{P:\Delta}(\alpha)(F_i) := \mathcal{EC}[\![\Delta \vdash P(F_i) : \Delta(F_i)]\!](\alpha).$$

A prefixed point of $T^{\mathcal{EC}}_{P:\Delta}$ is a *model* of P. An element $\alpha \in \mathcal{EC}[\![\Delta]\!]$ is a *valuation*.

Definition 3.19. *For algebraic poset A and complete lattice B, we define a bijection $h_{A,B} : [A \Rightarrow_c B] \to [\mathit{fin}(A) \Rightarrow_m B]$ as*

$$h_{A,B}(g) := g_{|\mathit{fin}}, \quad h^{-1}_{A,B}(k) := \left\{ \left(x, \bigsqcup_{i\in\omega} k(x_i) \right) \,\middle|\, \bigsqcup_{i\in\omega} x_i = x \text{ infinite, each } x_i \text{ finite} \right\} \oplus k$$

for all $g \in [A \Rightarrow_c B]$ and $k \in [\mathit{fin}(A) \Rightarrow_m B]$, where $g_{|\mathit{fin}} : \mathit{fin}(A) \to B$ is the restriction of g to a domain with only finite elements, and $a \oplus a'$ is the overriding union of $a : A \to B$ and $a' : A' \to B$ (which is the set-theoretic union of the graphs of a' and $a_{|A\setminus A'}$).

Lemma 3.20. *$h_{A,B}$ and $h^{-1}_{A,B}$ form a continuous bijection.*

Proof. Let $g \in [A \Rightarrow_c B]$ and $k \in [\mathit{fin}(A) \Rightarrow_m B]$.

$$h_{A,B}(h^{-1}_{A,B}(k)) = \left(\left\{ \left(x, \bigsqcup_{i\in\omega} k(x_i) \right) \,\middle|\, \bigsqcup_{i\in\omega} x_i = x \text{ infinite, each } x_i \text{ finite} \right\} \oplus k \right)_{|\mathit{fin}} = k$$

$$h^{-1}_{A,B}(h_{A,B}(g)) = \left\{ \left(x, \bigsqcup_{i\in\omega} g_{|\mathit{fin}}(x_i) \right) \,\middle|\, \bigsqcup_{i\in\omega} x_i = x \text{ infinite, each } x_i \text{ finite} \right\} \oplus g_{|\mathit{fin}} = g$$

Monotonicity of $h_{A,B}$ and $h^{-1}_{A,B}$ is trivial. Continuity of $h_{A,B}$ follows from:

$$h_{A,B}\left(\bigsqcup D\right) = \left(\bigsqcup D\right)_{|\mathit{fin}} = \bigsqcup_{d\in D} d_{|\mathit{fin}} = \bigsqcup_{d\in D} h_{A,B}(d)$$

For continuity of $h^{-1}_{A,B}$, we need to show that $h^{-1}_{A,B}\left(\bigsqcup D\right)(x) = \bigsqcup_{d\in D} h^{-1}_{A,B}(d)(x)$ for all directed sets $D \subseteq [\mathit{fin}(A) \Rightarrow_m B]$ and all $x \in A$. The case where $x \in \mathit{fin}(A)$ follows directly from Proposition 2.8, while $x \notin \mathit{fin}(A)$ requires some more work:

$$\begin{aligned}
h^{-1}_{A,B}\left(\bigsqcup D\right)(x) &= \bigsqcup_{i\in\omega} \left(\bigsqcup D\right)(x_i) \qquad \text{for } \bigsqcup_{i\in\omega} x_i = x, \text{ each } x_i \text{ finite} \\
&= \bigsqcup_{i\in\omega} \bigsqcup_{d\in D} d(x_i) \\
&= \bigsqcup_{d\in D} \bigsqcup_{i\in\omega} d(x_i) \qquad \text{Prop 2.10} \\
&= \bigsqcup_{d\in D} h^{-1}_{A,B}(d)(x)
\end{aligned}$$

We conclude that $h_{A,B}$ and $h^{-1}_{A,B}$ are continuous bijections. □

Lemma 3.21. *For algebraic poset A and complete lattice B,*

$$h_{A,B}(g) = g \circ h_{A,A}(\mathrm{id}_A)$$

for all $g \in [A \Rightarrow_c B]$.

Proof. Let $g \in [A \Rightarrow_c B]$, so that:

$$g \circ h_{A,A}(\mathrm{id}_A) = g \circ \mathrm{id}_{\mathit{fin}(A)} = g_{|\mathit{fin}} = h_{A,B}(g)$$

□

Lemma 3.22 (Charalambidis et al., 2013)**.** *For every relational sort ρ, $\mathcal{EC}[\![\rho]\!]$ is an algebraic lattice (ω-algebraic if A_ι is countable).*

Theorem 3.23. *There is a continuous bijection φ_ρ between $\mathcal{C}[\![\rho]\!]$ and $\mathcal{EC}[\![\rho]\!]$, for all relational sorts ρ.*

Proof. By induction on ρ. For $\rho = o$, $\varphi_o = \varphi_o^{-1}$ is the identity on $\mathcal{C}[\![o]\!] = \mathcal{EC}[\![o]\!]$.

Suppose that $\rho = \iota \to \rho_2$. By the induction hypothesis, $\varphi_{\rho_2} : \mathcal{C}[\![\rho_2]\!] \to \mathcal{EC}[\![\rho_2]\!]$ is a continuous bijection (with continuous inverse $\varphi_{\rho_2}^{-1}$). Since $\mathcal{C}[\![\iota]\!] = \mathcal{EC}[\![\iota]\!]$, the function $\varphi_{\iota\to\rho_2} : \mathcal{C}[\![\iota \to \rho_2]\!] \to \mathcal{EC}[\![\iota \to \rho_2]\!]$ defined by

$$\varphi_{\iota\to\rho_2}(f) := \varphi_{\rho_2} \circ f, \qquad \varphi^{-1}_{\iota\to\rho_2}(g) := \varphi_{\rho_2}^{-1} \circ g$$

is a bijection for all $f \in \mathcal{C}[\![\iota \to \rho_2]\!]$ and $g \in \mathcal{EC}[\![\iota \to \rho_2]\!]$. It is not hard to see that both $\varphi_{\iota\to\rho_2}$ and $\varphi^{-1}_{\iota\to\rho_2}$ are continuous by appealing to the induction hypothesis and the fact that the composition of multiple continuous functions is itself continuous.

Suppose that $\rho = \rho_1 \to \rho_2$. By the induction hypothesis, $\varphi_{\rho_1} : \mathcal{C}[\![\rho_1]\!] \to \mathcal{EC}[\![\rho_2]\!]$ and $\varphi_{\rho_2} : \mathcal{C}[\![\rho_2]\!] \to \mathcal{EC}[\![\rho_2]\!]$ are continuous bijections, with continuous inverses $\varphi_{\rho_1}^{-1}$ and $\varphi_{\rho_2}^{-1}$ resp. Let us define $\varphi_{\rho_1\to\rho_2} : \mathcal{C}[\![\rho_1 \to \rho_2]\!] \to \mathcal{EC}[\![\rho_1 \to \rho_2]\!]$ in the following way:

$$\varphi_{\rho_1\to\rho_2}(f) := \varphi_{\rho_2} \circ f \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{C}[\![\rho_1]\!]}(\varphi_{\rho_1}^{-1}), \quad \varphi^{-1}_{\rho_1\to\rho_2}(g) := \varphi_{\rho_2}^{-1} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(g) \circ \varphi_{\rho_1},$$

for all $f \in \mathcal{C}[\![\rho_1 \to \rho_2]\!]$ and $g \in \mathcal{EC}[\![\rho_1 \to \rho_2]\!]$. We verify that $\varphi_{\rho_1\to\rho_2}$ is indeed a

bijection using the induction hypothesis and repeated applications of Lemma 3.21.

$$\begin{aligned}
\varphi_{\rho_1\to\rho_2}(\varphi^{-1}_{\rho_1\to\rho_2}(g)) &= \varphi_{\rho_1\to\rho_2}(\varphi^{-1}_{\rho_2} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(g) \circ \varphi_{\rho_1}) \\
&= \varphi_{\rho_2} \circ \varphi^{-1}_{\rho_2} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(g) \circ \varphi_{\rho_1} \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{C}[\![\rho_1]\!]}(\varphi^{-1}_{\rho_1}) \\
&= \varphi_{\rho_2} \circ \varphi^{-1}_{\rho_2} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(g) \circ \varphi_{\rho_1} \circ \varphi^{-1}_{\rho_1} \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]}) \\
&= h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(g) \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]}) \\
&= h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(g)) \\
&= g
\end{aligned}$$

$$\begin{aligned}
\varphi^{-1}_{\rho_1\to\rho_2}(\varphi_{\rho_1\to\rho_2}(f)) &= \varphi^{-1}_{\rho_1\to\rho_2}(\varphi_{\rho_2} \circ f \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{C}[\![\rho_1]\!]}(\varphi^{-1}_{\rho_1})) \\
&= \varphi^{-1}_{\rho_2} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(\varphi_{\rho_2} \circ f \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{C}[\![\rho_1]\!]}(\varphi^{-1}_{\rho_1})) \circ \varphi_{\rho_1} \\
&= \varphi^{-1}_{\rho_2} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(\varphi_{\rho_2} \circ f \circ \varphi^{-1}_{\rho_1} \circ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})) \circ \varphi_{\rho_1} \\
&= \varphi^{-1}_{\rho_2} \circ h^{-1}_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_2]\!]}(\varphi_{\rho_2} \circ f \circ \varphi^{-1}_{\rho_1})) \circ \varphi_{\rho_1} \\
&= \varphi^{-1}_{\rho_2} \circ \varphi_{\rho_2} \circ f \circ \varphi^{-1}_{\rho_1} \circ \varphi_{\rho_1} \\
&= f
\end{aligned}$$

Finally, $\varphi_{\rho_1\to\rho_2}$ and $\varphi^{-1}_{\rho_1\to\rho_2}$ can be shown to be continuous by appealing to the induction hypothesis and the fact that the composition of multiple continuous functions is itself continuous. □

These bijections provide a canonical way to move between the universes of continuous and effectively continuous relations. We extend these bijections to mappings on valuations.

Definition 3.24. *Let $\alpha \in \mathcal{C}[\![\Delta]\!]$ and $\beta \in \mathcal{EC}[\![\Delta]\!]$. We define $\varphi_\Delta : \mathcal{C}[\![\Delta]\!] \to \mathcal{EC}[\![\Delta]\!]$ and its inverse $\varphi^{-1}_\Delta : \mathcal{EC}[\![\Delta]\!] \to \mathcal{C}[\![\Delta]\!]$:*

$$\varphi_\Delta(\alpha)(x) := \begin{cases} \alpha(x) & \text{if } \Delta(x) = \iota \\ \varphi_{\Delta(x)}(\alpha(x)) & \text{otherwise} \end{cases}$$

$$\varphi^{-1}_\Delta(\beta)(x) := \begin{cases} \beta(x) & \text{if } \Delta(x) = \iota \\ \varphi^{-1}_{\Delta(x)}(\beta(x)) & \text{otherwise} \end{cases}$$

Corollary 3.25. *For each sorting Δ, $\varphi_\Delta : \mathcal{C}[\![\Delta]\!] \to \mathcal{EC}[\![\Delta]\!]$ is a continuous bijection.*

Lemma 3.26. *For sorts $\sigma ::= \iota \mid \rho$, $\mathsf{cexists}_\sigma = \mathsf{ecexists}_\sigma \circ \varphi_{\sigma\to o}$.*

Proof. Let $\sigma = \iota$. Note that $\mathit{fin}(\mathcal{C}[\![\iota]\!]) = \mathcal{C}[\![\iota]\!] = \mathcal{EC}[\![\iota]\!] = \mathit{fin}(\mathcal{EC}[\![\iota]\!])$. It follows that $\mathcal{C}[\![\iota \to o]\!] = \mathcal{EC}[\![\iota \to o]\!]$. Clearly, any witness to the satisfiability of $s \in \mathcal{C}[\![\iota \to o]\!]$ is also a witness to the satisfiability of $\varphi_{\iota \to o}(s)$ and vice versa.

Let σ be some relational sort ρ. We observe that for all $s \in \mathcal{C}[\![\rho \to o]\!]$:

- Suppose that $\varphi_{\rho \to o}(s)(r) = 1$ for some $r \in \mathit{fin}(\mathcal{EC}[\![\rho]\!])$. Then,

  $$(\varphi_o \circ s \circ h_{\mathcal{EC}[\![\rho]\!],\mathcal{C}[\![\rho]\!]}(\varphi_\rho^{-1}))(r) = (s \circ \varphi_\rho^{-1} \circ h_{\mathcal{EC}[\![\rho]\!],\mathcal{EC}[\![\rho]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho]\!]}))(r) = 1$$

  However, since $h_{\mathcal{EC}[\![\rho]\!],\mathcal{EC}[\![\rho]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho]\!]})$ is simply the inclusion from $\mathit{fin}(\mathcal{EC}[\![\rho]\!])$ into $\mathcal{EC}[\![\rho]\!]$, this means that $s(\varphi_\rho^{-1}(r)) = 1$. Thus, witnesses $r \in \mathit{fin}(\mathcal{EC}[\![\rho]\!])$ to the satisfiability of $\varphi_{\rho \to o}(s) \in \mathcal{EC}[\![\rho \to o]\!]$ can be mapped to witnesses $\varphi_\rho^{-1}(r) \in \mathcal{C}[\![\rho]\!]$ to the satisfiability of $s \in \mathcal{C}[\![\rho \to o]\!]$.

- Suppose that $s(t) = 1$ for some $t \in \mathcal{C}[\![\rho]\!]$. Using a chain of equivalences:

  $$\begin{aligned} s(t) &= \varphi_{\rho \to o}^{-1}(\varphi_{\rho \to o}(s))(t) \\ &= (\varphi_o^{-1} \circ h^{-1}_{\mathcal{EC}[\![\rho]\!],\mathcal{EC}[\![o]\!]}(\varphi_{\rho \to o}(s)) \circ \varphi_\rho)(t) \\ &= h^{-1}_{\mathcal{EC}[\![\rho]\!],\mathcal{EC}[\![o]\!]}(\varphi_{\rho \to o}(s))(\varphi_\rho(t)) \end{aligned}$$

  If we unpack the definition of $h^{-1}_{\mathcal{EC}[\![\rho]\!],\mathcal{EC}[\![o]\!]}$, we obtain the following:

  $$1 = \begin{cases} \varphi_{\rho \to o}(s)(\varphi_\rho(t)) & \text{if } \varphi_\rho(t) \text{ is finite} \\ \bigsqcup_{i \in \omega} \varphi_{\rho \to o}(s)(x_i) & \text{if } \varphi_\rho(t) = \bigsqcup (x_i)_{i \in \omega} \text{ for finite } x_i\text{s} \end{cases}$$

  In latter case, there must exist (a finite) x_i such that $\varphi_{\rho \to o}(x_i)$. This means that a witness $t \in \mathcal{C}[\![\rho]\!]$ of the satisfiability of $s \in \mathcal{C}[\![\rho \to o]\!]$ can be mapped to either a witness $\varphi_\rho(t) \in \mathit{fin}(\mathcal{EC}[\![\rho]\!])$ or a witness $x_i \in \mathit{fin}(\mathcal{EC}[\![\rho]\!])$ to the satisfiability of $\varphi_{\rho \to o}(s) \in \mathcal{EC}[\![\rho \to o]\!]$.

□

Lemma 3.27. *For all goal terms $\Delta \vdash G : \rho$, $\varphi_\rho \circ \mathcal{C}[\![\Delta \vdash G : \rho]\!] \circ \varphi_\Delta^{-1} = \mathcal{EC}[\![\Delta \vdash G : \rho]\!]$.*

Proof. We proceed by induction on the structure of goal terms. Let $\beta \in \mathcal{EC}[\![\Delta]\!]$.

- For $\Delta \vdash x : \rho$,

  $$\varphi_\rho(\mathcal{C}[\![x]\!](\varphi_\Delta^{-1}(\beta)) = \varphi_\rho(\varphi_\Delta^{-1}(\beta)(x)) = \varphi_\rho(\varphi_\rho^{-1}(\beta(x))) = \beta(x) = \mathcal{EC}[\![x]\!](\beta).$$

- For $\Delta \vdash \psi : o$ with ψ a formula of the constraint language,

$$\varphi_o(\mathcal{C}[\![\psi]\!](\varphi_\Delta^{-1}(\beta))) = \mathcal{C}[\![\psi]\!](\varphi_\Delta^{-1}(\beta)) = \mathcal{S}[\![\psi]\!](\varphi_\Delta^{-1}(\beta)) = \mathcal{S}[\![\psi]\!](\beta) = \mathcal{EC}[\![\psi]\!](\beta),$$

using that the free variables of ψ are assumed to be first-order.

- For $\Delta \vdash G\,H : \rho_2$ with $\Delta \vdash G : \rho_1 \to \rho_2$ and $\Delta \vdash H : \rho_1$, we reason as follows, aided by the induction hypothesis:

$$\begin{aligned}
&\varphi_{\rho_2}(\mathcal{C}[\![G\,H]\!](\varphi_\Delta^{-1}(\beta))) \\
&= \varphi_{\rho_2}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta))(\mathcal{C}[\![H]\!](\varphi_\Delta^{-1}(\beta)))) \\
&= \varphi_{\rho_2}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta))(\varphi_{\rho_1}^{-1}(\varphi_{\rho_1}(\mathcal{C}[\![H]\!](\varphi_\Delta^{-1}(\beta)))))) \\
&= \varphi_{\rho_2}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta))(\varphi_{\rho_1}^{-1}(\mathcal{EC}[\![H]\!](\beta)))) \\
&= \bigsqcup \left\{ \varphi_{\rho_2}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta))(\varphi_{\rho_1}^{-1}(h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})(b)))) \;\middle|\; \begin{array}{l} b \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!]), \\ b \sqsubseteq \mathcal{EC}[\![H]\!](\beta) \end{array} \right\} \\
&= \bigsqcup \left\{ \varphi_{\rho_2}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta))(h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{C}[\![\rho_1]\!]}(\varphi_{\rho_1}^{-1})(b))) \;\middle|\; \begin{array}{l} b \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!]), \\ b \sqsubseteq \mathcal{EC}[\![H]\!](\beta) \end{array} \right\} \\
&= \bigsqcup \left\{ \varphi_{\rho_1\to\rho_2}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta)))(b) \;\middle|\; b \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!]), b \sqsubseteq \mathcal{EC}[\![H]\!](\beta) \right\} \\
&= \bigsqcup \left\{ \mathcal{EC}[\![G]\!](\beta)(b) \;\middle|\; b \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!]), b \sqsubseteq \mathcal{EC}[\![H]\!](\beta) \right\} \\
&= \mathcal{EC}[\![G\,H]\!](\beta)
\end{aligned}$$

To see that the introduction of the least upper bound is valid, we use case analysis on $\mathcal{EC}[\![H]\!](\beta)$. If this is a finite element, then $h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})$ is the identity function on $\mathcal{EC}[\![o]\!]$ and this step is clear. Otherwise, if $\mathcal{EC}[\![H]\!](\beta)$ is an infinite element, then $h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})$ is the embedding of $\mathit{fin}(\mathcal{EC}[\![\rho_1]\!])$ into $\mathcal{EC}[\![\rho_1]\!]$. Because $\mathcal{EC}[\![\rho_1]\!]$ is an algebraic lattice (Lemma 3.22), $\mathcal{EC}[\![H]\!](\beta)$ is the least upper bound of all smaller, finite elements, so that:

$$\mathcal{EC}[\![H]\!](\beta) = \bigsqcup \left\{ h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})(b) \;\middle|\; \begin{array}{l} b \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!]), \\ b \sqsubseteq \mathcal{EC}[\![H]\!](\beta) \end{array} \right\}$$

- For $\Delta \vdash G\,N : \rho$ with $\Delta \vdash G : \iota \to \rho_2$ and $\Delta \vdash N : \iota$, we use the induction hypothesis to obtain:

$$\begin{aligned}
\varphi_\rho(\mathcal{C}[\![G\,N]\!](\varphi_\Delta^{-1}(\beta))) &= \varphi_\rho((\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta)))(\mathcal{S}[\![N]\!](\varphi_\Delta^{-1}(\beta)))) \\
&= \varphi_{\iota\to\rho}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta)))(\mathcal{S}[\![N]\!](\varphi_\Delta^{-1}(\beta))) \\
&= \varphi_{\iota\to\rho}(\mathcal{C}[\![G]\!](\varphi_\Delta^{-1}(\beta)))(\mathcal{S}[\![N]\!](\beta)) \\
&= \mathcal{EC}[\![G]\!](\beta)(\mathcal{S}[\![N]\!](\beta)) \\
&= \mathcal{EC}[\![G\,N]\!](\beta)
\end{aligned}$$

- For $\Delta \vdash \lambda x.\, H : \rho_1 \to \rho_2$ with $\Delta, x : \rho_1 \vdash H : \rho_2$, for all $s \in \mathit{fin}(\mathcal{EC}[\![\rho_1]\!])$:

$$\begin{aligned}
&\varphi_{\rho_1\to\rho_2}(\mathcal{C}[\![\lambda x.\, H]\!](\varphi_\Delta^{-1}(\beta)))(s)\\
&= \varphi_{\rho_2}(\mathcal{C}[\![\lambda x.\, H]\!](\varphi_\Delta^{-1}(\beta))(h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{C}[\![\rho_1]\!]}(\varphi_{\rho_1}^{-1})(s)))\\
&= \varphi_{\rho_2}(\mathcal{C}[\![\lambda x.\, H]\!](\varphi_\Delta^{-1}(\beta))(\varphi_{\rho_1}^{-1}(h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})(s))))\\
&= \varphi_{\rho_2}(\mathcal{C}[\![H]\!](\varphi_\Delta^{-1}(\beta)[x \mapsto \varphi_{\rho_1}^{-1}(h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})(s))]))\\
&= \varphi_{\rho_2}(\mathcal{C}[\![H]\!](\varphi_\Delta^{-1}(\beta[x \mapsto h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})(s)])))\\
&= \mathcal{EC}[\![H]\!](\beta[x \mapsto h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})(s)])
\end{aligned}$$

  Note that $h_{\mathcal{EC}[\![\rho_1]\!],\mathcal{EC}[\![\rho_1]\!]}(\mathrm{id}_{\mathcal{EC}[\![\rho_1]\!]})$ is simply the embedding from $\mathit{fin}(\mathcal{EC}[\![\rho_1]\!])$ into $\mathcal{EC}[\![\rho_1]\!]$, which means the above is equal to $\mathcal{EC}[\![\lambda x.\, H]\!](\beta)(s)$.

- For $\Delta \vdash \lambda x.\, H : \iota \to \rho$ with $\Delta, x : \iota \vdash H : \rho$, we reason similarly to the previous case, using the induction hypothesis to obtain for all $s \in \mathcal{C}[\![\iota]\!]$:

$$\begin{aligned}
\varphi_{\iota\to\rho}(\mathcal{C}[\![\lambda x.\, H]\!](\varphi_\Delta^{-1}(\beta)))(s) &= \varphi_\rho(\mathcal{C}[\![\lambda x.\, H]\!](\varphi_\Delta^{-1}(\beta)))(s)\\
&= \varphi_\rho(\mathcal{C}[\![H]\!](\varphi_\Delta^{-1}(\beta[x \mapsto s])))\\
&= \mathcal{EC}[\![H]\!](\beta[x \mapsto s])\\
&= \mathcal{EC}[\![\lambda x.\, H]\!](\beta)(s)
\end{aligned}$$

- For $\Delta \vdash \vee : o \to o \to o$ or $\Delta \vdash \wedge : o \to o \to o$, the claim holds by definition.

- For $\Delta \vdash \exists_\sigma : (\sigma \to o) \to o$,

$$\begin{aligned}
\varphi_{(\sigma\to o)\to o}(\mathcal{C}[\![\exists_\sigma]\!](\varphi_\Delta^{-1}(\beta))) &= \varphi_{(\sigma\to o)\to o}(\mathsf{cexists}_\sigma)\\
&= \mathsf{cexists}_\sigma \circ h_{\mathcal{EC}[\![\sigma\to o]\!],\mathcal{C}[\![\sigma\to o]\!]}(\varphi_{\sigma\to o}^{-1})\\
&= \mathsf{cexists}_\sigma \circ \varphi_{\sigma\to o}^{-1} \circ h_{\mathcal{EC}[\![\sigma\to o]\!],\mathcal{EC}[\![\sigma\to o]\!]}(\mathrm{id}_{\mathcal{EC}[\![\sigma\to o]\!]})\\
&= \mathsf{cexists}_\sigma \circ \varphi_{\sigma\to o}^{-1}\\
&= \mathsf{ecexists}_\sigma\\
&= \mathcal{EC}[\![\exists_\sigma]\!](\beta)
\end{aligned}$$

  using Lemma 3.26 and that $h_{\mathcal{EC}[\![\sigma\to o]\!],\mathcal{EC}[\![\sigma\to o]\!]}(\mathrm{id}_{\mathcal{EC}[\![\sigma\to o]\!]})$ is the embedding from $\mathit{fin}(\mathcal{EC}[\![\sigma \to o]\!])$ into $\mathcal{EC}[\![\sigma \to o]\!]$.

□

Lemma 3.28 (Model translation)**.** *Fix a program* $\vdash P : \Delta$.

(i) If β *is a prefixed point of* $T^{\mathcal{EC}}_{P:\Delta}$ *then* $\varphi_\Delta^{-1}(\beta)$ *is a prefixed point of* $T^{\mathcal{C}}_{P:\Delta}$.

(ii) If α is a prefixed point of $T^{\mathcal{C}}_{P:\Delta}$, then $\varphi_\Delta(\alpha)$ is a prefixed point of $T^{\mathcal{EC}}_{P:\Delta}$.

Proof. Let β be a prefixed point of $T^{\mathcal{EC}}_{P:\Delta}$, and α of $T^{\mathcal{C}}_{P:\Delta}$. It follows from Lemma 3.27 that for any goal term $G : \rho$:

$$\mathcal{EC}[\![G]\!] = \varphi_\rho \circ \mathcal{C}[\![G]\!] \circ \varphi_\Delta^{-1} \qquad \mathcal{C}[\![G]\!] = \varphi_\rho^{-1} \circ \mathcal{EC}[\![G]\!] \circ \varphi_\Delta$$

By definition, $T^{\mathcal{C}}_{P:\Delta}(\alpha)(x) = \mathcal{C}[\![P(x)]\!](\alpha)$ and $T^{\mathcal{EC}}_{P:\Delta}(\beta)(x) = \mathcal{EC}[\![P(x)]\!](\beta)$ for goal term $P(x)$, so that we can deduce the following from the above equalities:

$$T^{\mathcal{C}}_{P:\Delta}(\varphi_\Delta^{-1}(\beta)) = \varphi_\rho^{-1}(T^{\mathcal{EC}}_{P:\Delta}(\beta)), \qquad T^{\mathcal{EC}}_{P:\Delta}(\varphi_\Delta(\alpha)) = \varphi_\rho(T^{\mathcal{C}}_{P:\Delta}(\alpha))$$

For part (i), observe that $T^{\mathcal{EC}}_{P:\Delta}(\beta) \sqsubseteq \beta$ (which holds because β is a prefixed point of $T^{\mathcal{EC}}_{P:\Delta}$) implies $\varphi_\Delta^{-1}(T^{\mathcal{EC}}_{P:\Delta}(\beta)) \sqsubseteq \varphi_\Delta^{-1}(\beta)$ by monotonicity. By the above equalities, we then have $T^{\mathcal{C}}_{P:\Delta}(\varphi_\Delta^{-1}(\beta)) \sqsubseteq \varphi_\Delta^{-1}(\beta)$. Thus, $\varphi_\Delta^{-1}(\beta)$ is indeed a prefixed point of $T^{\mathcal{C}}_{P:\Delta}$. Part (ii) is analogous. □

Corollary 3.29. *The HoCHC problem $\langle \Delta, P, G\rangle$ is solvable under the continuous interpretation if and only if it is solvable under the effectively continuous interpretation.*

Chapter 4

Higher-order model checking via HoCHC

Any new approach to (higher-order) program verification prompts the question how this new approach relates to existing techniques. One of the most promising forms of higher-order program verification has been *higher-order model checking* (HoMC, see Chapter 2.3).

Given a higher-order recursion scheme (HoRS) $\mathcal{G}$ and a property φ expressed in some logic, the higher-order model checking problem seeks to answer the question whether the (potentially infinite) tree generated by $\mathcal{G}$ satisfies φ.

Higher-order model checking is known to be decidable for monadic second-order logic, whereas HoCHC is generally only semi-decidable, given a suitable background theory. For HoCHC to be a useful fragment for higher-order program verification despite its complexity, it should ideally extend the expressivity of established approaches such as HoMC.

The HoMC *safety problem*, where φ is a safety property expressed by (say) a deterministic trivial automaton, can be solved via a decidable fragment of HoCHC (Wagner, 2019, private communication). Of course, safety is concerned with finite prefixes of trees; we do not need to encode an entire HoRS or value tree into HoCHC. However, if we look at a bigger class of HoMC problems, such as liveness, finite prefixes no longer suffice.

Although the meaning of HoRS and (monotone or continuous) HoCHC are both given by least fixpoints over posets of trees, there is a fundamental disparity due to the orderings on the respective posets. The meaning of a HoRS is a least fixpoint

with respect to the subtree ordering; HoCHC, on the other hand, concerns logically ordered relations, which are trivially ordered with respect to data types and trees in particular.

Let us consider the simplest example of a HoRS that generates an infinite tree. This HoRS contains a single rewrite rule $S = a\,S$ and generates a^ω, the unary tree of only as, in its least fixpoint. In HoCHC, we might encode this as a relation R_S of sort $\iota \to o$ that is the characteristic function of a^ω. However, due to the discrete order on trees in HoCHC, there does not exist a HoCHC goal clause $G : \iota \to o$ such that an infinite tree arises in the least fixpoint of the ascending Kleene chain, which is a consequence of the following proposition.

Proposition 4.1. *There does not exist a non-decreasing chain $C \subseteq \mathcal{C}[\![\iota \to o]\!]$ such that $\bigsqcup C = \chi_s$, the characteristic function of $s \in \mathcal{C}[\![\iota]\!]$, and $\chi_s \notin C$.*

Proof. Let $C \subseteq \mathcal{C}[\![\iota \to o]\!]$ be an increasing chain such that $\bigsqcup C = \chi_s$ and $\chi_s \notin C$, for some $s \in \mathcal{C}[\![\iota]\!]$. This means that C is an infinite set.

Let $C = \{c_0, c_1, c_2, \ldots\}$. Because each $c_i \sqsubseteq \bigsqcup C$, $c_i(t) = 0$ for all $t \in \mathcal{C}[\![\iota]\!]$ distinct from s. Furthermore, $\bigsqcup C$ is strictly larger than c_i, so it must be that $c_i(s) = 0$.

This is a contradiction, because $\bigsqcup C = \chi_s$, not the constant false function. □

Due to this intrinsic disparity between HoRS and HoCHC, it is unlikely that HoRS can be successfully embedded into HoCHC in their entirety. However, a *coinductive* version of HoCHC does allow for an intuitive encoding of HoRS into HoCHC. We introduce this coinductive HoCHC in Section 4.1.

In fact, the open HoRS equivalence problem (i.e. whether $[\![\mathcal{G}_1]\!] = [\![\mathcal{G}_2]\!]$ for two deterministic HoRS $\mathcal{G}_1, \mathcal{G}_2$) can be reduced to semi-decidability of coinductive HoCHC, giving rise to the following theorem.

Theorem 4.44. *The HoRS equivalence problem is decidable if the HoRS-to-HoCHC encoding lives in a semi-decidable fragment of coinductive HoCHC over Maher's complete and decidable theory of trees.*

Note that we could have taken a different approach to defining the relation $R_S : \iota \to o$ corresponding to starting nonterminal S. Instead of aiming for the characteristic function, we could define a relation that captures all finite prefixes of the tree generated

by S. After all, a tree is uniquely identified by its finite prefixes. This approach, however, would not allow us to reduce the HoRS equivalence problem to HoCHC.

We deviate from the canonical way to present HoRS semantics—which we have seen in Chapter 2.3—in Section 4.2. By adopting an equivalent semantics that is closer to HoCHC, we simplify our proofs. We introduce coinductive HoCHC in Section 4.1 before presenting the encoding of HoRS into a HoCHC logic program in Section 4.3 and its correctness in Section 4.4. We provide some examples of our encoding (Section 4.5) and apply this encoding to the HoRS equivalence problem (Section 4.6).

4.1 Coinductive HoCHC

Coinductive higher-order constrained Horn clauses allow us to reason about programs with infinite data types, notably the (potentially) infinite trees generated by HoRS.

A coinductive goal clause is syntactically identical to a traditional goal clause, and a coinductive logic program to a traditional logic program. If we think of the HoCHC problem as a triple $\langle \Delta, D, G \rangle$, then the coinductive definite clauses in D are given by $\forall \overline{x}.\, G \Leftarrow X\, \overline{x}$ instead of $\forall \overline{x}.\, G \Rightarrow X\, \overline{x}$. An intuitive way of thinking about this reversal of implications is taking the *backwards closure* of the logic program of the definite clauses. As a result, a coinductive *model* becomes a postfixed point, rather than a prefixed point, of the one-step consequence operator.

Definition 4.2 (Coinductive HoCHC problem)**.** *We say that a coinductive HoCHC problem* $\langle \Delta, D, G \rangle$ *is* solvable *just if, for all models* A *of the background theory* Th*, there exists a valuation* α *of the variables in* Δ *such that* $A, \alpha \vDash D$ *and* $A, \alpha \vDash G$*.*

Note that the problem statement differs from the (inductive) HoCHC problem, where solvability requires the existence of a valuation α such that $A, \alpha \vDash D$ and $A, \alpha \nvDash G$.

In the logic program presentation of the problem, this modified problem statement is the only difference with respect to traditional HoCHC. As a consequence, existing results for the monotone interpretation carry over. A dual argument to Cathcart Burn et al. (2018)'s Lemma 5 for inductive HoCHC shows that a coinductive HoCHC problem is *solvable under the standard interpretation* iff it is *solvable under the monotone interpretation*. Furthermore:

Lemma 4.3 (Cathcart Burn et al., 2018)**.** $\mathcal{M}[\![\Delta \vdash G : \rho]\!] \in \mathcal{M}[\![\Delta]\!] \Rightarrow_m \mathcal{M}[\![\rho]\!]$ *and* $T^{\mathcal{M}}_{P:\Delta} \in \mathcal{M}[\![\Delta]\!] \Rightarrow_m \mathcal{M}[\![\Delta]\!]$.

By the Knaster-Tarski Theorem 2.2 and $\mathcal{M}[\![\Delta]\!]$ being a complete lattice, the set of fixpoints of the monotone one-step consequence operator $T^{\mathcal{M}}_{P:\Delta}$ is a complete lattice. This guarantees the existence of a greatest fixpoint of $T^{\mathcal{M}}_{P:\Delta}$. Thus, monotone HoCHC enjoys a greatest model property.

Theorem 4.4 (Greatest model property for monotone HoCHC)**.** *Under the monotone interpretation, HoCHC definite clauses (and logic programs) possess greatest models.*

Furthermore, a greatest model witnesses the solvability of a coinductive HoCHC problem in the monotone or continuous setting, as in the example above, like a least model witnesses solvability for traditional HoCHC.

Theorem 4.5. *A coinductive HoCHC problem $\langle \Delta, P, G\rangle$ is solvable under the monotone interpretation if and only if, in all models of the background theory, it holds that $\mathcal{M}[\![G]\!](M_P) = 1$ for greatest model M_P of P.*

Proof. Recall that a coinductive HoCHC problem $\langle \Delta, P, G\rangle$ is solvable under the monotone interpretation if and only if, for all models A of the background theory Th, there exists a valuation α of the variables in Δ such that $A, \alpha \vDash P$ and $A, \alpha \vDash G$.

Clearly, $\mathcal{M}[\![G]\!](M_P) = 1$ for greatest model M_P of P and all models of the background theory implies solvability of $\langle \Delta, P, G\rangle$. For the converse, let A be a model of the background theory and let $\alpha \in \mathcal{M}[\![\Delta]\!]$ be a valuation such that $A, \alpha \vDash P$ and $A, \alpha \vDash G$. By Knaster-Tarski, $\alpha \sqsubseteq M_P$. By monotonicity, $A, \alpha \vDash G$ implies $A, M_P \vDash G$, as required. □

HoCHC and coinductive HoCHC are not equivalent in the sense that (say) the standard and the monotone interpretation are, as the following example demonstrates; it is not the case that a HoCHC problem $\langle \Delta, D, G\rangle$ is solvable if and only if the corresponding coinductive HoCHC problem $\langle \Delta, D, G\rangle$ is solvable (with the implications in the definite clauses reversed).

Example 4.6. Consider the HoCHC problems given by $\mathcal{P}_{\mathsf{ind}} = \langle \{R_S\}, D_{\mathsf{ind}}, R_S\, a^\omega\rangle$ and $\mathcal{P}'_{\mathsf{ind}} = \langle \{R_S\}, D_{\mathsf{ind}}, R_S\, b^\omega\rangle$, where D_{ind} consists of:

$$\forall r.\ (\exists r_1.\ (a\, r_1 = r) \wedge R_S\, r_1) \Rightarrow R_S\, r$$

Compare to the corresponding coinductive HoCHC problems $\mathcal{P}_{\mathsf{co}} = \langle \{R_S\}, D_{\mathsf{co}}, R_S\, a^\omega \rangle$ and $\mathcal{P}'_{\mathsf{co}} = \langle \{R_S\}, D_{\mathsf{co}}, R_S\, b^\omega \rangle$, where D_{co} consists of:

$$\forall r.\ (\exists r_1.\ (a\, r_1 = r) \wedge R_S\, r_1) \Leftarrow R_S\, r$$

Both problems have the same two models of the definite clauses. The relational variable R_S corresponds to the empty set in least model $M_\emptyset$, and to the singleton set $\{a^\omega\}$ in greatest model $M_{\{a^\omega\}}$.

Both $\mathcal{P}_{\mathsf{ind}}$ and $\mathcal{P}'_{\mathsf{ind}}$ are solvable, since $M_\emptyset$ is a witness to the refutation of both goal clauses. However, $\mathcal{P}_{\mathsf{co}}$ is solvable, while $\mathcal{P}'_{\mathsf{co}}$ is unsolvable. This follows from $M_{\{a^\omega\}}$ witnessing the acceptance of goal clause $R_S\, a^\omega$, while neither $M_{\{a^\omega\}}$ nor $M_\emptyset$ witnesses the acceptance of $R_S\, b^\omega$.

Coinductive HoCHC is not merely an academic indulgence. There is a tradition of coinduction and corecursion in logic programming (see Komendantskaya and Li, 2017; Gupta et al., 2007; Simon et al., 2007; Jaffar and Stuckey, 1986). This is hardly surprising, given that some well-formed logic programs do not have natural inductive interpretations. Examples of infinite data types that arise in practice—besides infinite trees—include infinite lists and streams (which are unary trees). Let us consider such an example.

Example 4.7. Given two individual sorts, sort $\mathbb{N}$ of natural numbers and sort $[\mathbb{N}]$ of lists over natural numbers, we define the following logic program:

$$\begin{aligned}
\mathsf{zeros} &= \lambda \ell.\ \exists \mathsf{tl}.\ (\ell = [0 \mid \mathsf{tl}] \wedge \mathsf{zeros}\, \mathsf{tl}) \\
\mathsf{incr} &= \lambda \ell.\ \exists n\, \mathsf{tl}\, \mathsf{tl}'.\ (\ell = [n \mid \mathsf{tl}] \wedge \mathsf{tl} = [n+1 \mid \mathsf{tl}'] \wedge \mathsf{incr}\, \mathsf{tl}) \\
\mathsf{map} &= \lambda f\, \ell_1\, \ell_2.\ \exists n_{1,2}\, \mathsf{tl}_{1,2}.\ (\ell_1 = [n_1 \mid \mathsf{tl}_1] \wedge \ell_2 = [n_2 \mid \mathsf{tl}_2] \wedge f\, n_1\, n_2 \wedge \mathsf{map}\, f\, \mathsf{tl}_1\, \mathsf{tl}_2)
\end{aligned}$$

where $\mathsf{zeros} : [\mathbb{N}] \to o$, $\mathsf{incr} : [\mathbb{N}] \to o$, and $\mathsf{map} : (\mathbb{N} \to \mathbb{N} \to o) \to [\mathbb{N}] \to [\mathbb{N}] \to o$.

Clearly, this logic program has no natural inductive interpretation. None of the clauses contain base cases to break out of the recursion, which means that the least model is empty. It is, however, intuitive to interpret them coinductively.

The first predicate is then interpreted as the characteristic function of the infinite list of zeros. The second predicate holds for all infinite lists of consecutive, increasing natural numbers. Finally, the predicate map becomes the relational representation of the usual map on (infinite) lists: its second list argument is the result of applying a function f to each argument of the first list.

4.1.1 Relation to Herbrand models

Herbrand models play a key role in the foundations of logic programming. A *Herbrand model* is an interpretation of a logic program characterised by the fact that constants—for us the first-order symbols from signature Σ—are interpreted as 'themselves'.

This notion is more common in *intensional* logic programming than in *extensional* logic programming like HoCHC (see Chapter 7.1.2 for a comparison). In intensional logic programming, models are *term models*, sets of terms, rather than elements from a fixed semantic domain. There, it makes sense to interpret a symbol as itself.

The remainder of this chapter considers HoCHC with a sort of individuals interpreted as the set of finite and infinite trees. Because trees can be thought of as terms, we use an extensional 'Herbrand' interpretation (see Section 4.2 and 4.6). Our 'Herbrand' interpretation is *complete*; it is allowed to contain infinite trees.

4.2 HoRS semantics

Although we have already defined higher-order recursion schemes and the trees they generate in Chapter 2.3, here we present an equivalent semantics that is closer to HoCHC and, thus, simplifies our proofs.

We fix a ranked alphabet Σ and write $\Sigma_\bot$ for $\Sigma \cup \{\bot\}$. The set of all finite and infinite $\Sigma_\bot$-labelled trees is denoted by $\mathcal{T}_{\Sigma_\bot}$. A tree context $C[\_] \in \mathcal{T}_{\{\_\} \cup \Sigma_\bot}$ is simply a tree over $\Sigma_\bot$ that contains some occurrences of a 'hole'; if we substitute a tree $t \in \mathcal{T}_{\Sigma_\bot}$ into the hole, we obtain a tree $C[t] \in \mathcal{T}_{\Sigma_\bot}$.

In the sequel, we consider $\mathcal{T}_{\Sigma_\bot}$ as a pointed poset with least element $\bot$ over the subtree ordering $\sqsubseteq$, which is defined as the least partial order such that $C[\bot] \sqsubseteq C[t]$ for every tree context $C[\_] \in \mathcal{T}_{\{\_\} \cup \Sigma_\bot}$ and every $t \in \mathcal{T}_{\Sigma_\bot}$.

Let $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$ be a deterministic higher-order recursion scheme (HoRS) as defined in Chapter 2.3. That is, $\mathcal{N}$ maps a nonterminal symbol to its sort, Σ maps a terminal symbol to its sort, $S \in \mathcal{N}$ is the designated start symbol, and there is one rewrite rule in $\mathcal{R}$ for each $F \in \mathcal{N}$ such that

$$\mathcal{R}(F) = \lambda x_1 \dots x_n . t,$$

where $t : \iota$ is an applicative term over $\mathcal{N} \cup \Sigma \cup \{x_1, \dots, x_n\}$ such that $x_1, \dots, x_n \in V_{\mathrm{RS}}$ for some finite set of distinct recursion scheme variables V_{RS}. We refer to any subterm

of $\lambda x_1 \ldots x_n.\, t$ as a *HoRS term* to distinguish it from HoCHC terms, which are terms of higher-order logic.

We assume that the tree generated by $\mathcal{G}$ does not contain $\bot$. This assumption is without loss of generality for our purpose, as we shall see in Section 4.6.2.

The meaning of a HoRS can be given by a number of different formalisms. We introduce an 'infinite Herbrand interpretation' that treats the rewrite rules as definitional equality in the style of a HoCHC logic program. Models are built incrementally from the smallest tree $\bot$. We adopt the typical Herbrand feature that constants and function symbols are assigned very simple meanings (the semantic counterpart of 'itself', as for term models). Because our models may contain infinite terms, our interpretation of HoRS is essentially the least complete Herbrand model (Levi et al., 1987).

Let us define an interpretation of the sorts over ι (i.e. the sorts of HoRS terms):

$$\mathcal{H}[\![\iota]\!] := \langle \mathcal{T}_{\Sigma_\bot}, \sqsubseteq \rangle \qquad \mathcal{H}[\![\sigma_1 \to \sigma_2]\!] := \mathcal{H}[\![\sigma_1]\!] \Rightarrow_c \mathcal{H}[\![\sigma_2]\!]$$

where $X \Rightarrow_c Y$ is the continuous function space between dcpos X and Y with respect to the subtree order $\sqsubseteq$ on $\mathcal{H}[\![\iota]\!]$, which is a non-trivial order if Σ is non-empty.

Lemma 4.8. *Each $\mathcal{H}[\![\sigma]\!]$ is an algebraic dcpo.*

Proof. The set of finite trees over $\Sigma_\bot$ is a countable pointed poset. There exists an isomorphism between the set of ideals over this poset (ordered by inclusion) and our poset $\mathcal{T}_{\Sigma_\bot}$ of finite and infinite trees. By appealing to this isomorphism, Theorem 2.1 tells us that $\langle \mathcal{T}_{\Sigma_\bot}, \sqsubseteq \rangle$ is an ω-algebraic dcpo. A straightforward inductive argument with Proposition 2.18 gives us that $\mathcal{H}[\![\sigma]\!]$ is an algebraic dcpo for all sorts σ over ι. □

Given the environment $\Gamma = \{x_1 : \tau_1, \ldots, x_k : \tau_k\}$, let $\mathcal{N}'$ denote the extended environment $\mathcal{N}, \Gamma := \mathcal{N} \cup \Gamma$, which we view as a function with domain $\mathsf{dom}(\mathcal{N}) \cup \{x_1, \ldots, x_k\}$, mapping each symbol to its sort. Set $\mathcal{H}[\![\mathcal{N}']\!] : \prod x \in \mathsf{dom}(\mathcal{N}').\, \mathcal{H}[\![\mathcal{N}'(x)]\!]$ with typical element α. This gives rise to a function

$$\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}'} : \mathcal{H}[\![\mathcal{N}']\!] \Rightarrow \mathcal{H}[\![\mathcal{N}']\!], \qquad \mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}'}(\alpha)(F) := \mathcal{H}[\![\mathcal{N}' \vdash \mathcal{R}(F) : \mathcal{N}(F)]\!](\alpha)$$

where each

$$\mathcal{H}[\![\mathcal{N}' \vdash t : \sigma]\!] : \mathcal{H}[\![\mathcal{N}']\!] \Rightarrow \mathcal{H}[\![\sigma]\!]$$

is defined by cases and recursion on syntax:

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash x]\!](\alpha) &= \alpha(x) \\
\mathcal{H}[\![\mathcal{N}' \vdash f]\!](\alpha) &= \widehat{F_f} \\
\mathcal{H}[\![\mathcal{N}' \vdash s\,t]\!](\alpha) &= \mathcal{H}[\![\mathcal{N}' \vdash s]\!](\alpha)(\mathcal{H}[\![\mathcal{N}' \vdash t]\!](\alpha)) \\
\mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma_1.\, t : \sigma_2]\!](\alpha) &= \lambda v \in \mathcal{H}[\![\sigma_1]\!].\, \mathcal{H}[\![\mathcal{N}', x : \sigma_1 \vdash t]\!](\alpha[x \mapsto v])
\end{aligned}$$

for $\widehat{F_f}$ the usual Herbrand interpretation of $f \in \Sigma$.

A fixpoint of $\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}'} : \mathcal{H}[\![\mathcal{N}']\!] \Rightarrow \mathcal{H}[\![\mathcal{N}']\!]$ is a *model* of $\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}'}$.

In the sequel, we prove that our semantics is well-defined. In particular, it is continuous and has a least model that captures the meaning of the input HoRS (see Theorem 4.12).

Lemma 4.9. *$\widehat{F_f} \in \mathcal{H}[\![\iota^{\mathrm{ar}(f)} \to \iota]\!]$ for all $f : \iota^{\mathrm{ar}(f)} \to \iota \in \Sigma$.*

Proof. This is straightforward from the subtree ordering. It follows that $C[s] \sqsubseteq C[t]$ for all tree contexts $C[\_] \in \mathcal{T}_{\{\_\}\cup\Sigma_\perp}$ and trees $s, t \in \mathcal{T}_{\Sigma_\perp} = \mathcal{H}[\![\iota]\!]$ such that $s \sqsubseteq t$. We can extend this inductively to multi-holed tree contexts: $C[s_1, \ldots, s_m] \sqsubseteq C[t_1, \ldots, t_m]$ for all tree contexts $C[\_] \in \mathcal{T}_{\{\_\}\cup\Sigma_\perp}$ and trees $s_i, t_i \in \mathcal{H}[\![\iota]\!]$ such that $s_i \sqsubseteq t_i$ for all $i \in [m]$. Monotonicity of $\widehat{F_f}$ follows directly.

To show continuity, let $D_1, \ldots, D_{\mathrm{ar}(f)}$ be directed sets in $\mathcal{H}[\![\iota]\!]$. Using a straightforward inductive argument, we can show that the least upper bound of the set

$$\{\widehat{F_f}\, d_1 \ldots d_{\mathrm{ar}(f)} \mid d_1 \in D_1, \ldots, d_{\mathrm{ar}(f)} \in D_{\mathrm{ar}(f)}\}$$

is

$$\widehat{F_f}\left(\bigsqcup D_1\right) \ldots \left(\bigsqcup D_{\mathrm{ar}(f)}\right),$$

and $\widehat{F_f}$ is continuous. □

Lemma 4.10. *For all HoRS terms $\mathcal{N}' \vdash t : \sigma$, (1) $\mathcal{H}[\![\mathcal{N}' \vdash t : \sigma]\!](\alpha)$ is continuous for all $\alpha \in \mathcal{H}[\![\mathcal{N}']\!]$, and (2) $\mathcal{H}[\![\mathcal{N}' \vdash t : \sigma]\!] : \mathcal{H}[\![\mathcal{N}']\!] \to \mathcal{H}[\![\sigma]\!]$ is continuous.*

Proof. We proceed by structural induction on t. Let $\alpha, \beta \in \mathcal{H}[\![\mathcal{N}']\!]$ such that $\alpha \sqsubseteq \beta$, and let $\mathcal{I} \subseteq \mathcal{H}[\![\mathcal{N}']\!]$ be a directed set of valuations.

Case $t = x : \sigma \in \mathcal{N}'$. Note that $\alpha \in \mathcal{H}[\![\mathcal{N}']\!]$ implies

$$\mathcal{H}[\![\mathcal{N}' \vdash x]\!](\alpha) = \alpha(x) \in \mathcal{H}[\![\sigma]\!]$$

and thus (1). For (2),

$$\mathcal{H}[\![\mathcal{N}' \vdash x]\!](\alpha) = \alpha(x) \sqsubseteq \beta(x) = \mathcal{H}[\![\mathcal{N}' \vdash x]\!](\beta)$$

and

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash x]\!] \left(\bigsqcup \mathcal{I}\right) &= \left(\bigsqcup \mathcal{I}\right)(x) \\
&= \bigsqcup_{I \in \mathcal{I}} I(x) \qquad \text{Prop 2.8, Lem 4.8} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash x]\!](I).
\end{aligned}$$

Case $t = f : \sigma \in \Sigma$**.** Lemma 4.9 gives us the first claim: $\mathcal{H}[\![\mathcal{N}' \vdash f]\!](\alpha) = \widehat{F_f} \in \mathcal{H}[\![\sigma]\!]$. For the second claim, observe that the meaning $\mathcal{H}[\![\mathcal{N}' \vdash f]\!](\alpha)$ is independent of the valuation α.

Case $t = s\,t'$**.** The first claim follows directly from the IH: $\mathcal{H}[\![\mathcal{N}' \vdash s]\!](\alpha) \in \mathcal{H}[\![\sigma \to \tau]\!]$ and $\mathcal{H}[\![\mathcal{N}' \vdash t']\!](\alpha) \in \mathcal{H}[\![\sigma]\!]$ for some sorts σ and τ. For the second claim,

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash s\,t']\!](\alpha) &= \mathcal{H}[\![\mathcal{N}' \vdash s]\!](\alpha)(\mathcal{H}[\![\mathcal{N}' \vdash t']\!](\alpha)) \\
&\sqsubseteq \mathcal{H}[\![\mathcal{N}' \vdash s]\!](\alpha)(\mathcal{H}[\![\mathcal{N}' \vdash t']\!](\beta)) \qquad \text{IH} \\
&\sqsubseteq \mathcal{H}[\![\mathcal{N}' \vdash s]\!](\beta)(\mathcal{H}[\![\mathcal{N}' \vdash t']\!](\beta)) \qquad \text{IH} \\
&= \mathcal{H}[\![\mathcal{N}' \vdash s\,t']\!](\beta)
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash s\,t']\!] \left(\bigsqcup \mathcal{I}\right) &= \mathcal{H}[\![\mathcal{N}' \vdash s]\!] \left(\bigsqcup \mathcal{I}\right) \left(\mathcal{H}[\![\mathcal{N}' \vdash t']\!] \left(\bigsqcup \mathcal{I}\right)\right) \\
&= \left(\bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash s]\!](I)\right) \left(\bigsqcup_{J \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash t']\!](J)\right) \qquad \text{IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash s]\!](I) \left(\bigsqcup_{J \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash t']\!](J)\right) \qquad \text{Prop 2.8} \\
&= \bigsqcup_{I,J \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash s]\!](I) \left(\mathcal{H}[\![\mathcal{N}' \vdash t']\!](J)\right) \qquad \text{Prop 2.4, IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash s]\!](I) \left(\mathcal{H}[\![\mathcal{N}' \vdash t']\!](I)\right) \qquad \text{Prop 2.10} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash s\,t']\!](I).
\end{aligned}$$

Case $t = \lambda x : \sigma.\, s$**.** For (1), let $z_1, z_2 \in \mathcal{H}[\![\sigma]\!]$ such that $z_1 \sqsubseteq z_2$. Then,

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](\alpha)\, z_1 &= \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](\alpha[x \mapsto z_1]) \\
&\sqsubseteq \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](\alpha[x \mapsto z_2]) && \text{IH} \\
&= \mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](\alpha)\, z_2.
\end{aligned}$$

Let $D \subseteq \mathcal{H}[\![\sigma]\!]$ be a directed set. Then,

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](\alpha) \left(\bigsqcup D\right) &= \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!] \left(\alpha \left[x \mapsto \bigsqcup D\right]\right) \\
&= \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!] \left(\bigsqcup_{d \in D} \alpha[x \mapsto d]\right) \\
&= \bigsqcup_{d \in D} \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](\alpha[x \mapsto d]) && \text{IH} \\
&= \bigsqcup_{d \in D} \mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](\alpha)\, d.
\end{aligned}$$

For (2),

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](\alpha) &= \lambda v.\, \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](\alpha[x \mapsto v]) \\
&\sqsubseteq \lambda v.\, \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](\beta[x \mapsto v]) && \text{IH} \\
&= \mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](\beta)
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!] \left(\bigsqcup \mathcal{I}\right) &= \lambda v.\, \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!] \left(\left(\bigsqcup \mathcal{I}\right)[x \mapsto v]\right) \\
&= \lambda v.\, \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!] \left(\bigsqcup_{I \in \mathcal{I}} I[x \mapsto v]\right) \\
&= \lambda v.\, \left(\bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](I[x \mapsto v])\right) && \text{IH} \\
&= \bigsqcup_{I \in \mathcal{I}} \lambda v.\, \mathcal{H}[\![\mathcal{N}', x : \sigma \vdash s]\!](I[x \mapsto v]) && \text{Prop 2.4, 2.8} \\
&= \bigsqcup_{I \in \mathcal{I}} \mathcal{H}[\![\mathcal{N}' \vdash \lambda x : \sigma.\, s]\!](I).
\end{aligned}$$

This concludes the proof. □

Corollary 4.11. *$\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}}$ is continuous for all deterministic HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$.*

Theorem 4.12. *There exists a least model of $\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}}$, for all deterministic HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$.*

Proof. By Corollary 4.11 and Kleene's Fixed-Point Theorem. □

Thus, $\mathsf{lfp}(\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}})(S)$ denotes the meaning of $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$, written $[\![\mathcal{G}]\!]$.

4.3 Encoding HoRS into a HoCHC logic program

In this section, we encode a HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$ into a HoCHC logic program that captures the meaning of the HoRS under the coinductive monotone interpretation. We assume that $[\![\mathcal{G}]\!]$ does not contain $\bot$, which is WLOG by Section 4.6.2.

$\bot$-freeness assumption. This assumption allows us to distinguish 'finished' trees from 'unfinished' trees; whenever a tree contains $\bot$, we know this a partial computation of a tree $[\![\mathcal{G}]\!]$, not the entire tree. In more technical terms, in Definition 4.20 we define an embedding $\mathbf{i}_\iota : \mathcal{H}[\![\iota]\!] \to \mathcal{M}[\![\iota \to o]\!]$ of trees into relations by $\mathbf{i}_\iota(t) = \lambda s.\, t \sqsubseteq s$, for all trees $t \in \mathcal{H}[\![\iota]\!]$. This embedding is key to our correctness proof. We rely on the fact that $\mathbf{i}_\iota(t)$ becomes the characteristic function of t at the end of the ascending Kleene chain of HoRS semantics.

Example 4.13. Given a HoRS $\mathcal{G} = \langle \{S\}, \{a\}, \{S = a\, S\}, S \rangle$, the ascending Kleene chain looks like this, with the trees written as terms:

$$\bot \sqsubseteq a\, \bot \sqsubseteq a\, (a\, \bot) \sqsubseteq a\, (a\, (a\, \bot)) \sqsubseteq \dots$$

The limit of this chain is a^ω, the infinite tree of as, for which it holds that

$$\mathbf{i}_\iota(a^\omega) = \lambda s.\, (a^\omega \sqsubseteq s) = \lambda s.\, (a^\omega = s),$$

because a^ω does not contain $\bot$. Thus, $\bot$-freeness allows us to capture characteristic functions of infinite trees generated by HoRS.

Determinism assumption. Embedding trees (of sort ι) into HoCHC relations of sort $\iota \to o$ allows us to define a relational variable $R_S : \iota \to o$ that holds of precisely the trees generated by a particular HoRS (in the greatest model). If HoRS $\mathcal{G}_1$ and $\mathcal{G}_2$ are deterministic, they each generate exactly one tree, and $R_{S_1} : \iota \to o$ and $R_{S_2} : \iota \to o$ are the characteristic functions of those respective trees. This allows us to query the existence of two trees t_1, t_2 such that $R_{S_1}\, t_1 \wedge R_{S_2}\, t_2$. Because $\mathcal{G}_1$ and

$\mathcal{G}_2$ are deterministic, these trees t_1 and t_2 are unique, so that $t_1 = t_2$ corresponds to equivalence of the original HoRS and $t_1 \neq t_2$ to inequivalence.

If the HoRS are not deterministic, then these queries become meaningless: the existence of t_1, t_2 such that $R_{S_1}\, t_1 \wedge R_{S_2}\, t_2 \wedge t_1 = t_2$ would merely mean $\mathcal{G}_1$ and $\mathcal{G}_2$ have a tree in common, while $R_{S_1}\, t_1 \wedge R_{S_2}\, t_2 \wedge t_1 \neq t_2$ would mean nothing at all. Thus, determinism is key to defining the positive and negative HoCHC instance, as in Section 4.6.3, for 'deciding' the HoRS equivalence problem (Theorem 4.44).

The relational lift on sorts. We define a constrained logic program $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$ over the coinductive monotone HoCHC interpretation where the set $\mathcal{M}[\![\iota]\!]$ of individuals is interpreted as the set of finite and infinite trees, which is the underlying set of $\mathcal{H}[\![\iota]\!] = \langle \mathcal{T}_{\Sigma_\perp}, \sqsubseteq \rangle$.

Intuitively, our constrained logic program is a continuation-passing style encoding of an input HoRS, where functions are embedded in predicates. This logic program contains one relational variable R_F (a predicate) of arity $n+1$ for each HoRS nonterminal $F \in \mathcal{N}$ (a function) of arity n.

We lift HoRS terms of sort σ to HoCHC terms of sort $\mathsf{Rel}^+(\sigma)$ using a *relational lift* on sorts and terms. The lifted sort $\mathsf{Rel}^+(\sigma)$ has an additional 'result' argument compared to σ (the propositional o in tail position). Additionally, the arguments of σ of order 1 and higher are lifted hereditarily, while arguments of sort ι are unaffected by the relational lift; hence $\mathsf{Rel}^-(\tau_1 \to \tau_2) = \mathsf{Rel}^+(\tau_1 \to \tau_2)$ but $\mathsf{Rel}^-(\iota) = \iota$.

Definition 4.14 (Relational lift on sorts)**.** *Given HoRS sort σ, we call $\mathsf{Rel}^+(\sigma)$ the* relational lift *of σ, defined by:*

$$\mathsf{Rel}^-(\iota) := \iota \qquad \mathsf{Rel}^-(\sigma_1 \to \sigma_2) := \mathsf{Rel}^-(\sigma_1) \to \mathsf{Rel}^+(\sigma_2)$$
$$\mathsf{Rel}^+(\iota) := \iota \to o \qquad \mathsf{Rel}^+(\sigma_1 \to \sigma_2) := \mathsf{Rel}^-(\sigma_1) \to \mathsf{Rel}^+(\sigma_2)$$

Example 4.15. A HoRS terminal symbol $f : \iota^{\mathrm{ar}(f)} \to \iota$ is lifted to a HoCHC term of sort $\mathsf{Rel}^+(\iota^{\mathrm{ar}(f)} \to \iota) = \iota^{\mathrm{ar}(f)} \to \iota \to o$. For higher-order HoRS terms, arguments are also lifted. E.g. the sort $\iota \to (\iota \to \iota) \to \iota$ is lifted to:

$$\begin{aligned}\mathsf{Rel}^+(\iota \to (\iota \to \iota) \to \iota) &= \mathsf{Rel}^-(\iota) \to \mathsf{Rel}^-(\iota \to \iota) \to \mathsf{Rel}^+(\iota) \\ &= \iota \to \mathsf{Rel}^+(\iota \to \iota) \to \iota \to o \\ &= \iota \to (\iota \to \iota \to o) \to \iota \to o\end{aligned}$$

The relational lift on terms. The constrained logic program $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$ is defined by

$$\Delta_{\mathcal{G}} := \left\{ R_F : \mathsf{Rel}^+(\sigma) \mid F : \sigma \in \mathcal{N} \right\}$$
$$P_{\mathcal{G}} := \left\{ R_F : \mathsf{Rel}^+(\sigma) = \ulcorner \mathcal{R}(F) \urcorner \mid F : \sigma \in \mathcal{N} \right\}$$

where $\ulcorner - \urcorner$ denotes the relational lift on terms of $\mathcal{R}(\mathcal{F})$ as defined in Definition 4.17. The following example gives an intuition of how the relational lift operates on HoRS terms.

Example 4.16. The first-order HoRS rewrite rule $S : \iota = I\,(G\,\mathsf{z})$ is mapped to the following HoCHC definition in the constrained logic program:

$$R_S : \iota \to o = \lambda r.\, \exists r_1\, r_2.\, R_I\, r_1\, r \wedge R_G\, r_2\, r_1 \wedge (\mathsf{z} = r_2)$$

Note that each HoRS subterm is represented by conjunct whose (existentially quantified) arguments r_i are bound by a subsequent conjunct. Now consider a higher-order rewrite rule $H : (\iota \to \iota) \to \iota = \lambda\varphi.\, \mathsf{s}\,(F\,\varphi\,\mathsf{z})$, which is mapped to:

$$R_H : (\iota \to \iota \to o) \to \iota \to o = \lambda r.\, \exists r_1\, r_2\, r_3.\, (\mathsf{s}\, r_1 = r) \wedge R_F\,(\lambda y\, r'.\, \varphi'\, y\, r')\, r_3\, r_1 \wedge (\mathsf{z} = r_3)$$

Now we see that r_i of sort ι are indeed bound by subsequent conjuncts, while r_i of higher-order sorts are redundant; instead, we in-line the lift of such a higher-order argument, like φ above.

In the HoRS-to-HoCHC encoding below, we annotate variables with superscripts of not merely sorts but of interpreted sorts—$\mathcal{H}[\![\sigma]\!]$ or $\mathcal{M}[\![\sigma]\!]$ for sort σ—to distinguish HoRS and HoCHC variables.

Definition 4.17 (HoRS-to-HoCHC encoding)**.** *Let the metavariable* $\$: \tau_1 \to \cdots \to \tau_l \to \sigma_1 \to \cdots \to \sigma_n \to \iota$ *range over* $\mathcal{N} \cup \Sigma \cup V_{RS}$*. We define a transformation* $\$ \mapsto \$'$ *according to the following table:*

	$\$$	$\$'$
Variables	$x : \iota$	$D_{x'} : \mathsf{Rel}^+(\iota)$
	$x : \sigma_1 \to \sigma_2$	$x' : \mathsf{Rel}^+(\sigma_1 \to \sigma_2)$
Terminals	$f : \iota^n \to \iota$	$D_f : \mathsf{Rel}^+(\iota^n \to \iota)$
Nonterminals	$F : \sigma$	$R_F : \mathsf{Rel}^+(\sigma)$

where

$$D_f := \lambda x_1 \ldots x_{\mathrm{ar}(f)} r.\, (f\, x_1 \ldots x_{\mathrm{ar}(f)} = r).$$

Note that R_F is a relational variable, but D_f is merely a shorthand; D_f is not a symbol—and neither is $D_{x'}$, which is defined analogously. This shorthand allows us to present the relational lift $\ulcorner - \urcorner$ on terms in a simpler way. Whenever D_f occurs in some $\ulcorner e \urcorner$, it occurs in a fully applied term $D_f\, t_1 \dots t_{\mathrm{ar}(f)}\, r$, which is β-equivalent to $f\, t_1 \dots t_{\mathrm{ar}(f)} = r$. It is this latter term we use in practice (and similarly for $D_{x'}$).

For each $F : \sigma \in \mathcal{N}$, we define $\ulcorner \mathcal{R}(F) \urcorner : \mathsf{Rel}^+(\sigma)$, called the relational lift on terms*, in the following way. We write $x' : \mathsf{Rel}^-(\tau)$ for the HoCHC variable that is the relational clone of HoRS variable $x : \tau \in V_{\mathrm{RS}}$, such that:*

$$\ulcorner \lambda x_1^{\mathcal{H}[\![C_1]\!]} \dots x_m^{\mathcal{H}[\![C_m]\!]}.\, e \urcorner := \lambda {x'}_1^{\mathcal{M}[\![\mathsf{Rel}^-(C_1)]\!]} \dots {x'}_m^{\mathcal{M}[\![\mathsf{Rel}^-(C_m)]\!]}.\, \ulcorner e \urcorner$$

For HoRS term $\$\, e_1 \dots e_l$, we define the relational lift as:

$$\ulcorner \$\, e_1 \dots e_l \urcorner := \begin{array}{l} \lambda y_1^{\mathcal{M}[\![\mathsf{Rel}^-(\sigma_1)]\!]} \dots y_n^{\mathcal{M}[\![\mathsf{Rel}^-(\sigma_n)]\!]}\, r. \\ \quad \exists r_1 \dots r_l. \left(\begin{array}{ll} & \$' \urcorner (e_1, r_1) \urcorner \dots \urcorner (e_l, r_l) \urcorner\, y_1 \dots y_n\, r \\ \wedge & \bigwedge_{i=1}^{l} \mathit{Prop}(e_i, r_i) \end{array} \right) \end{array}$$

with fresh HoCHC variables $y_1, \dots, y_n$, where

$$\urcorner (e : \sigma, r) \ulcorner := \begin{cases} r & \text{if } \sigma = \iota \\ \ulcorner e : \sigma \urcorner & \text{O/W} \end{cases} \qquad \mathit{Prop}(e : \sigma, r) := \begin{cases} \ulcorner e : \sigma \urcorner\, r & \text{if } \sigma = \iota \\ \mathsf{true} & \text{O/W.} \end{cases}$$

It is worth pointing out the sorts of the following terms:

$$\ulcorner e : \sigma \urcorner : \mathsf{Rel}^+(\sigma) \qquad \urcorner (e : \sigma, r) \ulcorner : \mathsf{Rel}^-(\sigma) \qquad \mathit{Prop}(e : \sigma, r) : o$$

It is known as a general folklore result that imperative or functional programs can be represented as logic programs akin to continuation-passing style. The above definition provides a systematic way of doing this for higher-order functions without increasing the order (except for order-0 HoRS, which yield order-1 logic programs).

Note there is nothing intrinsically coinductive about the HoRS-to-HoCHC encoding. It is easily adapted to reason about finite prefixes of a tree or about HoRS that generate finite trees, so that the encoding can be used in the inductive HoCHC setting too (as in Example 5.26).

Example 4.18. Consider the order-2 HoRS given by $\mathcal{G} = \langle \{S, F, B\}, \{\mathsf{c}, \mathsf{s}, \mathsf{z}\}, \mathcal{R}, S \rangle$ and the following rewrite rules in $\mathcal{R}$:

$$\begin{aligned} S : \iota &= F\, \mathsf{s} \\ F : (\iota \to \iota) \to \iota &= \lambda \varphi.\, \mathsf{c}\, (\varphi\, \mathsf{z})\, (F\, (B\, \varphi\, \varphi)) \\ B : (\iota \to \iota) \to (\iota \to \iota) \to \iota &= \lambda \varphi\, \psi\, x.\, \varphi\, (\psi\, x) \end{aligned}$$

This is transformed into the following logic program:

$$R_S = \lambda r.\, R_F\,(\lambda y\, r'.\, \mathsf{s}\, y = r')\, r$$
$$R_F = \lambda \varphi'\, r.\, \exists r_1\, r_2\, r_3.\, (\mathsf{c}\, r_1\, r_2 = r) \wedge \varphi'\, r_3\, r_1 \wedge (\mathsf{z} = r_3) \wedge R_F\,(\lambda y\, r'.\, R_B\, \varphi'\, \varphi'\, y\, r')\, r_2$$
$$R_B = \lambda \varphi'\, \psi'\, x'\, r.\, \exists r_1\, r_2.\, \varphi'\, r_1\, r \wedge \psi'\, r_2\, r_1 \wedge (x' = r_2)$$

Again, the intuition is that each HoRS subterm is represented by a conjunct whose arguments are existentially quantified r_i bound by a subsequent conjunct (if a tree) or in-lined (if higher sort).

Further examples of encoded HoRS, the trees they generate, and their correctness are presented in Section 4.5.

4.4 Correctness

Our encoded HoRS-to-HoCHC logic program $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$ contains a relational variable for each nonterminal symbol in the original HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$. We claim that the HoCHC variable $R_S : \iota \to o$ corresponding to HoRS start symbol $S : \iota$ valuates to the characteristic function of $[\![\mathcal{G}]\!]$ in the greatest model of $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$:

Theorem 4.32. *$\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash R_S]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}))\, t = 1$ if and only if $t = [\![\mathcal{G}]\!]$.*

To prove the correctness of the HoRS-to-HoCHC encoding, we establish a lockstep between iterations of the HoRS function $\mathcal{H}[\![\mathcal{G}]\!]_{\mathcal{N}}$ (in the ascending Kleene chain) and iterations of the HoCHC one-step consequence operator $T^{\mathcal{M}}_{\Delta_{\mathcal{G}}:P_{\mathcal{G}}}$.

The proof consists of four parts. First, we define two pairs of mappings between HoRS semantics and (coinductive) HoCHC semantics in Section 4.4.1. These mappings allow us to embed HoRS semantics into HoCHC. Second, in Section 4.4.2 we show that there exists a $\bot$-free tree t such that $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](T^{\mathcal{M}\,n}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}(\top_{\Delta_{\mathcal{G}}}))\, t = 1$, for every iteration n of the one-step consequence operator. Third, we show that each $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](T^{\mathcal{M}\,n}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}(\top_{\Delta_{\mathcal{G}}}))$ is included in the embedding of $\mathcal{H}[\![\mathcal{N} \vdash S]\!](\mathcal{H}[\![\mathcal{G}]\!]^n_{\mathcal{N}}(\bot_{\mathcal{N}}))$ into HoCHC, in Section 4.4.3, which is our lockstep.

Finally, in Section 4.4.4 we prove that these 'nonemptiness' and 'inclusion' results suffice to that show that R_S valuates to the characteristic function of $[\![\mathcal{G}]\!]$ in the greatest model of $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$ (Theorem 4.32).

4.4.1 Mappings

Definition 4.19 (Relatively monotone sort frame)**.** *For each sort σ over ι, we define*

$$\mathsf{Im}_\sigma := \{\theta \in \mathcal{D}[\![\mathsf{Rel}^-(\sigma)]\!] \mid \exists h \in \mathcal{H}[\![\sigma]\!].\, \mathbf{i}^-_\sigma(h) = \theta\}$$

where $\mathcal{D}[\![-]\!]$ denotes the relatively monotone frame*:*

$$\begin{aligned}
\mathcal{D}[\![\iota]\!] &:= \mathcal{M}[\![\iota]\!] \\
\mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!] &:= \mathcal{M}[\![\iota \to o]\!] \\
\mathcal{D}[\![\mathsf{Rel}^+(\sigma_1 \to \sigma_2)]\!] &:= \left[\mathcal{D}[\![\mathsf{Rel}^-(\sigma_1)]\!] \Rightarrow_{m[\mathsf{Im}_{\sigma_1}]} \mathcal{D}[\![\mathsf{Rel}^+(\sigma_2)]\!]\right]
\end{aligned}$$

The latter denotes the space of functions that are monotone with respect to Im_{σ_1}. That is, $f : \mathcal{D}[\![\mathsf{Rel}^-(\sigma_1)]\!] \to \mathcal{D}[\![\mathsf{Rel}^+(\sigma_2)]\!]$ is an element of $\mathcal{D}[\![\mathsf{Rel}^+(\sigma_1 \to \sigma_2)]\!]$ just if: $z_1 \sqsubseteq z_2$ implies $f\, z_1 \sqsubseteq f\, z_2$ for all $z_1, z_2 \in \mathsf{Im}_{\sigma_1}$.

Note that $\mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!] = \mathcal{D}[\![\iota \to o]\!] = \mathcal{M}[\![\iota \to o]\!]$. For higher-order relational sorts ρ, $\mathcal{D}[\![\rho]\!]$ captures a larger set of functions than $\mathcal{M}[\![\rho]\!]$ does.

Definition 4.20. *For all sorts σ over ι, we define two pairs of mappings:*

$$\mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!] \underset{\mathbf{j}_\sigma}{\overset{\mathbf{i}_\sigma}{\leftrightarrows}} \mathcal{H}[\![\sigma]\!] \qquad \mathcal{D}[\![\mathsf{Rel}^-(\sigma)]\!] \underset{\mathbf{j}^-_\sigma}{\overset{\mathbf{i}^-_\sigma}{\leftrightarrows}} \mathcal{H}[\![\sigma]\!]$$

For sort ι, all $t \in \mathcal{H}[\![\iota]\!]$ and $p \in \mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!]$, we define:

$$\mathbf{i}_\iota(t) := \lambda s.\, t \sqsubseteq s \qquad \mathbf{j}_\iota(p) := \begin{cases} \bot & \text{if } p = \lambda s.\, 0 \\ \textit{choice } \min\{t \mid p\, t\} & \text{O/W} \end{cases}$$

where choice *denotes an arbitrary choice function, which exists by the Axiom of Choice.*

For $\sigma = \sigma_1 \to \cdots \to \sigma_m \to \iota$ with $m > 0$, we define the following for all $h \in \mathcal{H}[\![\sigma]\!]$:

$$\mathbf{i}_\sigma\, h := \lambda x_1^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_1)]\!]} \ldots x_m^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_m)]\!]}.\, \mathbf{i}_\iota\left(h\,(\mathbf{j}^-_{\sigma_1}\, x_1) \ldots (\mathbf{j}^-_{\sigma_m}\, x_m)\right)$$

with

$$\mathbf{j}^-_\sigma : \mathcal{D}[\![\mathsf{Rel}^-(\sigma)]\!] \to \mathcal{H}[\![\sigma]\!] := \begin{cases} \textit{inclusion } \mathcal{D}[\![\iota]\!] \hookrightarrow \mathcal{H}[\![\iota]\!] & \text{if } \sigma = \iota \\ \mathbf{j}_\sigma & \textit{otherwise.} \end{cases}$$

Similarly, for all $\theta \in \mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]$:

$$\mathbf{j}_\sigma\, \theta := \textit{choice } \max\{h \in \mathcal{H}[\![\sigma]\!] \mid \theta \sqsubseteq \mathbf{i}_\sigma\, h\}$$

with

$$\mathbf{i}^-_\sigma : \mathcal{H}[\![\sigma]\!] \to \mathcal{D}[\![\mathsf{Rel}^-(\sigma)]\!] := \begin{cases} \textit{inclusion } \mathcal{H}[\![\iota]\!] \hookrightarrow \mathcal{D}[\![\iota]\!] & \text{if } \sigma = \iota \\ \mathbf{i}_\sigma & \textit{otherwise.} \end{cases}$$

The top case of $\mathbf{j}_\iota$ is not used in practice.

Lemma 4.21 (Well-sortedness)**.** *For all sorts σ over ι,*

(1) $\mathbf{i}_\sigma\left(\bot_{\mathcal{H}[\![\sigma]\!]}\right) = \top_{\mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]}$

(2) $\theta \in \mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]$ *implies* $\mathbf{j}_\sigma(\theta) \in \mathcal{H}[\![\sigma]\!]$

(3) $h \in \mathcal{H}[\![\sigma]\!]$ *implies that* $\mathbf{i}_\sigma(h) \in \mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]$

(4) $\mathbf{i}_\sigma$ *is injective*

(5) $\mathbf{i}_\sigma$ *is antitone*

(6) $\mathbf{j}_\sigma \circ \mathbf{i}_\sigma = \mathrm{id}_{\mathcal{H}[\![\sigma]\!]}$

(7) $\mathbf{j}^-_\sigma \circ \mathbf{i}^-_\sigma = \mathrm{id}_{\mathcal{H}[\![\sigma]\!]}$

Proof. We proceed by induction on sorts.

Case $\sigma = \iota$. For (1),

$$\mathbf{i}_\iota(\bot) = \lambda s.\,(\bot \sqsubseteq s) = \lambda s.\,1 = \top_{\mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!]}.$$

For (2), let $\theta \in \mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!]$. Observe that $\bot \in \mathcal{H}[\![\iota]\!]$ and $\{t \mid \theta\, t\} \subseteq \mathcal{H}[\![\iota]\!]$, so that the claim follows.

For (3), observe that $\mathbf{i}_\iota(h)$ takes an argument from $\mathcal{D}[\![\iota]\!] = \mathcal{M}[\![\iota]\!]$ and returns an element from $\mathcal{M}[\![o]\!]$, for all $h \in \mathcal{H}[\![\iota]\!]$. Trivially, $\mathbf{i}_\iota(h) \in \mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!] = \mathcal{M}[\![\iota \to o]\!]$.

For (5), let $h_1, h_2 \in \mathcal{H}[\![\iota]\!]$ such that $h_1 \sqsubseteq h_2$. Then:

$$\begin{aligned}\mathbf{i}_\iota(h_1) &= \lambda s.\,(h_1 \sqsubseteq s)\\ &\sqsupseteq \lambda s.\,(h_2 \sqsubseteq s)\\ &= \mathbf{i}_\iota(h_2)\end{aligned}$$

It follows that $\mathbf{i}_\iota$ is antitone.

For (6), let $t \in \mathcal{H}[\![\iota]\!]$, so that

$$\mathbf{j}_\iota(\mathbf{i}_\iota(t)) = \mathbf{j}_\iota(\lambda s.\,t \sqsubseteq s) = \mathit{choice}\ \min\{t' \mid (\lambda s.\,t \sqsubseteq s)\,t'\} = \mathit{choice}\,\{t\} = t.$$

Claim (7) is trivial because both $\mathbf{i}^-_\iota$ and $\mathbf{j}^-_\iota$ are identity mappings on the underlying set of $\mathcal{D}[\![\iota]\!]$ and $\mathcal{H}[\![\iota]\!]$.

Finally, claim (4) is a consequence of (6).

Case $\sigma = \sigma_1 \to \cdots \to \sigma_m \to \iota$ **for** $m > 0$**.** For (1),

$$\begin{aligned}
\mathbf{i}_\sigma\left(\bot_{\mathcal{H}[\![\sigma]\!]}\right) &= \lambda x_1^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_1)]\!]} \ldots x_m^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_m)]\!]}.\, \mathbf{i}_\iota\left(\bot_{\mathcal{H}[\![\sigma]\!]}\left(\mathbf{j}^-_{\sigma_1}\, x_1\right) \ldots \left(\mathbf{j}^-_{\sigma_m}\, x_m\right)\right) \\
&= \lambda x_1^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_1)]\!]} \ldots x_m^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_m)]\!]}.\, \mathbf{i}_\iota\left(\bot\right) \\
&= \lambda x_1^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_1)]\!]} \ldots x_m^{\mathcal{D}[\![\mathsf{Rel}^-(\sigma_m)]\!]}.\, \top_{\mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!]} \qquad \text{IH} \\
&= \top_{\mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]}.
\end{aligned}$$

For (2), let $\theta \in \mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]$. Then,

$$\mathbf{j}_\sigma(\theta) = \mathit{choice}\ \max\{h \in \mathcal{H}[\![\sigma]\!] \mid \theta \sqsubseteq \mathbf{i}_\sigma\, h\}.$$

Because $\mathbf{i}_\sigma(\bot_{\mathcal{H}[\![\sigma]\!]}) = \top_{\mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]}$ by (1), the set $\{h \in \mathcal{H}[\![\sigma]\!] \mid \theta \sqsubseteq \mathbf{i}_\sigma\, h\}$ is nonempty. Clearly, $\mathbf{j}_\sigma(\theta) \in \mathcal{H}[\![\sigma]\!]$.

For (3), let $h \in \mathcal{H}[\![\sigma]\!]$. Let $\mathbf{i}^-_{\sigma_i}(d_i), \mathbf{i}^-_{\sigma_i}(d_i') \in \mathsf{Im}_{\sigma_i}$ such that $\mathbf{i}^-_{\sigma_i}(d_i) \sqsubseteq \mathbf{i}^-_{\sigma_i}(d_i')$, for all $i \in [m]$. By the IH, in particular injectivity and antitonicity of $\mathbf{i}_{\sigma_i}$, it holds that $d_i \sqsupseteq d_i'$. Then,

$$\begin{aligned}
\mathbf{i}_\sigma(h)\left(\mathbf{i}^-_{\sigma_1}\, d_1\right) \ldots \left(\mathbf{i}^-_{\sigma_m}\, d_m\right) &= \mathbf{i}_\iota(h\left(\mathbf{j}^-_{\sigma_1}(\mathbf{i}^-_{\sigma_1}\, d_1)\right) \ldots \left(\mathbf{j}^-_{\sigma_m}(\mathbf{i}^-_{\sigma_m}\, d_m)\right)) \\
&= \mathbf{i}_\iota(h\, d_1 \ldots d_m) \qquad \text{IH} \\
&\sqsubseteq \mathbf{i}_\iota(h\, d_1' \ldots d_m') \qquad \text{IH} \\
&= \mathbf{i}_\sigma(h)\left(\mathbf{i}^-_{\sigma_1}\, d_1'\right) \ldots \left(\mathbf{i}^-_{\sigma_m}\, d_m'\right)
\end{aligned}$$

where we rely on monotonicity of $h \in \mathcal{H}[\![\sigma]\!]$ and antitonicity of $\mathbf{i}_\iota$. We conclude that $\mathbf{i}_\sigma(h) \in \mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]$.

For (4), let $h_1, h_2 \in \mathcal{H}[\![\sigma]\!]$ such that $h_1 \neq h_2$. Suppose for a contradiction that $\mathbf{i}_\sigma(h_1) = \mathbf{i}_\sigma(h_2)$. It follows from $h_1 \neq h_2$ that there exist $z_i \in \mathcal{H}[\![\sigma_i]\!]$, for $i \in [m]$, such that $h_1\, z_1 \ldots z_m \neq h_2\, z_1 \ldots z_m$. This means that $\mathbf{i}_\iota(h_1\, z_1 \ldots z_m) \neq \mathbf{i}_\iota(h_2\, z_1 \ldots z_m)$ by injectivity of $\mathbf{i}_\iota$. We rewrite both sides of the inequality using the IH on (6) to obtain

$$\mathbf{i}_\sigma(h_1)\left(\mathbf{i}^-_{\sigma_1}\, z_1\right) \ldots \left(\mathbf{i}^-_{\sigma_m}\, z_m\right) \neq \mathbf{i}_\sigma(h_2)\left(\mathbf{i}^-_{\sigma_1}\, z_1\right) \ldots \left(\mathbf{i}^-_{\sigma_m}\, z_m\right).$$

However, since $\mathbf{i}_\sigma(h_1) = \mathbf{i}_\sigma(h_2)$, this is a contradiction. We conclude that $\mathbf{i}_\sigma$ is injective.

For (5), let $h_1, h_2 \in \mathcal{H}[\![\sigma]\!]$ such that $h_1 \sqsubseteq h_2$. Let $z_i \in \mathcal{H}[\![\sigma_i]\!]$, for all $i \in [m]$. Then,

$$\begin{aligned}
\mathbf{i}_\sigma(h_1)\, z_1 \ldots z_m &= \mathbf{i}_\iota(h_1\left(\mathbf{j}^-_{\sigma_1}\, z_1\right) \ldots \left(\mathbf{j}^-_{\sigma_m}\, z_m\right)) \\
&\sqsupseteq \mathbf{i}_\iota(h_2\left(\mathbf{j}^-_{\sigma_1}\, z_1\right) \ldots \left(\mathbf{j}^-_{\sigma_m}\, z_m\right)) \qquad \text{IH} \\
&= \mathbf{i}_\sigma(h_2)\, z_1 \ldots z_m
\end{aligned}$$

It follows that $\mathbf{i}_\sigma$ is antitone.

For (6), let $h \in \mathcal{H}[\![\sigma]\!]$. Then,

$$\mathbf{j}_\sigma(\mathbf{i}_\sigma\, h) = \mathit{choice}\ \max\{h' \in \mathcal{H}[\![\sigma]\!] \mid \mathbf{i}_\sigma\, h \sqsubseteq \mathbf{i}_\sigma\, h'\} = \mathit{choice}\ \{h\} = h$$

using (4) (injectivity of $\mathbf{i}_\sigma$). Claim (7) is a consequence of (6). □

Lemma 4.22. *For all directed sets $D \subseteq \mathcal{H}[\![\iota]\!]$,*

$$\mathbf{i}_\iota \left(\bigsqcup D\right) = \bigsqcap \{\mathbf{i}_\iota(d) \mid d \in D\}.$$

Proof. Recall that $\mathcal{H}[\![\iota]\!]$ is a dcpo and $\mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!]$ a complete lattice, so that the bounds are defined.

$$\begin{aligned}
\mathbf{i}_\iota \left(\bigsqcup D\right) &= \lambda s.\, \left(\bigsqcup D \sqsubseteq s\right) \\
&= \lambda s.\, \bigsqcap \{d \sqsubseteq s \mid d \in D\} \qquad \star \\
&= \bigsqcap \{\lambda s.\, (d \sqsubseteq s) \mid d \in D\} \\
&= \bigsqcap \{\mathbf{i}_\iota(d) \mid d \in D\}
\end{aligned}$$

$\star$: Suppose that $\bigsqcup D \sqsubseteq s$ for $s \in \mathcal{D}[\![\iota]\!] = \mathcal{H}[\![\iota]\!]$. By transitivity, $d \sqsubseteq \bigsqcup D \sqsubseteq s$ for all $d \in D$. Since the greatest lower bound on $\mathcal{M}[\![o]\!]$ is conjunction, this implies $\bigsqcap\{d \sqsubseteq s \mid d \in D\}$. For the converse, suppose $\bigsqcap\{d \sqsubseteq s \mid d \in D\}$. This means that $d \sqsubseteq s$, for all $d \in D$. Thus, s is an upper bound on D. However, $\bigsqcup D$ is the least upper bound on this set, so $\bigsqcup D \sqsubseteq s$. □

4.4.2 Nonemptiness

We aim to show there exists a tree t that does not contain $\bot$, for every $n \geq 0$, such that $\mathcal{M}[\![\Delta_\mathcal{G} \vdash \ulcorner S \urcorner]\!](T^{\mathcal{M}\,n}_{P_\mathcal{G}:\Delta_\mathcal{G}}(\top_{\Delta_\mathcal{G}}))\, t = 1$ (Corollary 4.26). To this end, we define a family of logical relations in Definition 4.23 to capture this notion at higher sorts and for larger sort environments, as proved in Lemma 4.25.

Intuitively, such a relation holds whenever a predicate maps nonempty inputs to nonempty outputs, where 'nonempty' is with respect to $\bot$-free trees.

Definition 4.23. *We define a family of logical relations* $\bot\!-\!\mathsf{free}_\sigma \subseteq \mathcal{M}[\![\sigma]\!]$*:*

$$\begin{aligned}
\bot\!-\!\mathsf{free}_\iota(t) &:= t \text{ has no } \bot\text{-labelled leaves} \\
\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\iota)}(p) &:= \exists s \in \mathcal{M}[\![\iota]\!].\, \bot\!-\!\mathsf{free}_\iota(s) \wedge p\, s = 1 \\
\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\sigma_1\to\sigma_2)}(p) &:= \forall s \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_1)]\!].\, \bot\!-\!\mathsf{free}_{\mathsf{Rel}^-(\sigma_1)}(s) \Rightarrow \bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\sigma_2)}(p\, s) \\
\bot\!-\!\mathsf{free}_{\mathsf{Rel}^-(\Gamma)}(\theta) &:= \mathsf{dom}(\ulcorner\Gamma\urcorner) = \mathsf{dom}(\theta) \wedge \bigwedge_{x':\mathsf{Rel}^-(\sigma)\in\ulcorner\Gamma\urcorner} \bot\!-\!\mathsf{free}_{\mathsf{Rel}^-(\sigma)}(\theta(x'))
\end{aligned}$$

Alternatively, we can define:

$$\begin{aligned}
&\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\sigma_1\to\cdots\to\sigma_m\to\iota)}(p) \\
&\quad := \overline{\forall s \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma)]\!]}.\, \left(\bigwedge_{i\in[m]} \bot\!-\!\mathsf{free}_{\mathsf{Rel}^-(\sigma_i)}(s_i)\right) \Rightarrow \bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\iota)}(p\, s_1 \ldots s_m)
\end{aligned}$$

Lemma 4.24. *For all* $p_1, p_2 \in \mathcal{M}[\![\mathsf{Rel}^+(\iota)]\!]$*, if* $p_1 \sqsubseteq p_2$ *and* $\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\iota)}(p_1)$*, then* $\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\iota)}(p_2)$*.*

Proof. Straightforward, using that a witness $t \in \mathcal{M}[\![\iota]\!]$ of $\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\iota)}(p_1)$ is also a witness of $\bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\iota)}(p_2)$. □

Lemma 4.25. *For all* $n \geq 0$*, all typing judgements* $\mathcal{N}, \Gamma \vdash e : \sigma$ *of the HoRS* $\mathcal{G}$ *where* $\Gamma = \{x_1 : \tau_1, \ldots, x_k : \tau_k\}$*, and valuations* $\theta \in \mathcal{M}[\![\ulcorner\Gamma\urcorner]\!]$*,*

$$\mathcal{N}, \Gamma \vdash e : \sigma \wedge \bot\!-\!\mathsf{free}_{\mathsf{Rel}^-(\Gamma)}(\theta) \quad \Rightarrow \quad \bot\!-\!\mathsf{free}_{\mathsf{Rel}^+(\sigma)}\left(\mathcal{M}[\![\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner \vdash \ulcorner e\urcorner]\!](\beta^n)\right)$$

where

$$\begin{aligned}
\ulcorner\Gamma\urcorner &:= \{x'_1 : \mathsf{Rel}^-(\tau_1), \ldots, x'_k : \mathsf{Rel}^-(\tau_k)\} \\
\beta^n &:= \left(T^{\mathcal{M}\,n}_{P_\mathcal{G}:\Delta_\mathcal{G}}(\top_{\Delta_\mathcal{G}})\right)[\overline{x'} \mapsto \overline{\theta(x')}] \in \mathcal{M}[\![\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner]\!].
\end{aligned}$$

Notice that $\mathcal{N}, \Gamma \vdash e : \sigma$ *implies* $\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner \vdash \ulcorner e\urcorner : \mathsf{Rel}^+(\sigma)$*.*

Proof. We proceed by induction on $n \geq 0$ within which (both in the base case and the induction step) we use structural induction on HoRS term e. Some parts of the proof are presented out of order to avoid duplication. Figure 4.1 outlines the structure of the proof. We assume WLOG that e contains no λs.

We use the following shorthand for $e : \sigma = \sigma_1 \to \cdots \to \sigma_m \to \iota$ and $z_i \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_i)]\!]$ such that $\bot\!-\!\mathsf{free}_{\mathsf{Rel}^-(\sigma_i)}(z_i)$, for all $i \in [m]$:

$$A = \mathcal{M}[\![\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner \vdash \ulcorner e : \sigma\urcorner]\!](\beta^n)\, \overline{z}$$

Our proof strategy is to rewrite A and provide a witness to $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner e : \sigma\urcorner]\!](\beta^n)\,\overline{z})$, which proves $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner e : \sigma\urcorner]\!](\beta^n))$.

We present the following base cases w.r.t. the structure of e (also denoted b for *base case expression*). These cases hold for all $n \geq 0$.

Case $e = a : \iota \in \Sigma$. For all $n \geq 0$:

$$\begin{aligned} A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner a\urcorner]\!](\beta^n) \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \lambda r.\, a = r]\!](\beta^n) \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, \left(\widehat{F_a} = s\right) \end{aligned}$$

The $\bot$-free tree $\widehat{F_a}$ is a witness to $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner a\urcorner]\!](\beta^n))$.

Case $e = x : \iota \in V_{\mathrm{RS}}$. For all $n \geq 0$:

$$\begin{aligned} A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner x\urcorner]\!](\beta^n) \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \lambda r.\, x' = r]\!](\beta^n) \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, (\beta^n(x') = s) \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, (\theta(x') = s) \end{aligned}$$

The tree $\theta(x')$ is a witness to $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner x\urcorner]\!](\beta^n))$, because we know $\theta(x')$ to be $\bot$-free thanks to $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^-(\Gamma)}(\theta)$.

In the sequel, where $e : \sigma_1 \to \cdots \to \sigma_m \to \iota$, let $z_i \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_i)]\!]$ such that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^-(\sigma_i)}(z_i)$, for each $i \in [m]$.

Case $e = f : \sigma_1 \to \cdots \to \sigma_m \to \iota \in \Sigma$. For all $n \geq 0$:

$$\begin{aligned} A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner f\urcorner]\!](\beta^n)\,\overline{z} \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \lambda \overline{y}\, r.\, f\,\overline{y} = r]\!](\beta^n)\,\overline{z} \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, \left(\widehat{F_f}\,\overline{z} = s\right) \end{aligned}$$

The $\bot$-free tree $\widehat{F_f}\,\overline{z}$ is a witness to $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner f\urcorner]\!](\beta^n)\,\overline{z})$, which proves $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma_1 \to \cdots \to \sigma_m \to \iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner f\urcorner]\!](\beta^n))$.

Case $e = x : \sigma_1 \to \cdots \to \sigma_m \to \iota \in V_{\mathrm{RS}}$ *and* $m > 0$. For all $n \geq 0$:

$$\begin{aligned} A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner x\urcorner]\!](\beta^n)\,\overline{z} \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \lambda \overline{y}\, r.\, x'\,\overline{y}\, r]\!](\beta^n)\,\overline{z} \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, \beta^n(x')\,\overline{z}\, s \\ &= \theta(x')\,\overline{z} \end{aligned}$$

Since $\bot{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma_1\to\cdots\to\sigma_m\to\iota)}(\theta(x'))$ and $\bot{-}\mathsf{free}_{\mathsf{Rel}^-(\sigma_i)}(z_i)$, for each $i \in [m]$, it holds that $\bot{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\theta(x')\,\overline{z})$. This proves that $\bot{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner x\urcorner]\!](\beta^n))$.

This remaining base case expression is where we start needing induction on $n \geq 0$.

Case $e = F : \sigma_1 \to \cdots \to \sigma_m \to \iota \in \mathcal{N}$, *and* $n = 0$.

$$\begin{aligned} A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner F\urcorner]\!](\beta^0)\,\overline{z} \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \lambda\overline{y}\,r.\,R_F\,\overline{y}\,r]\!](\beta^0)\,\overline{z} \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\,\beta^0(R_F)\,\overline{z}\,s \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\,\top_{\mathcal{M}[\![\mathsf{Rel}^+(\sigma)]\!]}\,\overline{z}\,s \end{aligned}$$

Any tree $t \in \mathcal{M}[\![\iota]\!]$ such that $\bot{-}\mathsf{free}_{\iota}(t)$ is a witness to $\bot{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner F\urcorner]\!](\beta^0)\,\overline{z})$. Such a tree exists due to $[\![\mathcal{G}]\!]$ not containing bottom. This proves that $\bot{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner F\urcorner]\!](\beta^0))$.

This covers $n = 0$ for all base case expressions. We distinguish three induction hypotheses, where $S(n, e)$ denotes that the claim holds for n and expression e:

IH1 $S(0, e')$ for all expressions e' simpler than e

IH2 $S(n, e'')$ for n and all e''

IH3 $S(n+1, e')$ for all expressions e' simpler than e

The proof consists of four parts (in a logical sense but not a physical, to prevent duplication) which are related as in Figure 4.1.

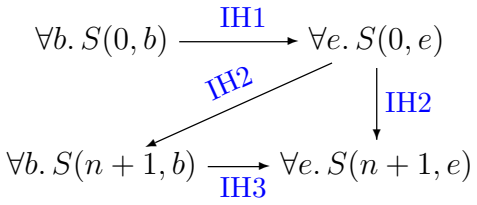


Figure 4.1: The inductive structure of the correctness proof of the HoRS-to-HoCHC encoding.

Thus, we have proved $S(0, b)$ for all base case expressions b. Next, we use IH1 to show that $S(0, e)$ for all expressions e.

In this inductive case we consider expressions $e = \$\,e_1 \ldots e_\ell : \sigma_1 \to \cdots \to \sigma_m \to \iota$ for some $\ell > 0$. As before, let $z_i \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_i)]\!]$ such that $\bot{-}\mathsf{free}_{\mathsf{Rel}^-(\sigma_i)}(z_i)$, for each

$i \in [m]$. We introduce some shorthands:

$$\begin{aligned}\Delta'_{\mathcal{G}} &:= \Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner, \overline{y}, r\\ \Delta''_{\mathcal{G}} &:= \Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner, \overline{y}, r, \overline{r}\\ \beta^{n,\overline{z},s} &:= \beta^n[\overline{y} \mapsto \overline{z}, r \mapsto s]\end{aligned}$$

Note that the sort τ of \$ is of the form

$$\tau = \tau_1 \to \cdots \to \tau_\ell \to \sigma_1 \to \cdots \to \sigma_m \to \iota$$

where $e_1 : \tau_1, \ldots, e_\ell : \tau_\ell$, for some $\ell > 0$. Sometimes we abbreviate $\sigma_1 \to \cdots \to \sigma_m \to \iota$ to σ.

In the sequel, steps marked with † use IH1 for $n = 0$, and IH3 for $n > 0$.

For all $n \geq 0$ and expressions $e = \$\,\overline{e}$, we can rewrite A to obtain:

$$\begin{aligned}
A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \ulcorner\$\,\overline{e}\urcorner]\!](\beta^n)\,\overline{z}\\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner\Gamma\urcorner \vdash \lambda\overline{y}\,r.\,\exists\overline{r}.\,\$'\,\urcorner(e_1, r_1)\urcorner \ldots \urcorner(e_\ell, r_\ell)\urcorner\,\overline{y}\,r \wedge \bigwedge_{i\in[\ell]} \mathit{Prop}(e_i, r_i)]\!](\beta^n)\,\overline{z}\\
&= \lambda s.\,\mathcal{M}[\![\Delta'_{\mathcal{G}} \vdash \exists\overline{r}.\,\$'\,\urcorner(e_1, r_1)\urcorner \ldots \urcorner(e_\ell, r_\ell)\urcorner\,\overline{y}\,r \wedge \bigwedge_{i\in[\ell]} \mathit{Prop}(e_i, r_i)]\!](\beta^{n,\overline{z},s})\\
&= \lambda s.\,\max\Big\{\\
&\qquad \min\{\\
&\qquad\quad \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \$'\,\urcorner(e_1, r_1)\urcorner \ldots \urcorner(e_\ell, r_\ell)\urcorner\,\overline{y}\,r]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]),\\
&\qquad\quad \min\{\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \mathit{Prop}(e_i, r_i)]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) \mid i \in [\ell]\}\\
&\qquad \}\\
&\quad \mid \forall i \in [\ell].\, r'_i \in \mathcal{M}[\![\mathsf{Rel}^-(\tau_i)]\!]\Big\}\\
&= \lambda s.\,\max\Big\{\\
&\qquad \min\{\\
&\qquad\quad \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \$']\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}])(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \urcorner(e_1, r_1)\urcorner]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]))\\
&\qquad\qquad\qquad \ldots(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \urcorner(e_\ell, r_\ell)\urcorner]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]))\,\overline{z}\,s,\\
&\qquad\quad \min\{\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \mathit{Prop}(e_i, r_i)]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) \mid i \in [\ell]\}\\
&\qquad \}\\
&\quad \mid \forall i \in [\ell].\, r'_i \in \mathcal{M}[\![\mathsf{Rel}^-(\tau_i)]\!]\Big\}
\end{aligned}$$

We now distinguish two cases for each subexpression $e_i : \tau_i$, namely $\tau_i = \iota$ and $\tau_i = \tau'_1 \to \tau'_2$.

If e_i is of sort ι, then the following holds:

$$\begin{aligned}
\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \mathit{Prop}(e_i : \iota, r_i)]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \ulcorner e_i \urcorner\, r_i]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) \\
&= \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \ulcorner e_i \urcorner]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}])\, r'_i \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'_i \\
\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \urcorner (e_i : \iota, r_i) \ulcorner]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash r_i]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) \\
&= r'_i
\end{aligned}$$

We know from † that $\bot\text{–}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n))$. This means that there exists $r'' \in \mathcal{M}[\![\iota]\!]$ such that $\bot\text{–}\mathsf{free}_{\iota}(r'')$ and $\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'' = 1$.

Otherwise, in case $e_i : \tau_i = \tau'_1 \to \tau'_2$, the following holds:

$$\begin{aligned}
\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \mathit{Prop}(e_i : \tau_i, r_i)]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \mathsf{true}]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) \\
&= 1 \\
\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \urcorner (e_i : \tau_i, r_i) \ulcorner]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \ulcorner e_i \urcorner]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}]) \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)
\end{aligned}$$

We know from † that $\bot\text{–}\mathsf{free}_{\mathsf{Rel}^+(\tau_i)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n))$.

As 'semantic equivalents' of the above terms, let us write

$$P_i : o := \begin{cases} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'_i & \text{if } \tau_i = \iota \\ 1 & \text{if } \tau_i = \tau'_1 \to \tau'_2 \end{cases}$$

and

$$T_i : \mathsf{Rel}^-(\tau_i) := \begin{cases} r'_i & \text{if } \tau_i = \iota \\ \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n) & \text{if } \tau_i = \tau'_1 \to \tau'_2 \end{cases}$$

for all $i \in [\ell]$. Additionally, we define:

$$S_i : \mathsf{Rel}^-(\tau_i) := \begin{cases} r''_i & \text{if } \tau_i = \iota \\ \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n) & \text{if } \tau_i = \tau'_1 \to \tau'_2 \end{cases}$$

where r''_i is an arbitrary ($\bot$-free) witness to $\bot\text{–}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n))$, which exists by †. This gives us $\bot\text{–}\mathsf{free}_{\mathsf{Rel}^-(\tau_i)}(S_i)$ for all $i \in [m]$.

We derive by abuse of notation, using the above:

$$A = \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \$']\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}])\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right)$$

We continue by case analysis on \$.

Case $e = f\, e_1 \ldots e_\ell$ *with* $f \in \Sigma$. For all $n \geq 0$:

$$
\begin{aligned}
A &= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash D_f]\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}])\, \overline{r'}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\widehat{F_f}\, \overline{r'}\ \overline{z} = s \wedge \bigwedge_{i \in [\ell]} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'_i \right) \\
&\sqsupseteq \lambda s.\, \left(\widehat{F_f}\, \overline{r''}\ \overline{z} = s \right) \qquad \dagger
\end{aligned}
$$

The $\bot$-free tree $\widehat{F_f}\, \overline{r''}\, \overline{z}$ witnesses $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)} \left(\lambda s.\, \left(\widehat{F_f}\, \overline{r''}\ \overline{z} = s \right) \right)$ and, therefore, $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner f\, \overline{e} \urcorner]\!](\beta^n)\, \overline{z})$. It follows that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner f\, \overline{e} \urcorner]\!](\beta^n))$.

Case $e = x\, e_1 \ldots e_\ell$ *with* $x \in V_{\mathrm{RS}}$. For all $n \geq 0$:

$$
\begin{aligned}
A &= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash x']\!](\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}])\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\beta^{n,\overline{z},s}[\overline{r} \mapsto \overline{r'}](x')\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\theta(x')\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&\sqsupseteq \lambda s.\, \theta(x')\, \overline{S}\ \overline{z}\ s \qquad \dagger \\
&= \theta(x')\, \overline{S}\ \overline{z}
\end{aligned}
$$

Recall that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\tau)}(\theta(x'))$. All arguments $\overline{S}$ and $\overline{z}$ are also $\bot$-free, so it follows that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\theta(x')\, \overline{S}\, \overline{z})$, which proves $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner x\, \overline{e} \urcorner]\!](\beta^n))$.

Case $e = F\, e_1 \ldots e_\ell$ *with* $F : \sigma \in \mathcal{N}$, *and* $n = 0$.

$$
\begin{aligned}
A &= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash R_F]\!](\beta^{0,\overline{z},s}[\overline{r} \mapsto \overline{r'}])\, \overline{T}\, \overline{z}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\beta^{0,\overline{z},s}[\overline{r} \mapsto \overline{r'}](R_F)\, \overline{T}\, \overline{z}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\top_{\mathcal{M}[\![\mathsf{Rel}^+(\tau)]\!]}\, \overline{T}\, \overline{z}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(1 \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, 1 \qquad \text{IH1}
\end{aligned}
$$

Any tree $\bot$-free $t \in \mathcal{M}[\![\iota]\!]$ is a witness to $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F\, \overline{e} \urcorner]\!](\beta^0)\, \overline{z})$. Such a tree exists due to $[\![\mathcal{G}]\!]$ not containing bottom. Finally, this proves that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F\, \overline{e} \urcorner]\!](\beta^0))$.

We have now established that $S(0, e'')$ holds for expressions all e''. The following case is the last remaining case to prove $S(n+1, b)$ for all base case expressions b:

Case $e = F : \sigma \in \mathcal{N}$, *for* $n + 1$.

$$
\begin{aligned}
A &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F \urcorner]\!](\beta^{n+1})\, \overline{z} \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, R_F\, \overline{y}\, r]\!](\beta^{n+1})\, \overline{z} \\
&= \lambda s.\, \beta^{n+1,\overline{z},s}(R_F)\, \overline{z}\, s \\
&= \lambda s.\, \beta^{n+1}(R_F)\, \overline{z}\, s \\
&= \lambda s.\, \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{z}\, s \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{z}
\end{aligned}
$$

IH2 gives us $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n))$, from which it follows that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{z})$ and, thus, $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F \urcorner]\!](\beta^{n+1}))$.

Finally, we present the remaining case to prove the claim that $S(n', e'')$ for all $n' \geq 0$ and all expressions e''.

Case $e = F\, e_1 \ldots e_\ell$ *with* $F \in \mathcal{N}$ *and* $\ell > 0$, *for* $n+1$.

$$
\begin{aligned}
A &= \lambda s.\, \exists \overline{r'}.\ \left(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash R_F]\!](\beta^{n+1,\overline{z},s}[\overline{r} \mapsto \overline{r'}])\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\ \left(\beta^{n+1,\overline{z},s}[\overline{r} \mapsto \overline{r'}](R_F)\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\ \left(\beta^{n+1}(R_F)\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\ \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{T}\ \overline{z}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&\sqsupseteq \lambda s.\, \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{S}\ \overline{z}\ s && \text{IH3} \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{S}\ \overline{z}
\end{aligned}
$$

By IH2, $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\tau)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n))$. All arguments $\overline{S}$ and $\overline{z}$ are also $\bot$-free, so it follows that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^n)\, \overline{S}\ \overline{z})$, which proves that $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\sigma)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F\, \overline{e} \urcorner]\!](\beta^{n+1}))$. □

Corollary 4.26. *For all* $n \geq 0$, $\bot\text{-}\mathsf{free}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\beta^n))$. *I.e. there exists a* $\bot$*-free tree* $t \in \mathcal{M}[\![\iota]\!]$ *such that* $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\beta^n)\, t$.

Lemma 4.27. *There exists a* $\bot$*-free tree* $t \in \mathcal{M}[\![\iota]\!]$ *such that* $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!]\, (\prod \beta^n)\ t$.

Proof. Note that the constructed HoCHC logic program $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$ is 'incremental', in the sense that each iteration of the one-step consequence operator (further) constrains a finite prefix of the trees it generates. This means that either a contradiction occurs in finite time (e.g. $\exists r_1.\, a\, r_1 = r \wedge b\, r_1 = r$ where a, b are distinct unary alphabet symbols) or no contradiction occurs and the program is strictly incremental.

By Corollary 4.26, no contradiction occurs after finite time. This means that no contradiction occurs at all and $\vdash P_{\mathcal{G}} : \Delta_{\mathcal{G}}$ is strictly incremental. It follows that there exists a $\bot$-free tree $t \in \mathcal{M}[\![\iota]\!]$ such that $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!]\, (\prod \beta^n)\ t$. □

4.4.3 Inclusion

We aim to show that, for every $n \geq 0$, $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](T^{\mathcal{M}\, n}_{P_{\mathcal{G}} : \Delta_{\mathcal{G}}}(\top_{\Delta_{\mathcal{G}}}))$ is included in $\mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N} \vdash S]\!]((\mathcal{H}[\![\mathcal{G}]\!]^n_{\mathcal{N}}(\bot_{\mathcal{N}}))))$. To this end, we define a family of logical relations in

Definition 4.28 to capture this notion at higher sorts and for larger sort environments, as proved in Lemma 4.30.

The intuition is that the relation comprises pairs that preserve order on order-preserving arguments.

Definition 4.28. *We define a family of logical relations* $\mathsf{Incl}_\sigma \subseteq \mathcal{M}[\![\sigma]\!] \times \mathcal{D}[\![\sigma]\!]$*:*

$$\begin{aligned}
\mathsf{Incl}_\iota(t_1, t_2) &:= t_1 = t_2 \\
\mathsf{Incl}_{\mathsf{Rel}^+(\iota)}(p_1, p_2) &:= p_1 \sqsubseteq p_2 \\
\mathsf{Incl}_{\mathsf{Rel}^+(\sigma_1 \to \sigma_2)}(p_1, p_2) &:= \forall w \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_1)]\!].\, \forall z \in \mathcal{H}[\![\sigma_1]\!]. \\
&\qquad \mathsf{Incl}_{\mathsf{Rel}^-(\sigma_1)}(w, \mathbf{i}^-_{\sigma_1}(z)) \Rightarrow \mathsf{Incl}_{\mathsf{Rel}^+(\sigma_2)}(p_1\, w, p_2\, \mathbf{i}^-_{\sigma_1}(z)) \\
\mathsf{Incl}_\Gamma(\theta_\beta, \theta_\alpha) &:= \mathsf{dom}(\ulcorner \Gamma \urcorner) = \mathsf{dom}(\theta_\beta) \wedge \mathsf{dom}(\Gamma) = \mathsf{dom}(\theta_\alpha) \wedge \\
&\qquad \bigwedge_{x:\sigma \in \Gamma} \mathsf{Incl}_{\mathsf{Rel}^-(\sigma)}(\theta_\beta(x'), \mathbf{i}^-_\sigma(\theta_\alpha(x)))
\end{aligned}$$

Alternatively, $\mathsf{Incl}_{\mathsf{Rel}^+(\sigma_1 \to \cdots \to \sigma_m \to \iota)}(p_1, p_2)$ *can be defined as*

$$\forall \overline{w}.\, \forall \overline{z}.\, \left(\bigwedge_{i \in [m]} \mathsf{Incl}_{\mathsf{Rel}^-(\sigma_i)}(w_i, \mathbf{i}^-_{\sigma_i}(z_i)) \right) \Rightarrow \mathsf{Incl}_{\mathsf{Rel}^+(\iota)}(p\, w_1 \ldots w_m, p_2\, \mathbf{i}^-_{\sigma_1}(z_1) \ldots \mathbf{i}^-_{\sigma_m}(z_m))$$

for all $\overline{w}$ *and* $\overline{z}$ *from the appropriate domains.*

Note that in general $\mathcal{M}[\![\sigma]\!]$ differs from $\mathcal{D}[\![\sigma]\!]$ and the arguments of Incl_σ do not necessarily live in the same set. However, for the sort we are interested in, namely $\mathsf{Rel}^+(\iota) = \iota \to o$, the denotations $\mathcal{M}[\![\iota \to o]\!]$ and $\mathcal{D}[\![\iota \to o]\!]$ coincide (idem for ι), so that the relations are well-defined.

Lemma 4.29. *For* $F \in \mathcal{H}[\![\sigma_1 \to \cdots \to \sigma_k \to \tau_1 \to \cdots \to \tau_\ell \to \iota]\!]$*,* $t_i \in \mathcal{H}[\![\sigma_i]\!]$ *for all* $i \in [k]$*, and* $z_j \in \mathcal{H}[\![\tau_j]\!]$ *for all* $j \in [\ell]$*,*

$$\begin{aligned}
\left(\lambda s.\, \exists \overline{r}.\, F\, \overline{r}\, \overline{z} = s \wedge \bigwedge_{i \in [k]} t_i \sqsubseteq r_i \right) &\sqsubseteq \left(\lambda s.\, \exists \overline{r}.\, F\, \overline{r}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i \in [k]} t_i \sqsubseteq r_i \right) \\
&\sqsubseteq \left(\lambda s.\, F\, \overline{t}\, \overline{z} \sqsubseteq s \right)
\end{aligned}$$

where $\overline{r} = r_1 \ldots r_k$*,* $\overline{t} = t_1 \ldots t_k$*, and* $\overline{z} = z_1 \ldots z_\ell$*.*

Proof.

$$\left(\lambda s.\, \exists \overline{r}.\, F\, \overline{r}\, \overline{z} = s \wedge \bigwedge_{i\in[k]} t_i \sqsubseteq r_i\right) \sqsubseteq \left(\lambda s.\, \exists \overline{r}.\, F\, \overline{r}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i\in[k]} t_i \sqsubseteq r_i\right)$$
$$\sqsubseteq \left(\lambda s.\, \exists \overline{r}.\, F\, \overline{t}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i\in[k]} t_i \sqsubseteq r_i\right)$$
$$\sqsubseteq \left(\lambda s.\, F\, \overline{t}\, \overline{z} \sqsubseteq s\right)$$

□

Lemma 4.30. *For all $n \geq 0$, all typing judgements $\mathcal{N}, \Gamma \vdash e : \sigma$ of the HoRS $\mathcal{G}$ where $\Gamma = \{x_1 : \tau_1, \ldots, x_k : \tau_k\}$, and valuations $\theta_\alpha \in \mathcal{H}[\![\Gamma]\!]$ and $\theta_\beta \in \mathcal{M}[\![\ulcorner\Gamma\urcorner]\!]$,*

$$\mathcal{N}, \Gamma \vdash e : \sigma \wedge \mathsf{Incl}_\Gamma(\theta_\beta, \theta_\alpha) \quad \Rightarrow$$
$$\mathsf{Incl}_{\mathsf{Rel}^+(\sigma)}\left(\mathcal{M}[\![\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner \vdash \ulcorner e\urcorner]\!](\beta^n), \mathbf{i}_\sigma(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e]\!](\alpha^n))\right)$$

where

$$\ulcorner\Gamma\urcorner := \{x_1' : \mathsf{Rel}^-(\tau_1), \ldots, x_k' : \mathsf{Rel}^-(\tau_k)\}$$
$$\alpha^n := (\mathcal{H}[\![\mathcal{G}]\!]^n_\mathcal{N}(\bot_\mathcal{N}))\,[\overline{x} \mapsto \theta_\alpha(x)] \in \mathcal{H}[\![\mathcal{N}, \Gamma]\!]$$
$$\beta^n := \left(T^{\mathcal{M}\,n}_{P_\mathcal{G}:\Delta_\mathcal{G}}(\top_{\Delta_\mathcal{G}})\right)\,[\overline{x'} \mapsto \overline{\theta_\beta(x')}] \in \mathcal{M}[\![\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner]\!].$$

Notice that $\mathcal{N}, \Gamma \vdash e : \sigma$ implies $\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner \vdash \ulcorner e\urcorner : \mathsf{Rel}^+(\sigma)$.

Proof. We proceed by induction on $n \geq 0$ within which (both in the base case and the induction step) we use structural induction on HoRS term e. Some parts of the proof are presented out of order to avoid duplication. In fact, the structure of this proof and the order of presentation correspond to the proof of Lemma 4.25, the structure of which is outlined in Figure 4.1. We again assume WLOG that e contains no λs.

We use the following shorthands for $e : \sigma = \sigma_1 \to \cdots \to \sigma_m \to \iota$, $w_i \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_i)]\!]$, and $z_i \in \mathcal{H}[\![\sigma_i]\!]$ such that $\mathsf{Incl}_{\mathsf{Rel}^-(\sigma_i)}(w_i, \mathbf{i}^-_{\sigma_i}(z_i))$, for all $i \in [m]$:

$$B = \mathcal{M}[\![\Delta_\mathcal{G}, \ulcorner\Gamma\urcorner \vdash \ulcorner e\urcorner]\!](\beta^n)\,\overline{w}$$
$$C = \mathbf{i}_\sigma(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e]\!](\alpha^n))\,\overline{\mathbf{i}^-(z)}$$

Thus, both B and C are both elements of $\mathcal{M}[\![\iota \to o]\!] = \mathcal{D}[\![\iota \to o]\!]$, and it suffices to show that $B \sqsubseteq C$.

We present the following base cases w.r.t. the structure of e (also denoted b for *base case expression*). These cases hold for all $n \geq 0$.

Case $e = a : \iota \in \Sigma$. For all $n \geq 0$:

$$\begin{aligned} B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner a \urcorner]\!](\beta^n) \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda r.\, a = r]\!](\beta^n) \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, \left(\widehat{F_a} = s\right) \\ &= \mathbf{i}_\iota(\widehat{F_a}) \\ &= \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash a]\!](\alpha^n)) \\ &= C \end{aligned}$$

Case $e = x : \iota \in V_{\mathrm{RS}}$. For all $n \geq 0$:

$$\begin{aligned} B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner x \urcorner]\!](\beta^n) \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda r.\, x' = r]\!](\beta^n) \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, (\beta^n(x') = s) \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, (\alpha^n(x) = s) \\ &\sqsubseteq \mathbf{i}_\iota(\alpha^n(x)) \\ &= \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash x]\!](\alpha^n)) \\ &= C \end{aligned}$$

Note $\mathsf{Incl}_\Gamma(\theta_\beta, \theta_\alpha)$ implies $\mathsf{Incl}_\iota(\beta^n(x'), \mathbf{i}_\iota^-(\alpha^n(x)))$ and $\beta^n(x') = \alpha^n(x)$, as used above.

In the sequel, where $e : \sigma_1 \to \cdots \to \sigma_m \to \iota$, let $w_j \in \mathcal{M}[\![\mathsf{Rel}^-(\sigma_j)]\!]$ and $z_j \in \mathcal{H}[\![\sigma_j]\!]$ such that $\mathsf{Incl}_{\mathsf{Rel}^-(\sigma_j)}(w_j, \mathbf{i}_{\sigma_j}^-(z_j))$, for each $j \in [m]$. Similar to what we just used, $\sigma_j = \iota$ implies $w_j = z_j$.

Case $e = f : \sigma_1 \to \cdots \to \sigma_m \to \iota \in \Sigma$. For all $n \geq 0$:

$$\begin{aligned} B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner f \urcorner]\!](\beta^n)\, \overline{w} \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, f\, \overline{y} = r]\!](\beta^n)\, \overline{z} \\ &= \lambda s \in \mathcal{M}[\![\iota]\!].\, \left(\widehat{F_f}\, \overline{z} = s\right) \\ &\sqsubseteq \mathbf{i}_\iota(\widehat{F_f}\, \overline{z}) \\ &= \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash f]\!](\alpha^n)\, \overline{z}) \\ &= \mathbf{i}_{\sigma_1 \to \cdots \to \sigma_m \to \iota}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash f]\!](\alpha^n))\, \overline{\mathbf{i}^-(z)} \qquad \text{Lem 4.21} \\ &= C \end{aligned}$$

Case $e = x : \sigma_1 \to \cdots \to \sigma_m \to \iota \in V_{\mathrm{RS}}$. For all $n \geq 0$:

$$\begin{aligned}
B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner x \urcorner]\!](\beta^n)\,\overline{w} \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, x'\, \overline{y}\, r]\!](\beta^n)\,\overline{w} \\
&= \lambda s \in \mathcal{M}[\![\iota]\!].\, \beta^n(x')\,\overline{w}\, s \\
&= \beta^n(x')\,\overline{w} \\
&\sqsubseteq \mathbf{i}^-_{\sigma_1 \to \cdots \to \sigma_m \to \iota}(\alpha^n(x))\,\overline{\mathbf{i}^-(z)} \\
&= \mathbf{i}_{\sigma_1 \to \cdots \to \sigma_m \to \iota}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash x]\!](\alpha^n))\,\overline{\mathbf{i}^-(z)} \\
&= C
\end{aligned}$$

Since $\mathsf{Incl}_{\mathsf{Rel}^-(\sigma_1 \to \cdots \to \sigma_m \to \iota)}(\beta^n(x'), \mathbf{i}^-_{\sigma_1 \to \cdots \to \sigma_m \to \iota}(\alpha^n(x)))$ and $\mathsf{Incl}_{\mathsf{Rel}^-(\sigma_j)}(w_j, \mathbf{i}^-_{\sigma_j}(z_j))$, for all $j \in [m]$, the inclusion above holds.

This remaining base case expression is where we start needing induction on $n \geq 0$.

Case $e = F : \sigma \in \mathcal{N}$, *and* $n = 0$.

$$\begin{aligned}
B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F \urcorner]\!](\beta^0)\,\overline{w} \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, R_F\, \overline{y}\, r]\!](\beta^0)\,\overline{w} \\
&= \lambda s \in \mathcal{M}[\![\iota]\!].\, \beta^0(R_F)\,\overline{w}\, s \\
&= \lambda s \in \mathcal{M}[\![\iota]\!].\, \top_{\mathcal{M}[\![\mathsf{Rel}^+(\sigma)]\!]}\,\overline{w}\, s \\
&= \top_{\mathcal{M}[\![\mathsf{Rel}^+(\iota)]\!]} \\
&= \top_{\mathcal{D}[\![\mathsf{Rel}^+(\iota)]\!]} \\
&= \top_{\mathcal{D}[\![\mathsf{Rel}^+(\sigma)]\!]}\,\overline{\mathbf{i}^-(z)} \\
&= \mathbf{i}_\sigma(\bot_{\mathcal{H}[\![\sigma]\!]})\,\overline{\mathbf{i}^-(z)} \qquad \text{Lem 4.21} \\
&= \mathbf{i}_\sigma(\alpha^0(F))\,\overline{\mathbf{i}^-(z)} \\
&= \mathbf{i}_\sigma(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F]\!](\alpha^0))\,\overline{\mathbf{i}^-(z)} \\
&= C
\end{aligned}$$

This covers $n = 0$ for all base case expressions. We distinguish three induction hypotheses, where $S(n, e)$ denotes that the claim holds for n and expression e:

IH1 $S(0, e')$ for all expressions e' simpler than e

IH2 $S(n, e'')$ for n and all e''

IH3 $S(n+1, e')$ for all expressions e' simpler than e

The proof consists of four parts (in a logical sense but not a physical, since parts of the proof would need to be duplicated) which are related as in Figure 4.1.

In the above, we have showed that $S(0, b)$ for all base case expressions b.

We rewrite B for the inductive case where $e = \$\, e_1 \ldots e_\ell$, using the shorthands:

$$\begin{aligned} \Delta'_{\mathcal{G}} &:= \Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner, \overline{y}, r \\ \Delta''_{\mathcal{G}} &:= \Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner, \overline{y}, r, \overline{r} \\ \beta^{n,\overline{w},s} &:= \beta^n[\overline{y} \mapsto \overline{w}, r \mapsto s] \end{aligned}$$

Recall that $\sigma_1 \to \cdots \to \sigma_m \to \iota$ is the sort of $e = \$\, e_1 \ldots e_\ell$, for some $m \geq n \geq 0$. This means that the sort τ of $\$$ is of the form

$$\tau = \tau_1 \to \cdots \to \tau_\ell \to \sigma_1 \to \cdots \to \sigma_m \to \iota$$

where $e_1 : \tau_1, \ldots, e_\ell : \tau_\ell$, for some $\ell > 0$. We may shorten $\sigma_1 \to \cdots \to \sigma_m \to \iota$ to σ.

For all $n \geq 0$ and expressions $e = \$\, \overline{e}$, we can rewrite B to obtain:

$$\begin{aligned} B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \$\, \overline{e} \urcorner]\!](\beta^n)\, \overline{w} \\ &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, \exists \overline{r}.\, \$' \urcorner (e_1, r_1) \urcorner \ldots \urcorner (e_\ell, r_\ell) \urcorner \overline{y}\, r \wedge \bigwedge_{i \in [\ell]} Prop(e_i, r_i)]\!](\beta^n)\, \overline{w} \\ &= \lambda s.\, \mathcal{M}[\![\Delta'_{\mathcal{G}} \vdash \exists \overline{r}.\, \$' \urcorner (e_1, r_1) \urcorner \ldots \urcorner (e_\ell, r_\ell) \urcorner \overline{y}\, r \wedge \bigwedge_{i \in [\ell]} Prop(e_i, r_i)]\!](\beta^{n,\overline{w},s}) \\ &= \lambda s.\, \max \Big\{ \\ &\qquad \min\{ \\ &\qquad\quad \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \$' \urcorner (e_1, r_1) \urcorner \ldots \urcorner (e_\ell, r_\ell) \urcorner \overline{y}\, r]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]), \\ &\qquad\quad \min\{\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash Prop(e_i, r_i)]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) \mid i \in [\ell]\} \\ &\qquad \} \\ &\quad \mid \forall i \in [\ell].\, r'_i \in \mathcal{M}[\![\mathsf{Rel}^-(\tau_i)]\!] \Big\} \\ &= \lambda s.\, \max \Big\{ \\ &\qquad \min\{ \\ &\qquad\quad \mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \$']\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}])(\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \urcorner (e_1, r_1) \urcorner]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}])) \\ &\qquad\qquad\qquad \ldots (\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash \urcorner (e_\ell, r_\ell) \urcorner]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]))\, \overline{w}\, s, \\ &\qquad\quad \min\{\mathcal{M}[\![\Delta''_{\mathcal{G}} \vdash Prop(e_i, r_i)]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) \mid i \in [\ell]\} \\ &\qquad \} \\ &\quad \mid \forall i \in [\ell].\, r'_i \in \mathcal{M}[\![\mathsf{Rel}^-(\tau_i)]\!] \Big\} \end{aligned}$$

In the sequel, steps marked with † use IH1 for $n = 0$, and IH3 for $n > 0$.

We now distinguish two cases for each subexpression $e_i : \tau_i$, namely $\tau_i = \iota$ and $\tau_i = \tau_1' \to \tau_2'$.

If e_i is of sort ι, then the following holds:

$$\begin{aligned}
\mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash Prop(e_i : \iota, r_i)]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash \ulcorner e_i \urcorner\, r_i]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash \ulcorner e_i \urcorner]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}])\, r_i' \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r_i' \\
\mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash \urcorner(e_i : \iota, r_i)\ulcorner]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash r_i]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) \\
&= r_i'
\end{aligned}$$

We know from † that $\mathsf{Incl}_{\mathsf{Rel}^+(\iota)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n), \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n))$.

Otherwise, in case $e_i : \tau_i = \tau_1' \to \tau_2'$, the following holds:

$$\begin{aligned}
\mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash Prop(e_i : \tau_i, r_i)]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash \mathsf{true}]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) \\
&= 1 \\
\mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash \urcorner(e_i : \tau_i, r_i)\ulcorner]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) &= \mathcal{M}[\![\Delta_{\mathcal{G}}'' \vdash \ulcorner e_i \urcorner]\!](\beta^{n,\overline{w},s}[\overline{r} \mapsto \overline{r'}]) \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)
\end{aligned}$$

We know from † that $\mathsf{Incl}_{\mathsf{Rel}^+(\tau_i)}(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n), \mathbf{i}_{\tau_i}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n))$.

As 'semantic equivalents' of the above terms, let us write

$$P_i : o := \begin{cases} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r_i' & \text{if } \tau_i = \iota \\ 1 & \text{if } \tau_i = \tau_1' \to \tau_2' \end{cases}$$

and

$$T_i : \mathsf{Rel}^-(\tau_i) := \begin{cases} r_i' & \text{if } \tau_i = \iota \\ \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n) & \text{if } \tau_i = \tau_1' \to \tau_2' \end{cases}$$

for all $i \in [\ell]$. Additionally, we define

$$P_i' : o := \begin{cases} \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n))\, r_i' & \text{if } \tau_i = \iota \\ 1 & \text{if } \tau_i = \tau_1' \to \tau_2' \end{cases}$$

and

$$T_i' : \mathsf{Rel}^-(\tau_i) := \begin{cases} r_i' & \text{if } \tau_i = \iota \\ \mathbf{i}_{\tau_i}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n)) & \text{if } \tau_i = \tau_1' \to \tau_2' \end{cases}$$

for all $i \in [\ell]$, to be used after applying the induction hypothesis †. And finally, for all $i \in [\ell]$,

$$S_i : \mathsf{Rel}^-(\tau_i) := \begin{cases} r_i' & \text{if } \tau_i = \iota \\ \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n) & \text{if } \tau_i = \tau_1' \to \tau_2' \end{cases}$$

We derive by abuse of notation, using the above:

$$B = \lambda s.\, \exists \overline{r'}.\ \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \$']\!](\beta^n)\, \overline{T}\ \overline{w}\ s \wedge \bigwedge_{i \in [\ell]} P_i \right)$$

We continue by case analysis on \$.

Case $e = f\, e_1 \dots e_\ell$ *with* $f \in \Sigma$. For all $n \geq 0$:

$$\begin{aligned}
B &= \lambda s.\, \exists \overline{r'}.\ \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash D_f]\!](\beta^n)\, \overline{r'}\ \overline{w}\ s \wedge \bigwedge_{i \in [\ell]} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\ \left(\widehat{F_f}\, \overline{r'}\ \overline{w} = s \wedge \bigwedge_{i \in [\ell]} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e_i \urcorner]\!](\beta^n)\, r'_i \right) \\
&\sqsubseteq \lambda s.\, \exists \overline{r'}.\ \left(\widehat{F_f}\, \overline{r'}\ \overline{w} = s \wedge \bigwedge_{i \in [\ell]} \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n))\, r'_i \right) \qquad \dagger \\
&= \lambda s.\, \exists \overline{r'}.\ \left(\widehat{F_f}\, \overline{r'}\ \overline{w} = s \wedge \bigwedge_{i \in [\ell]} (\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n) \sqsubseteq r'_i) \right) \\
&= \lambda s.\, \exists \overline{r'}.\ \left(\widehat{F_f}\, \overline{r'}\ \overline{z} = s \wedge \bigwedge_{i \in [\ell]} (\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_i]\!](\alpha^n) \sqsubseteq r'_i) \right) \\
&\sqsubseteq \lambda s.\, \left(\widehat{F_f}\, \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^n) \dots \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^n)\, \overline{z} \sqsubseteq s \right) \qquad \text{Lem 4.29} \\
&= \mathbf{i}_\iota(\widehat{F_f}\, \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^n) \dots \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^n)\, \overline{z}) \\
&= \mathbf{i}_\sigma(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash f]\!](\alpha^n))\, \mathbf{i}_\iota^-(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^n)) \dots \mathbf{i}_\iota^-(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^n))\, \overline{z} \\
&= C
\end{aligned}$$

Case $e = x\, e_1 \ldots e_\ell$ *with* $x \in V_{\mathrm{RS}}$ *of sort* $\tau = \tau_1 \to \tau_2$. For all $n \geq 0$:

$$
\begin{aligned}
B &= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash x']\!](\beta^n)\, \overline{T}\, \overline{w}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\beta^n(x')\, \overline{T}\, \overline{w}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&\sqsubseteq \lambda s.\, \exists \overline{r'}.\, \left(\mathbf{i}_\tau(\alpha^n(x))\, \overline{T'}\, \overline{\mathbf{i}^-(z)}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \qquad \dagger \\
&\sqsubseteq \lambda s.\, \exists \overline{r'}.\, \left(\mathbf{i}_\tau(\alpha^n(x))\, \overline{T'}\, \overline{\mathbf{i}^-(z)}\, s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \qquad \dagger \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\mathbf{i}_\iota(\alpha^n(x)\, \overline{S}\, \overline{z})\, s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \qquad \text{Lem 4.21} \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash x]\!](\alpha^n)\, \overline{S}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \\
&\sqsubseteq \lambda s.\, (\mathcal{H}[\![\mathcal{N}, \Gamma \vdash x\, \overline{e}]\!](\alpha^n)\, \overline{z} \sqsubseteq s) \qquad \text{Lem 4.29} \\
&= \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash x\, \overline{e}]\!](\alpha^n)\, \overline{z}) \\
&= \mathbf{i}_\tau(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash x\, \overline{e}]\!](\alpha^n))\, \overline{\mathbf{i}^-(z)} \qquad \text{Lem 4.21} \\
&= C
\end{aligned}
$$

Recall that $\mathsf{Incl}_{\mathsf{Rel}^-(\tau)}(\beta^n(x'), \mathbf{i}^-_\tau(\alpha^n(x)))$. The IH $\dagger$ gives us $\mathsf{Incl}_{\mathsf{Rel}^-(\tau_i)}(T_i, T'_i)$. Because we also have $\mathsf{Incl}_{\mathsf{Rel}^-(\sigma_j)}(w_j, \mathbf{i}^-_{\sigma_j}(z_j))$, we derive the first inclusion.

Case $e = F\, e_1 \ldots e_\ell$ *with* $F : \sigma \in \mathcal{N}$, *and* $n = 0$.

$$
\begin{aligned}
B &= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash R_F]\!](\beta^n)\, \overline{T}\, \overline{w}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&\sqsubseteq \lambda s.\, 1 \\
&= \mathbf{i}_\iota(\bot) \qquad \text{Lem 4.21} \\
&= \mathbf{i}_\iota(\bot_{\mathcal{H}[\![\sigma]\!]}\, \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^0) \ldots \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^0)\, \overline{z}) \\
&= \mathbf{i}_\iota(\alpha^0(F)\, \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^0) \ldots \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^0)\, \overline{z}) \\
&= \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F]\!](\alpha^0)\, \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^0) \ldots \mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^0)\, \overline{z}) \\
&= \mathbf{i}_\sigma(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F]\!](\alpha^0))\, \mathbf{i}^-_{\tau_1}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_1]\!](\alpha^0)) \ldots \mathbf{i}^-_{\tau_\ell}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash e_\ell]\!](\alpha^0))\, \overline{\mathbf{i}^-(z)} \\
&= C
\end{aligned}
$$

We have now established that $S(0, e'')$ holds for expressions all e''. The following case is the last remaining case to prove that $S(n+1, b)$ for all base case expressions b:

Case $e = F : \sigma \in \mathcal{N}$, *for* $n + 1$.

$$
\begin{aligned}
B &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F \urcorner]\!](\beta^{n+1})\,\overline{w} \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, R_F\, \overline{y}\, r]\!](\beta^{n+1})\,\overline{w} \\
&= \lambda s.\, \beta^{n+1}(R_F)\,\overline{w}\, s \\
&= \lambda s.\, \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^{n})\,\overline{w}\, s \\
&\sqsubseteq \lambda s.\, \mathbf{i}_{\sigma}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash \mathcal{R}(F)]\!](\alpha^{n}))\,\overline{\mathbf{i}^{-}(z)}\, s \quad \text{IH2} \\
&= \lambda s.\, \mathbf{i}_{\sigma}(\alpha^{n+1}(F))\,\overline{\mathbf{i}^{-}(z)}\, s \\
&= \mathbf{i}_{\sigma}(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F]\!](\alpha^{n+1}))\,\overline{\mathbf{i}^{-}(z)} \\
&= C
\end{aligned}
$$

Finally, we present the remaining case to prove the claim that $S(n', e'')$ for all $n' \geq 0$ and all expressions e''.

Case $e = F\, e_1 \dots e_\ell$ *with* $F \in \mathcal{N}$ *and* $\ell > 0$, *for* $n+1$.

$$
\begin{aligned}
B &= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash R_F]\!](\beta^{n+1})\, \overline{T}\, \overline{w}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\beta^{n+1}(R_F)\, \overline{T}\, \overline{w}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \mathcal{R}(F) \urcorner]\!](\beta^{n})\, \overline{T}\, \overline{w}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \\
&\sqsubseteq \lambda s.\, \exists \overline{r'}.\, \left(\mathbf{i}_\tau(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash \mathcal{R}(F)]\!](\alpha^n))\, \overline{T'}\, \overline{\mathbf{i}^-(z)}\, s \wedge \bigwedge_{i \in [\ell]} P_i \right) \qquad \text{IH2, } \dagger \\
&\sqsubseteq \lambda s.\, \exists \overline{r'}.\, \left(\mathbf{i}_\tau(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash \mathcal{R}(F)]\!](\alpha^n))\, \overline{T'}\, \overline{\mathbf{i}^-(z)}\, s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \qquad \dagger \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash \mathcal{R}(F)]\!](\alpha^n)\, \overline{S}\, \overline{z})\, s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \qquad \text{Lem 4.21} \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash \mathcal{R}(F)]\!](\alpha^n)\, \overline{S}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\alpha^{n+1}(F)\, \overline{S}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \\
&= \lambda s.\, \exists \overline{r'}.\, \left(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F]\!](\alpha^{n+1})\, \overline{S}\, \overline{z} \sqsubseteq s \wedge \bigwedge_{i \in [\ell]} P'_i \right) \\
&\sqsubseteq \mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F\, \overline{e}]\!](\alpha^{n+1})\, \overline{z}) \qquad \text{Lem 4.29} \\
&= \mathbf{i}_\tau(\mathcal{H}[\![\mathcal{N}, \Gamma \vdash F\, \overline{e}]\!](\alpha^{n+1}))\, \overline{\mathbf{i}^-(z)} \qquad \text{Lem 4.21} \\
&= C
\end{aligned}
$$

□

4.4.4 Main result: equality

Lemma 4.31. *For all typing judgements* $\mathcal{N} \vdash e : \sigma$ *of the HoRS* $\mathcal{G}$, *and non-increasing chains of valuations* $\mathcal{I} \subseteq \mathcal{M}[\![\Delta_{\mathcal{G}}]\!]$,

$$\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner e : \sigma \urcorner]\!] \left(\bigsqcap \mathcal{I} \right) = \bigsqcap_{I \in \mathcal{I}} \mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner e : \sigma \urcorner]\!](I).$$

Proof. Recall that $\mathcal{M}[\![\Delta_{\mathcal{G}}]\!]$ and $\mathcal{M}[\![\rho]\!]$ are complete lattices for each relational sort environment $\Delta_{\mathcal{G}}$ and relational sort ρ. Thus, we know that the greatest lower bounds exist.

To perform induction on the structure of HoRS term $e : \sigma$, we strengthen the claim to the following.

For all typing judgements $\mathcal{N}, \Gamma \vdash e : \sigma$ of the HoRS $\mathcal{G}$ where $\Gamma = \{x_1 : \tau_1, \ldots, x_k : \tau_k\}$, for all non-increasing chains of valuations $\mathcal{I} \subseteq \mathcal{M}[\![\Delta_{\mathcal{G}}]\!]$, and valuations $\theta \in \mathcal{M}[\![\ulcorner \Gamma \urcorner]\!]$,

$$\begin{aligned}&\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e : \sigma \urcorner]\!] \left(\left(\bigsqcap \mathcal{I}\right) [\overline{x'} \mapsto \overline{\theta(x')}]\right) \\ &= \bigsqcap_{I \in \mathcal{I}} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e : \sigma \urcorner]\!](I[\overline{x'} \mapsto \overline{\theta(x')}]).\end{aligned}$$

where $\ulcorner \Gamma \urcorner = \{x'_1 : \mathsf{Rel}^-(\tau_1), \ldots, x'_k : \mathsf{Rel}^-(\tau_k)\}$. We abbreviate $I[\overline{x'} \mapsto \overline{\theta(x')}]$ to I'. Note that

$$\left(\bigsqcap \mathcal{I}\right) [\overline{x'} \mapsto \overline{\theta(x')}] = \bigsqcap_{I \in \mathcal{I}} I[\overline{x'} \mapsto \overline{\theta(x')}] = \bigsqcap_{I \in \mathcal{I}} I',$$

so that we shorten the above equation to

$$\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e : \sigma \urcorner]\!] \left(\bigsqcap_{I \in \mathcal{I}} I'\right) = \bigsqcap_{I \in \mathcal{I}} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner e : \sigma \urcorner]\!](I').$$

Case $e = f \in \Sigma$. The meaning of $\ulcorner f \urcorner$ is independent of the valuation, as demonstrated by

$$\ulcorner f \urcorner = \lambda y_1 \ldots y_k\, r.\, (f\, y_1 \ldots y_k = r)$$

where $f : \iota^k \to \iota$. Thus, this case trivially holds.

Case $x \in V_{\mathrm{RS}}$. The meaning of $\ulcorner x \urcorner$ relies only on the θ part of the valuation, as evident from

$$\begin{aligned}\ulcorner x : \iota \urcorner &= \lambda r.\, (x' = r) \\ \ulcorner x : \sigma_1 \to \cdots \to \sigma_k \to \iota \urcorner &= \lambda y_1 \ldots y_k\, r.\, x'\, y_1 \ldots y_k\, r\end{aligned}$$

for $k > 0$. Thus, this case trivially holds.

Case $F \in \mathcal{N}$. Let $F : \sigma_1 \to \cdots \to \sigma_k \to \iota$ for some $k \geq 0$.

$$\begin{aligned}
\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F \urcorner]\!] \left(\prod_{I \in \mathcal{I}} I'\right) &= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda y_1 \ldots y_k\, r.\, R_F\, y_1 \ldots y_k\, r]\!] \left(\prod_{I \in \mathcal{I}} I'\right) \\
&= \lambda y_1 \ldots y_k\, r.\, \left(\prod_{I \in \mathcal{I}} I'\right) (R_F)\, y_1 \ldots y_k\, r \\
&= \lambda y_1 \ldots y_k\, r.\, \left(\prod_{I \in \mathcal{I}} I\right) (R_F)\, y_1 \ldots y_k\, r \\
&= \lambda y_1 \ldots y_k\, r.\, \left(\prod_{I \in \mathcal{I}} I(R_F)\right)\, y_1 \ldots y_k\, r \qquad \text{Prop 2.7} \\
&= \prod_{I \in \mathcal{I}} I(R_F) \\
&= \prod_{I \in \mathcal{I}} I'(R_F) \\
&= \prod_{I \in \mathcal{I}} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash R_F]\!](I') \\
&= \prod_{I \in \mathcal{I}} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda y_1 \ldots y_k\, r.\, R_F\, y_1 \ldots y_k\, r]\!](I') \\
&= \prod_{I \in \mathcal{I}} \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner F \urcorner]\!](I')
\end{aligned}$$

Note that the application of Proposition 2.7 is warranted by the codomain of $\mathcal{I}$ being a complete lattice (namely, a finite product of complete lattices $\mathcal{M}[\![\rho]\!]$).

Case $e = \$\,\overline{e}$ *with* $\overline{e} = e_1 \ldots e_\ell$ *for* $\ell > 0$. This case follows from applying the induction hypothesis in a straightforward though laborious unfolding of the lift and the semantics. Recall that:

$$\begin{aligned}
&\mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \ulcorner \$\,\overline{e} \urcorner]\!] \left(\prod_{I \in \mathcal{I}} I'\right) \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}}, \ulcorner \Gamma \urcorner \vdash \lambda \overline{y}\, r.\, \exists \overline{r}.\, \$' \urcorner (e_1, r_1) \urcorner \ldots \urcorner (e_\ell, r_\ell) \urcorner \overline{y}\, r \wedge \bigwedge_{i \in [\ell]} Prop(e_i, r_i)]\!] \left(\prod_{I \in \mathcal{I}} I'\right)
\end{aligned}$$

The previous cases show that the greatest lower bound is preserved by $\mathcal{M}[\![\$']\!]$. Observe that $\urcorner (e_i, r_i) \urcorner$ is r_i or $\ulcorner e_i \urcorner$. Either way, the greatest lower bound is preserved by $\mathcal{M}[\![\urcorner (e_i, r_i) \urcorner]\!]$. Similarly, $Prop(e_i, r_i)$ is either $\ulcorner e_i \urcorner\, r_i$ or true, and the greatest lower bound is thus preserved by $\mathcal{M}[\![Prop(e_i, r_i)]\!]$. This concludes the proof. □

Theorem 4.32. $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash R_S]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}))\, t = 1$ *if and only if* $t = [\![\mathcal{G}]\!]$.

Proof. Given a HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$ we define, for all $n \geq 0$:

$$\alpha^n := \mathcal{H}[\![\mathcal{G}]\!]^n_{\mathcal{N}}(\bot_{\mathcal{N}}) \in \mathcal{H}[\![\mathcal{N}]\!]$$
$$\beta^n := T^{\mathcal{M}\,n}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}(\top_{\Delta_{\mathcal{G}}}) \in \mathcal{M}[\![\Delta_{\mathcal{G}}]\!]$$

It holds that:

$$\begin{aligned}
\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash R_S]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}})) &= \mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}})) \\
&= \mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!]\left(\bigsqcap \beta^n\right) \\
&= \bigsqcap\{\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\beta^n) \mid n \geq 0\} \qquad \text{Lem 4.31} \\
&\sqsubseteq \bigsqcap\{\mathbf{i}_\iota(\mathcal{H}[\![\mathcal{N} \vdash S]\!](\alpha^n)) \mid n \geq 0\} \qquad \text{Lem 4.30} \\
&= \mathbf{i}_\iota\left(\bigsqcup\{\mathcal{H}[\![\mathcal{N} \vdash S]\!](\alpha^n) \mid n \geq 0\}\right) \qquad \text{Lem 4.22} \\
&= \mathbf{i}_\iota\left(\mathcal{H}[\![\mathcal{N} \vdash S]\!]\left(\bigsqcup \alpha^n\right)\right) \qquad \text{Lem 4.10} \\
&= \mathbf{i}_\iota([\![\mathcal{G}]\!]) \\
&= \lambda r.\,([\![\mathcal{G}]\!] = r) \qquad \bot\text{-freeness}
\end{aligned}$$

Note that we rely on Proposition 2.4 to apply Lemma 4.22 in the above.

Either $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}))$ is the constant false function, or it is $\lambda r.\,([\![\mathcal{G}]\!] = r)$. By Lemma 4.27, $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}))$ is not $\lambda r.\,0$, so we conclude that it is $\lambda r.\,([\![\mathcal{G}]\!] = r)$, instead.

It follows that $\mathcal{M}[\![\Delta_{\mathcal{G}} \vdash \ulcorner S \urcorner]\!](\mathsf{gfp}(T^{\mathcal{M}}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}))\,t = 1$ if and only if $t = [\![\mathcal{G}]\!]$. □

4.5 Examples

In this section, we provide examples of an order-0, order-1, and order-2 HoRS and their encodings into (coinductive) HoCHC logic programs. We outline the correctness of the encoded program in each example. The HoRS-to-HoCHC encoding is defined and described in Section 4.3 and its correctness in Section 4.4.

Notation 4.33. For HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$, we write

$$H^n_{\mathsf{co}} := \mathcal{H}[\![\mathcal{G}]\!]^n_{\mathcal{N}}(\bot_{\mathcal{N}}) \qquad T^n_{\mathsf{co}} := T^{\mathcal{M}\,n}_{P_{\mathcal{G}}:\Delta_{\mathcal{G}}}(\top_{\Delta_{\mathcal{G}}})$$

for all $n \geq 0$.

Example 4.34. Consider the order-0 HoRS $\mathcal{G}_0 = \langle \{a\}, \{S\}, \mathcal{R}_0, S \rangle$ where $\mathcal{R}_0$ consists of:

$$S = a\, S$$

This rewrite rule is lifted as follows:

$$\begin{aligned} \ulcorner a\, S \urcorner &= \lambda r.\, \exists r_1.\, (a\, r_1 = r) \wedge \ulcorner S \urcorner\, r_1 \\ \ulcorner S \urcorner &= \lambda r_2.\, R_S\, r_2 \end{aligned}$$

By β-equivalence, this is equivalent to

$$\ulcorner \mathcal{R}_0(S) \urcorner = \ulcorner a\, S \urcorner = \lambda r.\, \exists r_1.\, (a\, r_1 = r) \wedge R_S\, r_1$$

which means that

$$\begin{aligned} H^0_{\mathsf{co}}(S) &= \bot \\ H^{n+1}_{\mathsf{co}}(S) &= a^{n+1}\, \bot \\ \bigsqcup \{H^n_{\mathsf{co}}(S) \mid n \geq 0\} &= a^\omega \\ T^0_{\mathsf{co}}(R_S) &= \lambda r.\, 1 \\ T^{n+1}_{\mathsf{co}}(R_S) &= \lambda r.\, \exists s.\, (a^{n+1}\, s = r) \\ \bigsqcap \{T^n_{\mathsf{co}}(R_S) \mid n \geq 0\} &= \lambda r.\, (a^\omega = r). \end{aligned}$$

Correctness follows from

$$\begin{aligned} T^0_{\mathsf{co}}(R_S) &= \lambda r.\, 1 \\ &= \mathbf{i}_\iota(\bot) \\ &= \mathbf{i}_\iota(H^0_{\mathsf{co}}(S)) \\ T^{n+1}_{\mathsf{co}}(R_S) &= \lambda r.\, (a^{n+1}\, \bot \sqsubseteq t) \\ &= \mathbf{i}_\iota(a^{n+1}\, \bot) \\ &= \mathbf{i}_\iota(H^{n+1}_{\mathsf{co}}(S)) \end{aligned}$$

and finally

$$\bigsqcap \{T^n_{\mathsf{co}}(R_S) \mid n \geq 0\} = \lambda r.\, (a^\omega = r) = \mathbf{i}_\iota(a^\omega) = \mathbf{i}_\iota \left(\bigsqcup \{H^n_{\mathsf{co}}(S) \mid n \geq 0\} \right).$$

Example 4.35. Consider the order-1 HoRS $\mathcal{G}_1 = \langle \{a, b, c\}, \{S, F\}, \mathcal{R}_1, S \rangle$ where $\mathcal{R}_1$ consists of:

$$\begin{aligned} S &= F\, c \\ F &= \lambda x.\, a\, x\, (F\, (b\, x)) \end{aligned}$$

These rewrite rules are lifted as follows:

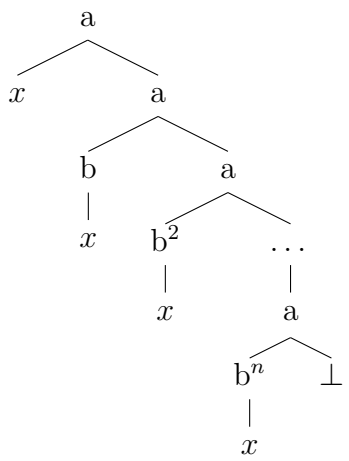
$$\begin{aligned}
\ulcorner F\, c \urcorner &= \lambda r.\, \exists r_1.\, R_F\, r_1\, r \wedge \ulcorner c \urcorner\, r_1 \\
\ulcorner c \urcorner &= \lambda r_2.\, (c = r_2) \\
\ulcorner \lambda x.\, a\, x\, (F\, (b\, x)) \urcorner &= \lambda x'\, r_3.\, \exists r_4\, r_5.\, (a\, r_4\, r_5 = r_3) \wedge \ulcorner x \urcorner\, r_4 \wedge \ulcorner F\, (b\, x) \urcorner\, r_5 \\
\ulcorner x \urcorner &= \lambda r_6.\, (x' = r_6) \\
\ulcorner F\, (b\, x) \urcorner &= \lambda r_7.\, \exists r_8.\, R_F\, r_8\, r_7 \wedge \ulcorner b\, x \urcorner\, r_8 \\
\ulcorner b\, x \urcorner &= \lambda r_9.\, \exists r_{10}.\, (b\, r_{10} = r_9) \wedge \ulcorner x \urcorner\, r_{10}
\end{aligned}$$

By $\alpha\beta$-equivalence and moving around existential quantifiers, this is equivalent to:

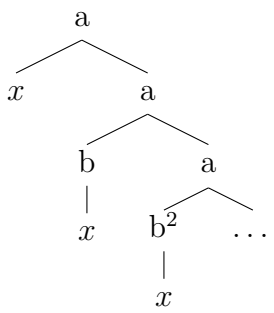
$$\begin{aligned}
\ulcorner \mathcal{R}_2(S) \urcorner &= \ulcorner F\, c \urcorner \\
&= \lambda r.\, \exists r_1.\, R_F\, r_1\, r \wedge (c = r_1) \\
\ulcorner \mathcal{R}_2(F) \urcorner &= \ulcorner \lambda x.\, a\, x\, (F\, (b\, x)) \urcorner \\
&= \lambda x'\, r_2.\, \exists r_{3-6}.\, (a\, r_3\, r_4 = r_2) \wedge (x' = r_3) \wedge R_F\, r_5\, r_4 \wedge (b\, r_6 = r_5) \wedge (x' = r_6)
\end{aligned}$$

We define a family of trees, for $n \geq 0$,

$t_n[x, \bot]$:=



$t_\infty[x]$:=



which means that

$$\begin{aligned}
H^0_{\mathsf{co}}(S) = H^1_{\mathsf{co}}(S) &= \bot \\
H^{n+2}_{\mathsf{co}}(S) &= t_n[c, \bot] \\
\bigsqcup \{H^n_{\mathsf{co}}(S) \mid n \geq 0\} &= t_\infty[c] \\
H^0_{\mathsf{co}}(F) &= \lambda x.\, \bot \\
H^{n+1}_{\mathsf{co}}(F) &= \lambda x.\, t_n[x, \bot] \\
\bigsqcup \{H^n_{\mathsf{co}}(F) \mid n \geq 0\} &= \lambda x.\, t_\infty[x]
\end{aligned}$$

and

$$\begin{aligned}
T^0_{\mathsf{co}}(R_S) = T^1_{\mathsf{co}}(R_S) &= \lambda r.\, 1 \\
T^{n+2}_{\mathsf{co}}(R_S) &= \lambda r.\, \exists r'.\, (t_n[c, r'] = r) \\
\bigsqcap\{T^n_{\mathsf{co}}(R_S) \mid n \geq 0\} &= \lambda r.\, (t_\infty[c] = r) \\
T^0_{\mathsf{co}}(R_F) &= \lambda x\, r.\, 1 \\
T^{n+1}_{\mathsf{co}}(R_F) &= \lambda x\, r.\, \exists r'.\, (t_n[x, r'] = r) \\
\bigsqcap\{T^n_{\mathsf{co}}(R_F) \mid n \geq 0\} &= \lambda x\, r.\, (t_\infty[x] = r).
\end{aligned}$$

Correctness follows from

$$\begin{aligned}
T^0_{\mathsf{co}}(R_S) = T^1_{\mathsf{co}}(R_S) &= \mathbf{i}_\iota(H^0_{\mathsf{co}}(S)) = \mathbf{i}_\iota(H^1_{\mathsf{co}}(S)) \\
T^{n+2}_{\mathsf{co}}(R_S) &= \lambda r.\, (t_n[c, \bot] \sqsubseteq t) \\
&= \mathbf{i}_\iota(t_n[c, \bot]) \\
&= \mathbf{i}_\iota(H^{n+2}_{\mathsf{co}}(S)) \\
T^0_{\mathsf{co}}(R_F)\, \mathbf{i}^-_\iota(z) &= \lambda r.\, 1 \\
&= \mathbf{i}_\iota(\bot) \\
&= \mathbf{i}_\iota(H^0_{\mathsf{co}}(F)\, z) \\
T^{n+1}_{\mathsf{co}}(R_F)\, \mathbf{i}^-_\iota(z) &= \lambda r.\, \exists r'.\, (t_n[z, r'] = r) \\
&\sqsubseteq \lambda r.\, (t_n[z, \bot] \sqsubseteq r) \\
&= \mathbf{i}_\iota(t_n[z, \bot]) \\
&= \mathbf{i}_\iota(H^{n+1}_{\mathsf{co}}(F)\, z)
\end{aligned}$$

for all $z \in \mathcal{H}[\![\iota]\!]$, and finally

$$\bigsqcap\{T^n_{\mathsf{co}}(R_S) \mid n \geq 0\} = \lambda r.\, (t_\infty[c] = r) = \mathbf{i}_\iota(t_\infty[c]) = \mathbf{i}_\iota\left(\bigsqcup\{H^n_{\mathsf{co}}(S) \mid n \geq 0\}\right).$$

Example 4.36. Consider the order-2 HoRS $\mathcal{G}_2 = \langle\{f, g, a\}, \{S, B, F\}, \mathcal{R}_2, S\rangle$ where $\mathcal{R}_2$ consists of:

$$\begin{aligned}
S &= F\, g \\
B &= \lambda\varphi\, \psi\, x.\, \varphi\, (\psi\, x) \\
F &= \lambda\varphi.\, f\, (\varphi\, a)\, (F\, (B\, \varphi\, \varphi))
\end{aligned}$$

These rewrite rules are lifted as follows:

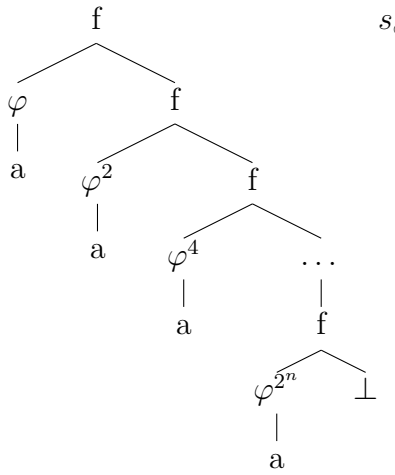
$$\begin{aligned}
\ulcorner F\, g \urcorner &= \lambda r.\, \exists r_1.\, R_F\, \ulcorner g \urcorner\, r \wedge \mathsf{true} \\
\ulcorner g \urcorner &= \lambda y\, r_1.\, (g\, y = r_1) \\
\ulcorner \lambda \varphi\, \psi\, x.\, \varphi\, (\psi\, x) \urcorner &= \lambda \varphi'\, \psi'\, x'\, r_2.\, \exists r_3.\, \varphi'\, r_3\, r_2 \wedge \ulcorner \psi\, x \urcorner\, r_3 \\
\ulcorner \psi\, x \urcorner &= \lambda r_4.\, \exists r_5.\, \psi'\, r_5\, r_4 \wedge \ulcorner x \urcorner\, r_5 \\
\ulcorner x \urcorner &= \lambda r_6.\, (x' = r_6) \\
\ulcorner \lambda \varphi.\, f\, (\varphi\, a)\, (F\, (B\, \varphi\, \varphi)) \urcorner &= \lambda \varphi'\, r_7.\, \exists r_8\, r_9.\, (f\, r_8\, r_9 = r_7) \wedge \ulcorner \varphi\, a \urcorner\, r_8 \wedge \ulcorner F\, (B\, \varphi\, \varphi) \urcorner\, r_9 \\
\ulcorner \varphi\, a \urcorner &= \lambda r_{10}.\, \exists r_{11}.\, \varphi'\, r_{11}\, r_{10} \wedge \ulcorner a \urcorner\, r_{11} \\
\ulcorner a \urcorner &= \lambda r_{12}.\, (a = r_{12}) \\
\ulcorner F\, (B\, \varphi\, \varphi) \urcorner &= \lambda r_{13}.\, \exists r_{14}.\, R_F\, \ulcorner B\, \varphi\, \varphi \urcorner\, r_{13} \wedge \mathsf{true} \\
\ulcorner B\, \varphi\, \varphi \urcorner &= \lambda y_1\, r_{15}.\, \exists r_{16}\, r_{17}.\, R_B\, \ulcorner \varphi \urcorner\, \ulcorner \varphi \urcorner\, y_1\, r_{15} \wedge \mathsf{true} \wedge \mathsf{true} \\
\ulcorner \varphi \urcorner &= \lambda y_2\, r_{18}.\, \varphi'\, y_2\, r_{18}
\end{aligned}$$

After simplification, this becomes:

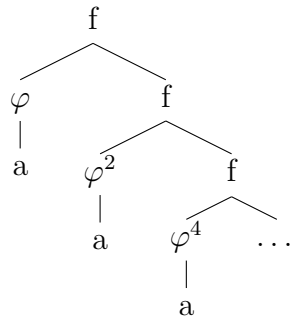
$$\begin{aligned}
\ulcorner \mathcal{R}_2(S) \urcorner &= \lambda r.\, R_F\, (\lambda y\, r_1.\, g\, y = r_1)\, r \\
\ulcorner \mathcal{R}_2(B) \urcorner &= \lambda \varphi'\, \psi'\, x'\, r_2.\, \exists r_3\, r_4.\, \varphi'\, r_3\, r_2 \wedge \psi'\, r_4\, r_3 \wedge (x' = r_4) \\
\ulcorner \mathcal{R}_2(F) \urcorner &= \lambda \varphi'\, r_5.\, \exists r_{6-8}.\, (f\, r_6\, r_7 = r_5) \wedge \varphi'\, r_8\, r_6 \wedge (a = r_8) \wedge R_F\, (R_B\, \varphi'\, \varphi')\, r_7
\end{aligned}$$

We define a family of trees, for $n \geq 0$,

$s_n[\varphi, \bot]$:=



$s_\infty[\varphi]$:=



which means that:

$$
\begin{aligned}
H^0_{\mathsf{co}}(S) = H^1_{\mathsf{co}}(S) &= \bot \\
H^{n+2}_{\mathsf{co}}(S) &= s_n[g, \bot] \\
\bigsqcup \{H^n_{\mathsf{co}}(S) \mid n \geq 0\} &= s_\infty[g] \\
H^0_{\mathsf{co}}(F) &= \lambda\varphi.\, \bot \\
H^{n+1}_{\mathsf{co}}(F) &= \lambda\varphi.\, s_n[\varphi, \bot] \\
\bigsqcup \{H^n_{\mathsf{co}}(F) \mid n \geq 0\} &= \lambda\varphi.\, s_\infty[\varphi] \\
H^0_{\mathsf{co}}(B) &= \lambda\varphi\, \psi\, x.\, \bot \\
H^{n+1}_{\mathsf{co}}(B) = \bigsqcup \{H^n_{\mathsf{co}}(B) \mid n \geq 0\} &= \lambda\varphi\, \psi\, x.\, \varphi\, (\psi\, x) \\
T^0_{\mathsf{co}}(R_S) = T^1_{\mathsf{co}}(R_S) &= \lambda r.\, 1 \\
T^{n+2}_{\mathsf{co}}(R_S) &= \lambda r.\, \exists r_1.\, (s_n[g, r_1] = r) \\
\bigsqcap \{T^n_{\mathsf{co}}(R_S) \mid n \geq 0\} &= \lambda r.\, (s_\infty[g] = r) \\
T^0_{\mathsf{co}}(R_F) &= \lambda\varphi'\, r.\, 1 \\
T^{n+1}_{\mathsf{co}}(R_F) &= \lambda\varphi'\, r.\, \exists r_1.\, (s_n[\varphi, r_1] = r) \\
\bigsqcap \{T^n_{\mathsf{co}}(R_F) \mid n \geq 0\} &= \lambda\varphi'\, r.\, (s_\infty[\varphi] = r) \\
T^0_{\mathsf{co}}(R_B) &= \lambda\varphi'\, \psi'\, x'\, r.\, 1 \\
T^{n+1}_{\mathsf{co}}(R_B) = \bigsqcap \{T^n_{\mathsf{co}}(R_B) \mid n \geq 0\} &= \lambda\varphi'\, \psi'\, x'\, r.\, (\varphi\, (\psi\, x) = r)
\end{aligned}
$$

Note that we have taken some liberties with notation. Correctness follows from

$$
\begin{aligned}
T^0_{\mathsf{co}}(R_S) = T^1_{\mathsf{co}}(R_S) &= \mathbf{i}_\iota(H^0_{\mathsf{co}}(S)) = \mathbf{i}_\iota(H^1_{\mathsf{co}}(S)) \\
T^{n+2}_{\mathsf{co}}(R_S) &= \lambda r.\, (s_n[g, \bot] \sqsubseteq r) \\
&= \mathbf{i}_\iota(s_n[g, \bot]) \\
&= \mathbf{i}_\iota(H^{n+2}_{\mathsf{co}}(S)) \\
T^0_{\mathsf{co}}(R_F)\, \mathbf{i}^-_\iota(z_1) &= \mathbf{i}_\iota(H^0_{\mathsf{co}}(F)\, z_1) \\
T^{n+1}_{\mathsf{co}}(R_F)\, \mathbf{i}^-_\iota(z_1) &= \lambda r.\, \exists r_1.\, (s_n[z_1, r_1] = r) \\
&\sqsubseteq \lambda r.\, (s_n[z_1, \bot] \sqsubseteq r) \\
&= \mathbf{i}_\iota(s_n[z_1, \bot]) \\
&= \mathbf{i}_\iota(H^{n+1}_{\mathsf{co}}(F)\, z_1)
\end{aligned}
$$

and finally

$$\begin{aligned} T^0_{\mathsf{co}}(R_B)\,\mathbf{i}^-_{\iota\to\iota}(z_1)\,\mathbf{i}^-_{\iota\to\iota}(z_2)\,\mathbf{i}^-_{\iota}(z_3) &= \mathbf{i}_\iota(H^0_{\mathsf{co}}(B)\,z_1\,z_2\,z_3) \\ T^{n+1}_{\mathsf{co}}(R_B)\,\mathbf{i}^-_{\iota\to\iota}(z_1)\,\mathbf{i}^-_{\iota\to\iota}(z_2)\,\mathbf{i}^-_{\iota}(z_3) &= \lambda r.\,(z_1\,(z_2\,z_3) = r) \\ &\sqsubseteq \lambda r.\,(z_1\,(z_2\,z_3) \sqsubseteq r) \\ &= \mathbf{i}_\iota(z_1\,(z_2\,z_3)) \\ &= \mathbf{i}_\iota(H^{n+1}_{\mathsf{co}}(B)\,z_1\,z_2\,z_3) \end{aligned}$$

for all $z_1, z_2 \in \mathcal{H}[\![\iota \to \iota]\!]$ and $z_3 \in \mathcal{H}[\![\iota]\!]$. It follows that:

$$\bigsqcap\{T^n_{\mathsf{co}}(R_S) \mid n \geq 0\} = \lambda r.\,(s_\infty[g] = r) = \mathbf{i}_\iota(s_\infty[g]) = \mathbf{i}_\iota\left(\bigsqcup\{H^n_{\mathsf{co}}(S) \mid n \geq 0\}\right)$$

4.6 HoRS equivalence problem

The higher-order recursion scheme (HoRS) equivalence problem asks whether two given deterministic recursion schemes $\mathcal{G}_1, \mathcal{G}_2$ generate the same tree (i.e. whether $[\![\mathcal{G}_1]\!] = [\![\mathcal{G}_2]\!]$, see e.g. Ong, 2015).

The HoRS equivalence problem is recursively equivalent to the $\lambda\mathbf{Y}$-calculus Böhm tree equivalence problem, which asks whether the Böhm trees of two given $\lambda\mathbf{Y}$-terms are equal (Clairambault and Murawski, 2013). The question of the decidability of the latter "has been there from the beginning of the subject" (Walukiewicz, 2016). To the best of our knowledge, this problem is still open. Notice, however, that the related $\lambda\mathbf{Y}$-calculus word problem (whether two $\lambda\mathbf{Y}$-terms are $\beta\eta\mathbf{Y}$-equivalent) is undecidable (Statman, 2004).

Restricted to order 1, the HoRS equivalence problem is equivalent to the DPDA equivalence problem (Courcelle, 1978). Thus, the fundamental result of Sénizergues (2001), and subsequent refinements by Stirling (2001), Sénizergues (2002), and Jancar (2012) provide a decision procedure for the equivalence of first-order HoRS.

In this section, we reduce decidability of the HoRS equivalence problem to semi-decidability of coinductive HoCHC. Our procedure formulates a positive and negative instance of the (monotone) coinductive HoCHC problem over a decidable background theory, corresponding to equivalence and inequivalence of the input HoRS, respectively. We present the background theory in Section 4.6.1 and the two HoCHC instances in Section 4.6.3.

If coinductive HoCHC is semi-decidable over a decidable background theory—or specifically if the image of our HoRS-to-HoCHC encoding is semi-decidable over Maher's theory of trees (Maher, 1988)—then these two instances can be solved concurrently to obtain a full decision procedure for the HoRS equivalence problem.

Theorem 4.37. *The HoRS equivalence problem is decidable if the HoRS-to-HoCHC encoding lives in a semi-decidable fragment of coinductive HoCHC over Maher's complete and decidable theory of trees.*

Recall that our HoRS-to-HoCHC encoding from Section 4.3 assumes an input HoRS that generates a $\bot$-free tree. This allows us to distinguish 'finished' trees from 'unfinished' trees, as required for correctness and described in Section 4.3. Therefore, we first transform a HoRS into a $\bot$-free tree generating HoRS.

The 'decision procedure' for the HoRS equivalence problem consists of the following stages:

1. $\bot$-free transform on HoRS (Sec 4.6.2)
2. Encoding HoRS into HoCHC (Sec 4.3)
3. Concurrently solving the positive and negative HoCHC instance (pending semi-decidability)

4.6.1 Maher's theory of trees

A *theory* T is a set of *sentences* (i.e. closed formulas), which is *complete* if either $T \vDash \varphi$ or $T \vDash \neg\varphi$, for every sentence φ. An *axiomatisation* of an algebra $\mathfrak{A}$ is a recursive set of sentences which are true of $\mathfrak{A}$. The *theory of an algebra* $\mathfrak{A}$ is a the set of all sentences true of $\mathfrak{A}$.

Maher's first-order equational theory of trees, denoted by T_Σ, is complete for any finite or infinite alphabet Σ (Maher, 1988). It is axiomatised by three axioms:

$$\forall f \in \Sigma. \qquad \forall \overline{x}\,\overline{y}.\, f\,\overline{x} = f\,\overline{y} \leftrightarrow \overline{x} = \overline{y} \tag{4.1}$$

$$\forall f, g \in \Sigma. \qquad f \neq g \rightarrow \forall \overline{x}\,\overline{y}.\, f\,\overline{x} \neq g\,\overline{y} \tag{4.2}$$

$$\forall (x = t(x, y)). \qquad z = t(z, y) \rightarrow x = z \tag{4.3}$$

where $x = t(x, y)$ ranges over all rational solved forms (see Maher, 1988, p. 355).

The first two axioms specify what it means for two trees to be equal: they have the same root (4.2) and the children are equal (4.1). Furthermore, if two trees have the same 'definition' (rational solved form), then they must be equal (4.3).

In case Σ is finite, we need to add the Domain Closure Axiom to obtain completeness:

$$\forall x.\ \bigvee_{f \in \Sigma} \exists \overline{z}.\ x = f\ \overline{z} \tag{DCA}$$

Fix a ranked alphabet Σ, viewed as tree constructors. We write T_Σ for the first-order theory of equations of finite and infinite trees constructed from Σ. The theory T_Σ has good algorithmic properties, making it an exceedingly appropriate choice of background theory for HoCHC. Maher (1988) first showed that the theory is not only complete but also decidable. The theory has several models, including the set of finite and infinite trees over Σ—which we are interested in—and the set of rational trees. Henceforth, we call T_Σ the *Maher theory*.

Theorem 4.38 (Maher, 1988; Djelloul et al., 2008)**.** *The Maher theory of finite and infinite trees over Σ, T_Σ, is complete and decidable. It has several models, including the set of finite and infinite Σ-labelled trees.*

More recently Djelloul et al. (2008) presented a so-called full first-order constraint solver for (an augmented version of) the theory T_Σ. Their algorithm transforms a first-order formula of T_Σ into a disjunction of simple formulas; each disjunct is either the formula *true*, the formula *false*, or a formula with at least one free variable (which is equivalent to neither *true* nor *false*) with an explicit description of the solutions of the free variables. Questions of expressivity and complexity of the Maher theory are explored in Colmerauer and Dao (2003). Recent work by Zaiser and Ong (2020) has improved the performance of Djelloul et al. (2008)'s solver and adapted the theory to algebraic (co)data types.

We are interested in the Maher theory $T_{\Sigma_\perp}$ over a finite alphabet $\Sigma_\perp = \Sigma \cup \{\perp\}$, for input HoRS $\mathcal{G}_1 = \langle \mathcal{N}_1, \Sigma, \mathcal{R}_1, S_1 \rangle$ and $\mathcal{G}_2 = \langle \mathcal{N}_2, \Sigma, \mathcal{R}_2, S_2 \rangle$. Note that the assumption that both HoRS have the same alphabet Σ is WLOG; if they have distinct alphabets Σ_1 and Σ_2, respectively, we can take Σ to be their union. In the Maher theory $T_{\Sigma_\perp}$, the 'unfinished' tree $\perp$ is treated as any other nullary terminal symbol.

4.6.2 Computability of $\bot$-free transform of HoRS

Intuitively, eliminating $\bot$ from $[\![\mathcal{G}]\!]$ allows us to distinguish 'unfinished' trees from 'finished' (but diverging) trees in e.g. the proofs in Section 4.4.

As usual, let $\Sigma_\bot$ be a finite ranked alphabet Σ extended with (nullary) $\bot$. Let $b : \iota \to \iota \notin \Sigma_\bot$ (for 'bottom') be a fresh terminal symbol.

Definition 4.39. *Given a $\Sigma_\bot$-labelled tree, its* $\bot$-free conversion *is obtained by replacing every $\bot$-labelled node by the infinite linear tree $b\ (b\ (b\ \cdots))$.*

Lemma 4.40 (Computability of $\bot$-free transform of HoRS)**.** *There is an algorithm that, given a HoRS $\mathcal{G}$, returns a HoRS—call it the* $\bot$-free transform *of $\mathcal{G}$—that generates the $\bot$-free conversion of $[\![\mathcal{G}]\!]$.*

For clarity, we *convert* trees, but *transform* HoRS (their generators).

It is clear from freshness of $b : \iota \to \iota \notin \Sigma_\bot$ that the following holds.

Proposition 4.41. *Two HoRS are equivalent if and only if their respective $\bot$-free transforms are equivalent.*

Before we present a three-stage algorithm to transform a HoRS $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$ to its $\bot$-free transform and an example, we require some background on logical reflection.

Some background on HoRS and logical reflection. Let $\mathfrak{R}$ be a class of generators of Σ-labelled trees, and $\mathcal{L}$ be a logical language for describing correctness properties of these trees. Define the ranked alphabet $\Sigma' := \{\underline{f} : \sigma \mid f : \sigma \in \Sigma\}$ that is a copy of Σ. Given a generator $\mathcal{G} \in \mathfrak{R}$ and property $\varphi \in \mathcal{L}$, we say that $\mathcal{G}_\varphi$ is a *φ-reflection* of $\mathcal{G}$ just if

1. $\mathcal{G}$ and $\mathcal{G}_\varphi$ generate the same underlying tree, and
2. if node α of the tree $[\![\mathcal{G}]\!]$ has label f, then node α of $[\![\mathcal{G}_\varphi]\!]$ is labelled $\underline{f}$ if α satisfies φ and f otherwise.

We say that $\mathfrak{R}$ is *reflective w.r.t. $\mathcal{L}$* just if there is an algorithm that transforms a given pair $\langle \mathcal{G}, \varphi \rangle$ to $\mathcal{G}_\varphi$.

Theorem 4.42 (Broadbent et al., 2010)**.** *HoRS are reflective w.r.t. modal μ-calculus and monadic second-order logic.*

Stage 1: $\Sigma \cup \{b\}$-labelling $\mathcal{G}_1$. The input HoRS $\mathcal{G}$ is first transformed to a *b-productive* counterpart $\mathcal{G}_1 := \langle \mathcal{N}, \Sigma \cup \{b\}, \mathcal{R}', S \rangle$. The idea is that in the potentially infinite process of generating the tree $[\![\mathcal{G}_1]\!]$ from the start nonterminal S by leftmost-outermost rewriting, each rewriting step is witnessed by either a terminal symbol from Σ or by b. The set $\mathcal{R}'$ of rewrite rules of $\mathcal{G}_1$ is defined as follows. For every $F : \sigma_1 \to \cdots \to \sigma_n \to \iota \in \mathcal{N}$, let $\overline{x} = x_1 \dots x_n$ so that:

- if $\mathcal{R}(F) = \lambda \overline{x}.\, f\, t_1 \dots t_m$ for some $f \in \Sigma$, then $\mathcal{R}'(F) := \mathcal{R}(F)$
- if $\mathcal{R}(F) = \lambda \overline{x}.\, \$\, t_1 \dots t_m$ for $\$ \in \mathcal{N} \cup V_{\mathrm{RS}}$, then $\mathcal{R}'(F) := \lambda \overline{x}.\, b\, (\$\, t_1 \dots t_m)$

Notice that the tree $[\![\mathcal{G}_1]\!]$, by construction, does not have any $\bot$-labelled nodes. Intuitively we can get $[\![\mathcal{G}]\!]$ back from $[\![\mathcal{G}_1]\!]$ by erasing finite b^*, and replacing infinite b^ω by $\bot$.

Stage 2: From $\Sigma \cup \{b\}$-labelling $\mathcal{G}_1$ to $\Sigma \cup \{b, s\}$-labelling $\mathcal{G}_2$. We define a modal μ-calculus formula:

$$\varphi := p_b \wedge \mu X. \left(\bigvee_{f \in \Sigma} ①\, p_f \vee ①\, X \right)$$

where p_b (resp. p_f for $f \in \Sigma$) is a propositional variable that denotes that a node is labelled with b (resp. $f \in \Sigma$). Because μ is a least fixpoint operator and $①\, P$ means that predicate P holds for the leftmost child of a node, this formula holds for b-labelled nodes (p_b holds) that are not part of some infinite branch b^ω (after finite time we encounter a descendant that is labelled from Σ, not b). Please refer to Kaivola (1995) for the precise syntax and semantics of the modal μ-calculus.

Let $s : \iota \to \iota \notin \Sigma_\bot$ (for 'step') be another fresh terminal symbol. Consider the following operation on $\Sigma \cup \{b\}$-labelled trees.

> For every node α, if $\alpha \vDash \varphi$ then rewrite the label at α to s, otherwise do nothing.

This operation leaves exactly those occurrences of b in some infinite b^ω (which witnesses $\bot$) intact, while rewriting finite paths b^* to s^*. We call this operation *b-to-s conversion*.

Thanks to Theorem 4.42, the main result in Broadbent et al. (2010), there is an algorithm that, given $\mathcal{G}_1$, returns a HoRS $\mathcal{G}_2$ over $\Sigma \cup \{b, s\}$ that generates the b-to-s conversion of $[\![\mathcal{G}_1]\!]$. In the language of Broadbent et al. (2010), $\mathcal{G}_2$ is the *φ-reflection* of $\mathcal{G}_1$ where φ (above) is a property definable in the modal μ-calculus.

Stage 3: From $\Sigma \cup \{b, s\}$-labelling $\mathcal{G}_2$ to $\Sigma \cup \{b\}$-labelling $\mathcal{G}_3$. Although the tree $[\![\mathcal{G}_2]\!]$ does not have infinite paths exclusively labelled by s, it may still have nodes labelled by s. We construct a $\Sigma \cup \{b\}$-labelling HoRS $\mathcal{G}_3$ that generates the tree $[\![\mathcal{G}_2]\!]$ but with these remaining s-labelled nodes cut out, which is easily achieved by replacing every occurrence of the terminal symbol s in the rewrite rules of $\mathcal{G}_2$ by the identity nonterminal I. To be precise, if $\mathcal{R}_2$ is the set of rewrite rules of $\mathcal{G}_2$, then the resultant HoRS

$$\mathcal{G}_3 := \langle \mathcal{N} \cup \{I\}, \Sigma \cup \{b\}, \{F = \lambda \overline{x}.t[I/s] \mid F = \lambda \overline{x}.t \in \mathcal{R}_2\} \cup \{I = \lambda x.x\}, S \rangle$$

is the $\bot$-free transform of the input HoRS $\mathcal{G}$.

Example 4.43. Let $\mathcal{G} = \langle \{S, F, G\}, \{\mathsf{cons}, \mathsf{succ}, \mathsf{zero}\}, \mathcal{R}, S \rangle$ be a HoRS with $\mathcal{R}$:

$$\begin{aligned} S &= F\,\mathsf{zero} \\ F &= \lambda x.\,\mathsf{cons}\,(G\,x)\,(F\,(\mathsf{succ}\,x)) \\ G &= \lambda x.\,G\,(\mathsf{succ}\,x) \end{aligned}$$

In stage 1, our $\Sigma_\bot$-labelling HoRS $\mathcal{G}$ is transformed into a $\Sigma \cup \{b\}$-labelling HoRS $\mathcal{G}_1 = \langle \{S, F, G\}, \{\mathsf{cons}, \mathsf{succ}, \mathsf{zero}, b\}, \mathcal{R}_1, S \rangle$ where $\mathcal{R}_1$ contains some bs (in red) to make each rewrite rule productive:

$$\begin{aligned} S &= b\,(F\,\mathsf{zero}) \\ F &= \lambda x.\,\mathsf{cons}\,(G\,x)\,(F\,(\mathsf{succ}\,x)) \\ G &= \lambda x.\,b\,(G\,(\mathsf{succ}\,x)) \end{aligned}$$

Figure 4.2 shows the effect on the tree generated by the HoRS.

Stage 2—where a $\Sigma \cup \{b\}$-labelling $\mathcal{G}_1$ is transformed into a $\Sigma \cup \{b, s\}$-labelling $\mathcal{G}_2$—is generally non-trivial, because some rewrite rules may produce bs in both finite b^* and infinite b^ω, hence we rely on Theorem 4.42 by Broadbent et al. (2010). However, in this example, stage 2 happens to be straightforward; the above situation does not occur. This yields a $\Sigma \cup \{b, s\}$-labelling HoRS $\mathcal{G}_2 = \langle \{S, F, G\}, \{\mathsf{cons}, \mathsf{succ}, \mathsf{zero}, b, s\}, \mathcal{R}_2, S \rangle$ with $\mathcal{R}_2$:

$$\begin{aligned} S &= s\,(F\,\mathsf{zero}) \\ F &= \lambda x.\,\mathsf{cons}\,(G\,x)\,(F\,(\mathsf{succ}\,x)) \\ G &= \lambda x.\,b\,(G\,(\mathsf{succ}\,x)) \end{aligned}$$

Stage 3 finally gives us the $\bot$-free transform of the original HoRS $\mathcal{G}$. The resulting HoRS is $\mathcal{G}_3 = \langle \{S, F, G\}, \{\mathsf{cons}, \mathsf{succ}, \mathsf{zero}, b\}, \mathcal{R}_3, S \rangle$, where $\mathcal{R}_3$ has all occurrences of s replaced by the identity nonterminal I:

$$\begin{aligned} S &= I\,(F\,\mathsf{zero}) \\ F &= \lambda x.\, \mathsf{cons}\,(G\,x)\,(F\,(\mathsf{succ}\,x)) \\ G &= \lambda x.\, b\,(G\,(\mathsf{succ}\,x)) \end{aligned}$$

The following figure shows the conversion on trees corresponding to the three stages of the HoRS transformation.

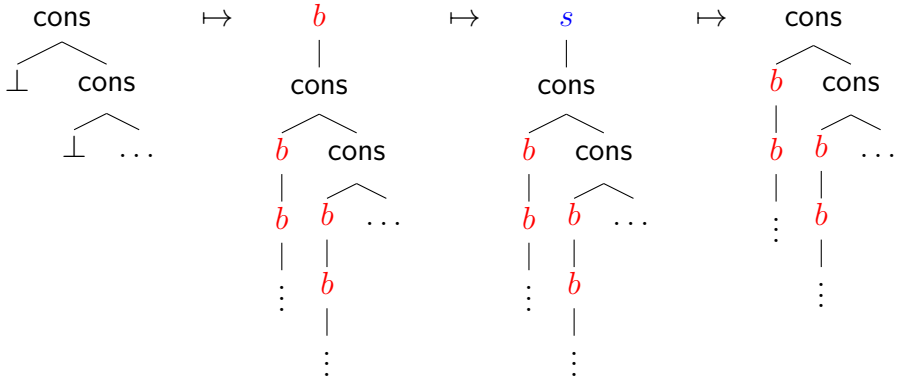


Figure 4.2: Conversion $[\![\mathcal{G}]\!] \mapsto [\![\mathcal{G}_1]\!] \mapsto [\![\mathcal{G}_2]\!] \mapsto [\![\mathcal{G}_3]\!]$

4.6.3 'Decision procedure'

Let $\mathcal{G}_1 = \langle \mathcal{N}_1, \Sigma, \mathcal{R}_1, S_1 \rangle$ and $\mathcal{G}_2 = \langle \mathcal{N}_2, \Sigma, \mathcal{R}_2, S_2 \rangle$ be deterministic HoRS. Assume the trees they generate are $\bot$-free, which is WLOG due to Section 4.6.2. Consider these HoCHC goal formulas:

$$\begin{aligned} Eq_1 &:= \exists r_1\, r_2.\, (R_{S_1}\, r_1 \wedge R_{S_2}\, r_2) \wedge (r_1 = r_2) \\ Eq_0 &:= \exists r_1\, r_2.\, (R_{S_1}\, r_1 \wedge R_{S_2}\, r_2) \wedge (r_1 \neq r_2) \end{aligned}$$

Using the definitions from Section 4.3, we define HoCHC problems

$$\mathcal{P}_i := \langle \Delta_{\mathcal{G}_1} \cup \Delta_{\mathcal{G}_2}, P_{\mathcal{G}_1} \cup P_{\mathcal{G}_2}, Eq_i \rangle,$$

for $i \in \{0, 1\}$, with the Maher theory $T_{\Sigma_\bot}$ as the constraint language and the set $\mathcal{T}_{\Sigma_\bot}$ of finite and infinite trees as the designated model. Note that $r_1 = r_2$ and $r_1 \neq r_2$ in the goal formulas are tree constraints from the Maher theory.

Thanks to Theorem 4.32, we have

- $[\![\mathcal{G}_1]\!] = [\![\mathcal{G}_2]\!]$ iff $\mathcal{P}_1$ is solvable,
- $[\![\mathcal{G}_1]\!] \neq [\![\mathcal{G}_2]\!]$ iff $\mathcal{P}_0$ is solvable.

Recall that the Maher theory $T_{\Sigma_\perp}$ is decidable—to be exact, the question $T_{\Sigma_\perp} \vDash \varphi$ for first-order tree constraints φ like $r_1 = r_2$ and $r_1 \neq r_2$ above. Note, however, that $\mathcal{P}_1$ and $\mathcal{P}_0$ are coinductive HoCHC problems. It is an open question whether coinductive HoCHC problems over a (semi-)decidable background theory—like the Maher theory $T_{\Sigma_\perp}$—can be semi-decided via a reduction to a first-order problem, like inductive HoCHC can (Pham et al., 2018; Ong and Wagner, 2019).

If there exists a such semi-decision procedure for solving coinductive HoCHC over $T_{\Sigma_\perp}$, then we can decide $[\![\mathcal{G}_1]\!] = [\![\mathcal{G}_2]\!]$ by dovetailing our two HoCHC problems to obtain a full decision procedure.

Theorem 4.44. *The HoRS equivalence problem is decidable if the HoRS-to-HoCHC encoding lives in a semi-decidable fragment of coinductive HoCHC over Maher's complete and decidable theory of trees.*

Chapter 5

Solving HoCHC through SLD-resolution

Once we have established the logical foundations of HoCHC, the next step is looking at solution methods. Such methods bridge the gap between theory and practice and allow us to use HoCHC in higher-order program verification.

Resolution proof methods for higher-order logic go back decades in theory (Andrews, 1971; Huet, 1973; Benzmüller and Kohlhase, 1998) and implementation (see e.g. Jensen and Pietrzykowski, 1976). Because the standard semantics of higher-order logic is wildly undecidable (Gödel, 1931), such proof methods are only refutation-complete for the Henkin semantics, where a formula is deemed valid just if it is true in all Henkin frames. Some of these resolution proof methods have been compared by Benzmüller (2002).

The HoCHC fragment of HoL, however, enjoys better algorithmic properties—as long as the background theory is semi-decidable. In this chapter, we provide a sound and refutation-complete resolution proof system that semi-decides HoCHC (unsolvability). We employ the continuous semantics in our proofs, relying on the standard-continuous equivalence from Chapter 3.4.3 to generalise our result to the standard semantics.

We adapt an approach called *Selective Linear Definite clause resolution*. This *SLD-resolution* is a refinement of resolution that is sound and refutation-complete for first-order Horn clauses (Kowalski, 1974). It relies on an SLD-inference rule:

$$\frac{G = L_1 \wedge \cdots \wedge L_n \qquad L \leftarrow K_1 \wedge \cdots \wedge K_m}{G' = (L_1 \wedge \cdots \wedge L_{i-1} \wedge K_1 \wedge \cdots \wedge K_m \wedge L_{i+1} \wedge \ldots L_n)\theta} \; \theta \text{ unifies } L, L_i$$

That is, given a goal clause G we derive another goal clause G' using a substitution θ that effectively substitutes the RHS of a definite clause for a fully applied occurrence of a relational variable in G.

SLD-resolution is *selective* in that at most one literal is chosen to be 'unfolded' in each step. It is *linear* in the sense that SLD-proofs have a linear shape rather than a more complicated tree structure. SLD-resolution is the main computation procedure used in Prolog.

Resolution proof systems create less overhead than Reynolds-style defunctionalisation algorithms like the one employed by Pham et al. (2018) for solving HoCHC. As a consequence, our SLD-resolution is more user friendly and easier to understand than Pham et al. (2018)'s work. Implementation, however, might be harder, because it involves a higher degree of nondeterministic choice.

Charalambidis et al. (2013) have developed SLD-resolution for a fragment of higher-order logic that is closely related to HoCHC, namely a positive existential fragment that corresponds to higher-order Horn clauses without constraints. SLD-resolution is sound and refutation-complete for this fragment. Charalambidis et al. (2013) also created a prototype implementation of their procedure in Haskell[1].

How can SLD-resolution help us solve the HoCHC problem? Given a HoCHC problem $\langle \Delta, P, G \rangle$, if SLD-resolution derives a satisfiable background formula from G, then the least model of P satisfies G, and $\langle \Delta, P, G \rangle$ is unsolvable. Furthermore, if satisfiability of background formulas is semi-decidable, then we can use SLD-resolution to semi-decide unsolvable HoCHC instances over this background theory. In this chapter, we work towards the following theorem.

Theorem 5.22 (Reduction)**.** *A HoCHC problem $\langle \Delta_{rel}, P, G \rangle$ is unsolvable if and only if there exists an SLD-refutation $G \twoheadrightarrow \varphi$ for some constraint formula φ.*

SLD-resolution is recursively enumerable; there are countably many possible (finite) SLD-refutations that we can iterate over. As such, SLD-resolution provides us with a semi-decision procedure for HoCHC that is guaranteed to terminate for unsolvable instances.

[1] http://code.haskell.org/hopes

Theorem 5.24 (Termination)**.** *If a HoCHC problem $\langle \Delta_{rel}, P, G \rangle$ is unsolvable over a semi-decidable background theory, then an SLD-refutation of $\langle \Delta_{rel}, P, G \rangle$ can be found in finite time.*

Note that Charalambidis et al. (2013) come from a logic programming perspective, whereas ours is purely logical. Their work on SLD-resolution for higher-order Horn clauses without constraints is motivated by finding solution sets. HoCHC, on the other hand, concerns safety problems, where we are interested in the existence of a violation. Furthermore, our goal clauses are closed (over the relational variables).

Our lack of interest in full solution sets gives us an advantage over Charalambidis et al. (2013), because we can eliminate higher-order (existential) quantification from goal clauses if we use the monotone or continuous semantics. We substitute the greatest relation of sort ρ for all occurrences of an existentially quantified variable of sort ρ, following Cathcart Burn et al. (2018).

Let us define the universal relation of sort ρ by $U_\rho : \rho := \lambda \overline{x}.\, \mathsf{true}$. Note that U_ρ is a well-formed goal term for all relational sorts ρ. Because of monotonicity of our semantics, it holds that:

$$\mathcal{C}[\![\exists x : \rho.\, G : o]\!] = \mathcal{C}[\![G[U_\rho / x]]\!]$$

Thus, a syntactic substitution allows us to eliminate higher-order existentials from goal clauses and logic programs.

This means that only individually sorted variables occur under quantifiers, and SLD-refutations for HoCHC never have to 'guess' higher-order substitutions. Even with this simplification, we can still find witnesses to the unsolvability of a HoCHC problem. We do not lose all witnesses, merely the full solution set.

We can simplify even further. Because a first-order background theory is incorporated in HoCHC as constraints, we resolve a goal clause $G : o$ to some satisfiable constraint formula φ instead of all the way to true. This means that we do not have to assign meaning to free variables during an SLD-derivation. In fact, this allows us to get rid of substitutions altogether.

Eliminating substitution increases the power of our proof system, because the use of substitutions requires *definability* of the set of individuals to achieve completeness. That is, every element of the semantic domain must be captured by the meaning of some finite term definable in the logic. Without substitutions, we do not need

definability. Maher's theory of trees (Maher, 1988) is an example of a background theory that is decidable in which not all individuals are definable in this sense (infinite trees are generally not). Yet without substitutions, we can solve HoCHC instances over the Maher theory.

Before we present our SLD proof system in Section 5.2, we introduce some HoCHC notation in Section 5.1. Correctness of the system is outlined in the subsequent three sections: soundness (Section 5.3), refutation-completeness (Section 5.4), and termination (Section 5.5). We provide some example derivations in Section 5.6. Finally, Section 5.7 outlines the additional steps we would need to take—and sacrifices to make—if we were interested in full solution sets.

Note that since the author started this work on SLD-resolution for HoCHC, another resolution proof system has been published by Ong and Wagner (2019). Our approach is more algorithmic than their strictly logical approach. However, the approaches are morally identical even if they differ in the details. One such detail is that we use a continuous interpretation of HoCHC, while Ong and Wagner (2019) work with the standard interpretation directly.

5.1 Input HoCHC problem

The work in this chapter does not rely on the choice of background theory, as long as the theory is semi-decidable, unlike in Chapter 4 where we fix a specific background theory. Without semi-decidability, only soundness can be attained.

The HoCHC fragment of higher-order logic is larger than the fragment studied by Charalambidis et al. (2013), which coincides with higher-order Horn clauses (i.e. without constraints). To see that this is the case, observe that we allow $\forall$, $\Rightarrow$, and $\neg$ in constraint formulas—provided these are part of the constraint language—while Charalambidis et al. (2013) do not allow these operators at all.

Because of this, we need to assume that our background theory Th is semi-decidable if we are to obtain a refutation-complete method for semi-deciding unsolvable HoCHC instances via SLD-resolution. This assumption is not required for soundness.

Our instance $\langle \Delta_{rel}, P, G \rangle$ of the (continuous) HoCHC problem has the following form:

$$\begin{aligned} \Delta_{rel} &:= \{x_1 : \rho_1, \quad \ldots, \quad x_k : \rho_k\} \\ P &:= \{x_1 : \rho_1 = G_1, \quad \ldots, \quad x_k : \rho_k = G_k\} \end{aligned}$$

and a goal term $\Delta_{rel} \vdash G : o$, for some goal terms $\Delta_{rel} \vdash G_i : \rho_i$, for all $1 \leq i \leq k$.

Notation 5.1. A sort environment Δ can be partitioned into Δ_{rel} and Δ_{var} that denote the relational variables and the remaining free variables, resp. Similarly, a valuation $\alpha \in \mathcal{C}[\![\Delta]\!]$ can be partitioned into $\alpha_{rel} \in \mathcal{C}[\![\Delta_{rel}]\!]$ and $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. We write $\alpha_{rel} \cup \alpha_{var}$ for α and $\Delta_{rel}, \Delta_{var}$ for Δ.

5.2 Proof system

Charalambidis et al. (2013)'s proof system consists of over a dozen proof rules. Many of these rules are redundant for us due to our deferring to the semi-decidable background theory, rather than resolving down to true. Furthermore, most rules simply distribute and propagate without doing any heavy lifting.

We use the following definitions, where the first two rules roughly correspond to the SLD-inference rule and the remaining rules are syntax-directed 'bookkeeping'.

Definition 5.2 (Proof rules)**.** *Let $\vdash P : \Delta_{rel}$ be a program. We say that $\Delta' \vdash G' : o$ is derived from $\Delta \vdash G : o$ in one step—and denote this by $G \to G'$—if G and G' match one of the following:*

(1) $y\, E_1 \dots E_n \to P(y)\, E_1 \dots E_n$ *for* $y \in \Delta_{rel}$

(2) $(\lambda x_1 \dots x_n.\, G)\, E_1 \dots E_n \to G[E_1/x_1, \dots, E_n/x_n]$

(3) $G_1 \vee G_2 \to G_1$

(4) $G_1 \vee G_2 \to G_2$

(5) $\exists x.\, G \to G$

(6) $G_1 \wedge G_2 \to G_1' \wedge G_2$ *if* $G_1 \to G_1'$

(7) $G_1 \wedge G_2 \to G_1 \wedge G_2'$ *if* $G_2 \to G_2'$

We assume that the LHS is not a constraint formula. Furthermore, we write $G \twoheadrightarrow G'$ if there exist $G_0, G_1, \dots, G_n$ such that $G = G_0 \to G_1 \to \cdots \to G_n = G'$, for some $n \geq 0$.

Note that $G \to G'$ means that $\Delta \vdash G : o$ and $\Delta' \vdash G' : o$ such that Δ' is either Δ or $\Delta, x : \iota$ for some variable $x : \iota$.

Intuitively, we unfold rules from the logic program until we derive a formula from the background theory that can be decided.

Definition 5.3 (SLD-derivation)**.** *An* SLD-derivation *of* $\langle \Delta, P, G \rangle$ *is either*

(i) a finite sequence $G = G_0 \to G_1 \to \cdots \to G_n$ *of goal terms, or*

(ii) an infinite sequence $G = G_0 \to G_1 \to G_2 \to \ldots$ *of goal terms.*

Definition 5.4 (SLD-refutation)**.** *Assume that* $\langle \Delta, P, G \rangle$ *has a finite SLD-derivation* $G = G_0 \twoheadrightarrow G_n = \varphi$ *for some satisfiable constraint formula* φ*. Then, we say that* $\langle \Delta, P, G \rangle$ *has an* SLD-refutation *(of length* n*).*

SLD-resolution typically involves substitutions and a form of higher-order unification (Dowek, 2001). Unlike in first-order unification, most general unifiers do not always exist in the higher-order setting, although higher-order matching—where one of the two terms is closed—enjoys better algorithmic properties and is, in fact, decidable (Stirling, 2009).

We have eliminated higher-order existentials, though, so our only 'unification' is instantiating bound variables.

5.3 Soundness

Lemma 5.5 up to Lemma 5.7 show that the semantics is invariant over β-reduction. We rely on this in the soundness proofs, culminating in Theorem 5.10.

Lemma 5.5 (Substitution Lemma, sort ι)**.** *Let* T *be a term sorted over* ι*. Let* $\alpha = (\alpha_{rel} \cup \alpha_{var}) \in \mathcal{C}[\![\Delta]\!]$ *be a valuation, and* θ *a substitution of some variables in* Δ_{var}*. Then,*

$$\mathcal{S}[\![\Delta \vdash \theta(T)]\!](\alpha) = \mathcal{S}[\![\Delta \vdash T]\!](\alpha_{rel} \cup \alpha'_{var}),$$

where

$$\alpha'_{var}(x) = \begin{cases} \mathcal{S}[\![\Delta \vdash \theta(x)]\!](\alpha) & \text{if } x \in \mathsf{dom}(\theta) \text{ and } x : \iota \\ \mathcal{C}[\![\Delta \vdash \theta(x)]\!](\alpha) & \text{if } x \in \mathsf{dom}(\theta) \text{ and not } x : \iota \\ \alpha_{var}(x) & \text{otherwise} \end{cases}$$

Proof. By induction on the structure of $T : \sigma$, where σ is a sort over ι.

If $T = f \in \Sigma$, then $\theta(f) = f$ and trivially $\mathcal{S}[\![\Delta \vdash \theta(f)]\!](\alpha) = \mathcal{S}[\![\Delta \vdash f]\!](\alpha_{rel} \cup \alpha'_{var})$.

If $T = x \in \Delta_{var}$ and $x \in \mathsf{dom}(\theta)$, then $x : \iota$ and

$$\mathcal{S}[\![\Delta \vdash \theta(x)]\!](\alpha) = \alpha'_{var}(x) = \mathcal{S}[\![\Delta \vdash x]\!](\alpha_{rel} \cup \alpha'_{var}).$$

If $T = x \in \Delta_{var}$ and $x \notin \mathsf{dom}(\theta)$, then $\theta(x) = x$ and

$$\begin{aligned}\mathcal{S}[\![\Delta \vdash \theta(x)]\!](\alpha) &= \mathcal{S}[\![\Delta \vdash x]\!](\alpha) \\ &= \alpha_{var}(x) \\ &= \alpha'_{var}(x) \\ &= \mathcal{S}[\![\Delta \vdash x]\!](\alpha_{rel} \cup \alpha'_{var}).\end{aligned}$$

Suppose that the claim holds for all for all T' smaller than T.

$$\begin{aligned}\mathcal{S}[\![\Delta \vdash \theta(T_1\, T_2)]\!](\alpha) &= \mathcal{S}[\![\Delta \vdash \theta(T_1)]\!](\alpha)\,(\mathcal{S}[\![\Delta \vdash \theta(T_2)]\!](\alpha)) \\ &= \mathcal{S}[\![\Delta \vdash T_1]\!](\alpha_{rel} \cup \alpha'_{var})\,(\mathcal{S}[\![\Delta \vdash T_2]\!](\alpha_{rel} \cup \alpha'_{var})) \quad \text{IH} \\ &= \mathcal{S}[\![\Delta \vdash T_1\, T_2]\!](\alpha_{rel} \cup \alpha'_{var})\end{aligned}$$

$$\begin{aligned}\mathcal{S}[\![\Delta \vdash \theta(\lambda x : \sigma.\, T')]\!](\alpha) &= \mathcal{S}[\![\Delta \vdash \lambda x : \sigma.\, \theta(T')]\!](\alpha) \\ &= \lambda v.\, \mathcal{S}[\![\Delta, x : \sigma \vdash \theta(T')]\!](\alpha[x \mapsto v]) \\ &= \lambda v.\, \mathcal{S}[\![\Delta, x : \sigma \vdash T']\!]((\alpha_{rel} \cup \alpha'_{var})[x \mapsto v]) \quad \text{IH} \\ &= \mathcal{S}[\![\Delta \vdash \lambda x : \sigma.\, T']\!](\alpha_{rel} \cup \alpha'_{var})\end{aligned}$$

□

Lemma 5.6 (Substitution Lemma, sort ρ)**.** *Let $\Delta \vdash G$ be a goal term. Let $\alpha = (\alpha_{rel} \cup \alpha_{var}) \in \mathcal{C}[\![\Delta]\!]$ be a valuation, and θ a substitution of some variables in Δ_{var}. Then,*

$$\mathcal{C}[\![\Delta \vdash \theta(G)]\!](\alpha) = \mathcal{C}[\![\Delta \vdash G]\!](\alpha_{rel} \cup \alpha'_{var}),$$

where

$$\alpha'_{var}(x) = \begin{cases} \mathcal{S}[\![\Delta \vdash \theta(x)]\!](\alpha) & \text{if } x \in \mathsf{dom}(\theta) \text{ and } x : \iota \\ \mathcal{C}[\![\Delta \vdash \theta(x)]\!](\alpha) & \text{if } x \in \mathsf{dom}(\theta) \text{ and not } x : \iota \\ \alpha_{var}(x) & \text{otherwise} \end{cases}$$

Proof. By induction on the structure of G.

Let $G = x : \rho$. If $x : \rho \in \Delta_{rel}$ then

$$\mathcal{C}[\![\Delta \vdash \theta(x)]\!](\alpha) = \mathcal{C}[\![\Delta \vdash x]\!](\alpha) = \mathcal{C}[\![\Delta \vdash x]\!](\alpha_{rel} \cup \alpha'_{var}).$$

Otherwise, $x : \rho \notin \Delta_{rel}$, and either $x \in \mathsf{dom}(\theta)$ or $x \notin \mathsf{dom}(\theta)$. In the former case:

$$\mathcal{C}[\![\Delta \vdash \theta(x)]\!](\alpha) = \alpha'_{var}(x) = \mathcal{C}[\![\Delta \vdash x]\!](\alpha_{rel} \cup \alpha'_{var})$$

In the latter case:

$$\mathcal{C}[\![\Delta \vdash \theta(x)]\!](\alpha) = \mathcal{C}[\![\Delta \vdash x]\!](\alpha) = \alpha_{var}(x) = \alpha'_{var}(x) = \mathcal{C}[\![\Delta \vdash x]\!](\alpha_{rel} \cup \alpha'_{var})$$

Let $G = \varphi : o$. We proceed by induction on the structure of $\varphi : o$. The claim trivially holds for true and false. If $G = P\,T_1 \ldots T_n$ for some (first-order) predicate P, then we use Lemma 5.5 to obtain:

$$\begin{aligned}
&\mathcal{C}[\![\Delta \vdash \theta(P\,T_1 \ldots T_n)]\!](\alpha) \\
&= \mathcal{S}[\![\Delta \vdash \theta(P\,T_1 \ldots T_n)]\!](\alpha) \\
&= \mathcal{S}[\![\Delta \vdash \theta(P)]\!](\alpha)\,(\mathcal{S}[\![\Delta \vdash \theta(T_1)]\!](\alpha)) \ldots (\mathcal{S}[\![\Delta \vdash \theta(T_n)]\!](\alpha)) \\
&= \mathcal{S}[\![\Delta \vdash P]\!](\alpha_{rel} \cup \alpha'_{var})\,(\mathcal{S}[\![\Delta \vdash \theta(T_1)]\!](\alpha)) \ldots (\mathcal{S}[\![\Delta \vdash \theta(T_n)]\!](\alpha)) \\
&= \mathcal{S}[\![\Delta \vdash P]\!](\alpha_{rel} \cup \alpha'_{var})\,(\mathcal{S}[\![\Delta \vdash T_1]\!](\alpha_{rel} \cup \alpha'_{var})) \ldots (\mathcal{S}[\![\Delta \vdash T_n]\!](\alpha_{rel} \cup \alpha'_{var})) \\
&= \mathcal{S}[\![\Delta \vdash P\,T_1 \ldots T_n]\!](\alpha_{rel} \cup \alpha'_{var}) \\
&= \mathcal{C}[\![\Delta \vdash P\,T_1 \ldots T_n]\!](\alpha_{rel} \cup \alpha'_{var})
\end{aligned}$$

The cases where G is a logical operator (e.g. conjunction, disjunction, implication, negation) or quantifier follow from the fact these are constants. This establishes the case where $G = \varphi : o$. Now suppose the claim holds for all goal terms smaller than G.

Let $G = G'\,H$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash \theta(G'\,H)]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash \theta(G')\,\theta(H)]\!](\alpha) \\
&= \mathcal{C}[\![\Delta \vdash \theta(G')]\!](\alpha)(\mathcal{C}[\![\Delta \vdash \theta(H)]\!](\alpha)) \\
&= \mathcal{C}[\![\Delta \vdash G']\!](\alpha_{rel} \cup \alpha'_{var})(\mathcal{C}[\![\Delta \vdash H]\!](\alpha_{rel} \cup \alpha'_{var})) \quad \text{IH} \\
&= \mathcal{C}[\![\Delta \vdash G'\,H]\!](\alpha_{rel} \cup \alpha'_{var})
\end{aligned}$$

Let $G = H\,N$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash \theta(H\,N)]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash \theta(H)\,\theta(N)]\!](\alpha) \\
&= \mathcal{C}[\![\Delta \vdash \theta(H)]\!](\alpha)(\mathcal{S}[\![\Delta \vdash \theta(N)]\!](\alpha)) \\
&= \mathcal{C}[\![\Delta \vdash H]\!](\alpha_{rel} \cup \alpha'_{var})(\mathcal{S}[\![\Delta \vdash N]\!](\alpha_{rel} \cup \alpha'_{var})) \quad \text{IH, Lem 5.5} \\
&= \mathcal{C}[\![\Delta \vdash H\,N]\!](\alpha_{rel} \cup \alpha'_{var})
\end{aligned}$$

Let $G = \lambda x.\, H$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash \theta(\lambda x.\, H)]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash \lambda x.\, \theta(H)]\!](\alpha) \\
&= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash \theta(H)]\!](\alpha[x \mapsto v]) \\
&= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H]\!]((\alpha_{rel} \cup \alpha'_{var})[x \mapsto v]) \quad \text{IH} \\
&= \mathcal{C}[\![\Delta \vdash \lambda x.\, H]\!](\alpha_{rel} \cup \alpha'_{var})
\end{aligned}$$

Finally, the equality holds trivially if G is a constant $\wedge$, $\vee$, or $\exists_\sigma$. □

Lemma 5.7. *Let $\Delta \vdash (\lambda x : \sigma.\, G)\, E$ be a goal term. For all $\alpha \in \mathcal{C}[\![\Delta]\!]$,*

$$\mathcal{C}[\![\Delta \vdash (\lambda x : \sigma.\, G)\, E]\!](\alpha) = \mathcal{C}[\![\Delta \vdash G[E/x]]\!](\alpha).$$

Proof. We need to distinguish the cases where x and E have sort ι and where they have relational sort ρ.

First, if $x : \iota$, then $\Delta, x : \iota \vdash G$. Let $\alpha \in \mathcal{C}[\![\Delta]\!]$, and let $\beta, \beta' \in \mathcal{C}[\![\Delta, x : \iota]\!]$ such that

$$\beta(y) = \begin{cases} w & \text{if } x = y \\ \alpha(y) & \text{if } x \neq y \end{cases} \qquad \beta'(y) = \begin{cases} \mathcal{S}[\![\Delta, x : \iota \vdash E]\!](\beta) & \text{if } x = y \\ \alpha(y) & \text{if } x \neq y \end{cases}$$

for some arbitrary $w \in \mathcal{C}[\![\iota]\!]$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash (\lambda x : \iota.\, G)\, E]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash \lambda x : \iota.\, G]\!](\alpha)(\mathcal{S}[\![\Delta \vdash E : \iota]\!](\alpha)) \\
&= \mathcal{C}[\![\Delta, x : \iota \vdash G]\!](\alpha[x \mapsto \mathcal{S}[\![\Delta \vdash E]\!](\alpha)]) \\
&= \mathcal{C}[\![\Delta, x : \iota \vdash G]\!](\alpha[x \mapsto \mathcal{S}[\![\Delta, x : \iota \vdash E]\!](\beta)]) \\
&= \mathcal{C}[\![\Delta, x : \iota \vdash G]\!](\beta') \\
&= \mathcal{C}[\![\Delta, x : \iota \vdash G[E/x]]\!](\beta) \qquad \text{Lem 5.6} \\
&= \mathcal{C}[\![\Delta \vdash G[E/x]]\!](\alpha)
\end{aligned}$$

The second case, where $x : \rho$ and $\Delta, x : \rho \vdash G$, is very similar. Let $\alpha \in \mathcal{C}[\![\Delta]\!]$, and let $\beta, \beta' \in \mathcal{C}[\![\Delta, x : \rho]\!]$ such that

$$\beta(y) = \begin{cases} w & \text{if } x = y \\ \alpha(y) & \text{if } x \neq y \end{cases} \qquad \beta'(y) = \begin{cases} \mathcal{C}[\![\Delta, x : \rho \vdash E]\!](\beta) & \text{if } x = y \\ \beta(y) & \text{if } x \neq y \end{cases}$$

for some arbitrary $w \in \mathcal{C}[\![\rho]\!]$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash (\lambda x : \rho.\, G)\, E]\!](\alpha) &= \mathcal{C}[\![\Delta \vdash \lambda x : \rho.\, G]\!](\alpha)(\mathcal{C}[\![\Delta \vdash E : \rho]\!](\alpha)) \\
&= \mathcal{C}[\![\Delta, x : \rho \vdash G]\!](\alpha[x \mapsto \mathcal{C}[\![\Delta \vdash E : \rho]\!](\alpha)]) \\
&= \mathcal{C}[\![\Delta, x : \rho \vdash G]\!](\alpha[x \mapsto \mathcal{C}[\![\Delta, x : \rho \vdash E]\!](\beta)]) \\
&= \mathcal{C}[\![\Delta, x : \rho \vdash G]\!](\beta') \\
&= \mathcal{C}[\![\Delta, x : \rho \vdash G[E/x]]\!](\beta) \qquad \text{Lem 5.6} \\
&= \mathcal{C}[\![\Delta \vdash G[E/x]]\!](\alpha)
\end{aligned}$$

□

Lemma 5.8. *Let $\vdash P : \Delta_{rel}$ be a program, let $\Delta \vdash G : o$ and $\Delta' \vdash G' : o$ be goal terms such that $G \to G'$. Then, for every model $M \in \mathcal{C}[\![\Delta_{rel}]\!]$ of P, every $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$, and every $\alpha'_{var} \in \mathcal{C}[\![\Delta'_{var}]\!]$ such that $\alpha_{var} \subseteq \alpha'_{var}$, it holds that:*

$$\mathcal{C}[\![\Delta \vdash G]\!](M \cup \alpha_{var}) \sqsupseteq \mathcal{C}[\![\Delta' \vdash G']\!](M \cup \alpha'_{var})$$

Proof. Note that Δ' is either Δ or $\Delta, x : \iota$ for some variable $x : \iota$. Let us write α for $M \cup \alpha_{var}$ and α' for $M \cup \alpha'_{var}$.

Let $G = G_1 \wedge \cdots \wedge G_m$. We proceed by induction on m.

Base case $m = 1$**.** By case analysis on G.

- Let $G = y\, E_1 \ldots E_n$ with $y \in \Delta_{rel}$. The inclusion $\mathcal{C}[\![\Delta \vdash y\, E_1 \ldots E_n]\!](\alpha) \sqsupseteq \mathcal{C}[\![\Delta \vdash P(y)\, E_1 \ldots E_n]\!](\alpha)$ follows from the fact that M in $\alpha = M \cup \alpha_{var}$ is a model of P.
- Let $G = (\lambda x_1 \ldots x_n.\, H)\, E_1 \ldots E_n$. By repeated application of Lemma 5.7.
- Let $G = G_1 \vee G_2$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash G_1 \vee G_2]\!](\alpha) &= \max\{\mathcal{C}[\![\Delta \vdash G_1]\!](\alpha), \mathcal{C}[\![\Delta \vdash G_2]\!](\alpha)\} \\
&\sqsupseteq \mathcal{C}[\![\Delta \vdash G_i]\!](\alpha) \qquad \text{for all } i \in \{1, 2\}
\end{aligned}$$

- Let $G = \exists x : \iota.\, G$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash \exists x : \iota.\, G]\!](\alpha) &= \max\{\mathcal{C}[\![\Delta \vdash \lambda x : \iota.\, G]\!](\alpha)(r) \mid r \in \mathcal{C}[\![\iota]\!]\} \\
&= \max\{\mathcal{C}[\![\Delta, x : \iota \vdash G]\!](\alpha[x \mapsto r]) \mid r \in \mathcal{C}[\![\iota]\!]\} \\
&\sqsupseteq \mathcal{C}[\![\Delta, x : \iota \vdash G]\!](\alpha[x \mapsto r]) \qquad \text{for all } r \in \mathcal{C}[\![\iota]\!]
\end{aligned}$$

Inductive case $m > 1$. Let $G = G_1 \wedge G_2$. Suppose WLOG $G_1 \to G_1'$.

$$\begin{aligned}
\mathcal{C}[\![\Delta \vdash G_1 \wedge G_2]\!](\alpha) &= \min\{\mathcal{C}[\![\Delta \vdash G_1]\!](\alpha), \mathcal{C}[\![\Delta \vdash G_2]\!](\alpha)\} \\
&\sqsupseteq \min\{\mathcal{C}[\![\Delta' \vdash G_1']\!](\alpha'), \mathcal{C}[\![\Delta \vdash G_2]\!](\alpha)\} && \text{IH} \\
&= \min\{\mathcal{C}[\![\Delta' \vdash G_1']\!](\alpha'), \mathcal{C}[\![\Delta' \vdash G_2]\!](\alpha')\} \\
&= \mathcal{C}[\![\Delta' \vdash G_1' \wedge G_2]\!](\alpha')
\end{aligned}$$

□

Lemma 5.9. *Let $\vdash P : \Delta_{rel}$ be a program and $\Delta \vdash G : o$ a goal term. Let $G \twoheadrightarrow \varphi$ be an SLD-refutation. Then, for every model $M \in \mathcal{C}[\![\Delta_{rel}]\!]$ of P, every valuation $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$, and every valuation $\alpha'_{var} \in \mathcal{C}[\![\Delta'_{var}]\!]$ such that $\Delta_{rel}, \Delta'_{var} \vdash \varphi$ and $\alpha_{var} \subseteq \alpha'_{var}$:*

$$\mathcal{C}[\![\Delta \vdash G]\!](M \cup \alpha_{var}) \sqsupseteq \mathcal{C}[\![\Delta' \vdash \varphi]\!](M \cup \alpha'_{var})$$

Proof. Using Lemma 5.8 and induction on length n of the (finite) SLD-refutation. □

Theorem 5.10 (Soundness)**.** *If $G \twoheadrightarrow \varphi$ for a satisfiable background formula φ, then the HoCHC problem $\langle \Delta_{rel}, P, G\rangle$ is unsolvable.*

Proof. A direct consequence of Lemma 5.9. □

5.4 Completeness

Definition 5.11. *Let $\vdash P : \Delta_{rel}$ be a program and let $\Delta \vdash G : o$ be a goal term. We define S_G to be the set of goal terms that can be obtained from G by replacing zero or more occurrences of relational variables $y \in \Delta_{rel}$ by $P(y)$. Additionally, define $\widehat{G}$ to be the goal obtained by replacing every such occurrence.*

Lemma 5.12. *Let $\vdash P : \Delta_{rel}$ be a program, $\Delta \vdash G$ a goal term, and $\alpha_{rel} \in \mathcal{C}[\![\Delta_{rel}]\!]$ and $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. Then, $\mathcal{C}[\![\Delta \vdash G]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var}) = \mathcal{C}[\![\Delta \vdash \widehat{G}]\!](\alpha_{rel} \cup \alpha_{var})$.*

Proof. By structural induction on G. We write α for $\alpha_{rel} \cup \alpha_{var}$. The cases for logical constants are independent of valuation and, therefore, trivial. The remaining two

base cases are as follows:

$$\begin{aligned}\mathcal{C}[\![\Delta \vdash x : \rho]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var}) &= T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel})(x)\\ &= \mathcal{C}[\![\Delta_{rel} \vdash P(x)]\!](\alpha_{rel})\\ &= \mathcal{C}[\![\Delta_{rel} \vdash \widehat{x}]\!](\alpha_{rel})\\ &= \mathcal{C}[\![\Delta \vdash \widehat{x}]\!](\alpha)\end{aligned}$$

$$\begin{aligned}\mathcal{C}[\![\Delta \vdash \varphi : o]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var}) &= \mathcal{S}[\![\Delta \vdash \varphi : o]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})\\ &= \mathcal{S}[\![\Delta \vdash \varphi : o]\!](\alpha)\\ &= \mathcal{C}[\![\Delta \vdash \varphi : o]\!](\alpha)\\ &= \mathcal{C}[\![\Delta \vdash \widehat{\varphi}]\!](\alpha)\end{aligned}$$

There are three inductive cases.

$$\begin{aligned}&\mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, H]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})\\ &= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H]\!]((T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})[x \mapsto v])\\ &= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash H]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var}[x \mapsto v])\\ &= \lambda v \in \mathcal{C}[\![\sigma]\!].\, \mathcal{C}[\![\Delta, x : \sigma \vdash \widehat{H}]\!](\alpha[x \mapsto v]) \qquad \text{IH}\\ &= \mathcal{C}[\![\Delta \vdash \lambda x : \sigma.\, \widehat{H}]\!](\alpha)\\ &= \mathcal{C}[\![\Delta \vdash \widehat{\lambda x : \sigma.\, H}]\!](\alpha)\end{aligned}$$

$$\begin{aligned}&\mathcal{C}[\![\Delta \vdash H\ N]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})\\ &= \mathcal{C}[\![\Delta \vdash H]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})(\mathcal{S}[\![\Delta \vdash N]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var}))\\ &= \mathcal{C}[\![\Delta \vdash H]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})(\mathcal{S}[\![\Delta \vdash N]\!](\alpha))\\ &= \mathcal{C}[\![\Delta \vdash \widehat{H}]\!](\alpha)(\mathcal{S}[\![\Delta \vdash N]\!](\alpha)) \qquad \text{IH}\\ &= \mathcal{C}[\![\Delta \vdash \widehat{H}\ N]\!](\alpha)\\ &= \mathcal{C}[\![\Delta \vdash \widehat{H\ N}]\!](\alpha)\end{aligned}$$

$$\begin{aligned}&\mathcal{C}[\![\Delta \vdash H_1\ H_2]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})\\ &= \mathcal{C}[\![\Delta \vdash H_1]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var})(\mathcal{C}[\![\Delta \vdash H_2]\!](T^{\mathcal{C}}_{P:\Delta}(\alpha_{rel}) \cup \alpha_{var}))\\ &= \mathcal{C}[\![\Delta \vdash \widehat{H_1}]\!](\alpha)(\mathcal{C}[\![\Delta \vdash \widehat{H_2}]\!](\alpha)) \qquad \text{IH}\\ &= \mathcal{C}[\![\Delta \vdash \widehat{H_1}\ \widehat{H_2}]\!](\alpha)\\ &= \mathcal{C}[\![\Delta \vdash \widehat{H_1\ H_2}]\!](\alpha)\end{aligned}$$

□

Lemma 5.13. *Let $\vdash P : \Delta_{rel}$ be a program, and $\Delta \vdash G : o$ and $\Delta \vdash G' : o$ goal terms such that $G' \in S_G$. If $G' \to H'$, then $G \twoheadrightarrow H$ where $H' \in S_H$.*

Proof. Let $G = G_1 \wedge \cdots \wedge G_m$. We proceed by induction on m.

Base case $m = 1$**.** By case analysis on G.

- Let $G = y\, E_1 \ldots E_m$ with $y \in \Delta_{rel}$, so $G' = y\, E'_1 \ldots E'_m$ or $G' = P(y)\, E'_1 \ldots E'_m$, where either $E'_i = E_i$ (if $E_i : \iota$) or $E'_i \in S_{E_i}$, for all $i \in [m]$.

  Suppose the former. Then, $H' = P(y)\, E'_1 \ldots E'_m$.

  $$G = y\, E_1 \ldots E_m \to P(y)\, E_1 \ldots E_m = H$$

  Suppose the latter. If $y : o$, then $G = y \to P(y) \to H' = H$. Otherwise, $P(y)$ is of the form $\lambda \overline{x}.\, J$, so that $H' = J[E'_1/x_1, \ldots E'_m/x_m]$.

  $$G = y\, E_1 \ldots E_m \to P(y)\, E_1 \ldots E_m \to J[E_1/x_1, \ldots E_m/x_m] = H$$

- Let $G = (\lambda x_1 \ldots x_m.\, G_1)\, E_1 \ldots E_m$. Then, $G' = (\lambda x_1 \ldots x_m.\, G'_1)\, E'_1 \ldots E'_m$ for some $G'_1 \in S_{G_1}$ and, for all $i \in [m]$, either $E'_i = E_i$ (if $E_i : \iota$) or $E'_i \in S_{E_i}$. Furthermore, $H' = G'_1[E'_1/x_1, \ldots, E'_m/x_m]$.

  $$G = (\lambda x_1 \ldots x_m.\, G_1)\, E_1 \ldots E_m \to G_1[E_1/x_1, \ldots, E_m/x_m] = H$$

- Let $G = G_1 \vee G_2$. Then, $G' = G'_1 \vee G'_2$ for some $G'_1 \in S_{G_1}$ and $G'_2 \in S_{G_2}$, and $H' = G'_i$ for some $i \in \{1, 2\}$.

  $$G = G_1 \vee G_2 \to G_i = H$$

- Let $G = \exists x.\, G_1$. Then, $G' = \exists x.\, G'_1$ for some $G'_1 \in S_{G_1}$, and $H' = G'_1$.

  $$G = \exists x.\, G_1 \to G_1 = H$$

Inductive case $m > 1$**.** Let $G = G_1 \wedge G_2$. Then, $G' = G'_1 \wedge G'_2$ where $G'_1 \in S_{G_1}$ and $G'_2 \in S_{G_2}$. Assume WLOG that $G'_1 \to H'_1$, so that $H' = H'_1 \wedge G'_2$. By the IH, $G'_1 \to H'_1$ gives us $G_1 \twoheadrightarrow H_1$ such that $H'_1 \in S_{H_1}$.

$$G = G_1 \wedge G_2 \twoheadrightarrow H_1 \wedge G_2 = H$$

□

Lemma 5.14. *Let $\vdash P : \Delta_{rel}$ be a program, and $\Delta \vdash G : o$ and $\Delta \vdash \varphi' : o$ goal terms such that $\varphi' \in S_G$. If φ' is satisfiable, then there exists an SLD-refutation $G \twoheadrightarrow \varphi$.*

Proof. Let $G = G_1 \wedge \cdots \wedge G_m$. We proceed by induction on m. Suppose that $\varphi' \in S_G$ is satisfiable. Note that if G is a background formula then $G = \varphi' = \varphi$.

Base case $m = 1$**.** By case analysis on G.

- Let $G = y\, E_1 \ldots E_m$ with $y \in \Delta_{rel}$. Note that it cannot be that $\varphi' = y\, E'_1 \ldots E'_m$, as relational variables do not occur in background formulas. Instead, it must be that $\varphi' = P(y)\, E'_1 \ldots E'_m$. For this to indeed be a background formula, $y : o$ or all x_i are of sort ι. In the former case, clearly, $G = y \to P(y) = \varphi'$. In the latter case, $\varphi' = P(y)\, E_1 \ldots E_m$ and also $G = y\, E_1 \ldots E_m \to P(y)\, E_1 \ldots E_m = \varphi'$.
- Let $G = G_1 \vee G_2$. Then, $\varphi' = \varphi'_1 \vee \varphi'_2 \in S_G$ for $\varphi'_1 \in S_{G_1}$ and $\varphi'_2 \in S_{G_2}$, with at least one of these satisfiable. By the induction hypothesis, there exists an SLD-refutation $G_1 \twoheadrightarrow \varphi_1$ or $G_2 \twoheadrightarrow \varphi_2$. It follows that $G = G_1 \vee G_2 \to G_i \twoheadrightarrow \varphi_i$ is an SLD-refutation for some $i \in \{1, 2\}$.
- Let $G = \exists x.\, G_1$. Then, $\varphi' = \exists x.\, \varphi'_1$ for some satisfiable $\varphi'_1 \in S_{G_1}$. By the induction hypothesis, there exists an SLD-refutation $G_1 \twoheadrightarrow \varphi_1$. It follows that $G = \exists x.\, G_1 \to G_1 \twoheadrightarrow \varphi_1$.

Note that the case where $G = (\lambda x_1 \ldots x_m.\, G_1)\, E_1 \ldots E_m$ does not apply.

Inductive case $m > 1$**.** Let $G = G_1 \wedge G_2$. Then, $\varphi' = \varphi'_1 \wedge \varphi'_2 \in S_G$ for satisfiable $\varphi'_1 \in S_{G_1}$ and $\varphi'_2 \in S_{G_2}$. By the induction hypothesis, there exist SLD-refutations $G_1 \twoheadrightarrow \varphi_1$ and $G_2 \twoheadrightarrow \varphi_2$. It follows that

$$G = G_1 \wedge G_2 \twoheadrightarrow \varphi_1 \wedge G_2 \twoheadrightarrow \varphi_1 \wedge \varphi_2$$

is an SLD-refutation. □

Lemma 5.15. *Let $\vdash P : \Delta_{rel}$ be a program, and $\Delta \vdash G : o$ and $\Delta \vdash G' : o$ goal terms such that $G' \in S_G$. If there exists an SLD-refutation $G' \twoheadrightarrow \varphi'$, then there also exists an SLD-refutation $G \twoheadrightarrow \varphi$.*

Proof. By induction on the length n of the SLD-refutation $G' \twoheadrightarrow \varphi'$. The result for $n = 0$ is proved in Lemma 5.14. In the remainder of the proof, we assume that G'—and thus G—is not background formula, because these cannot occur on the LHS of a proof rule.

Case $n = 1$**.** We proceed by structural induction G.

- Let $G = y\, E_1 \dots E_m$ with $y \in \Delta_{rel}$, so $G' = y\, E'_1 \dots E'_m$ or $G' = P(y)\, E'_1 \dots E'_m$, where either $E'_i = E_i$ (if $E_i : \iota$) or $E'_i \in S_{E_i}$, for all $i \in [m]$.

  Suppose the former. Then, $\varphi' = P(y)\, E'_1 \dots E'_m$. For this to indeed be a background formula, $y : o$ or all x_i are of sort ι. Both of these mean that $G = G'$, and the claim trivially holds.

  Suppose the latter. If $y : o$, then $G \to G'$ and the claim holds. Otherwise, $P(y)$ is of the form $\lambda x_1 \dots x_m.\, G_1$ for $m > 0$, so that $\varphi' = G_1[E'_1/x_1, \dots E'_m/x_m]$. For this to be a background formula, knowing that $m > 0$, only x_i of sort ι may occur in G_1. It follows that $\varphi' = G_1[E'_1/x_1, \dots E'_m/x_m] = G_1[E_1/x_1, \dots E_m/x_m]$ and

  $$G = y\, E_1 \dots E_m \to P(y)\, E_1 \dots E_m \to G_1[E_1/x_1, \dots E_m/x_m] = \varphi'$$

  is an SLD-refutation.

- Let $G = (\lambda x_1 \dots x_m.\, G_1)\, E_1 \dots E_m$ for some $m > 0$. Then, G' is of the form $(\lambda x_1 \dots x_m.\, G'_1)\, E'_1 \dots E'_m$ for some $G'_1 \in S_{G_1}$ and, for all $i \in [m]$, either $E'_i = E_i$ (if $E_i : \iota$) or $E'_i \in S_{E_i}$. Furthermore, $\varphi' = G'_1[E'_1/x_1, \dots, E'_m/x_m]$. For this to be a background formula, knowing that $m > 0$, only x_i of sort ι may occur in G'_1. Then, $\varphi' = G'_1[E'_1/x_1, \dots, E'_m/x_m] = G'_1[E_1/x_1, \dots, E_m/x_m]$, which we shorten to $G'_1[\overline{E}/\overline{x}]$.

  Note that $G'_1 \in S_{G_1}$ implies $G'_1[\overline{E}/\overline{x}] \in S_{G_1[\overline{E}/\overline{x}]}$. Since $\varphi' = G'_1[\overline{E}/\overline{x}]$ is a satisfiable background formula, we can use the induction hypothesis for SLD-refutations of length 0 to obtain $G_1[\overline{E}/\overline{x}] \twoheadrightarrow \varphi$. Then,

  $$G = (\lambda x_1 \dots x_m.\, G_1)\, E_1 \dots E_m \to G_1[E_1/x_1, \dots, E_m/x_m] \twoheadrightarrow \varphi.$$

- Let $G = G_1 \vee G_2$. Then, $G' = G'_1 \vee G'_2$ for some $G'_1 \in S_{G_1}$ and $G'_2 \in S_{G_2}$. Recall that G' cannot be a background formula, so at most one of G'_1 and G'_2 can be a constraint. Furthermore, because G' must resolve to a background formula in one step, exactly one of G'_1 and G'_2 is a constraint. Suppose WLOG that this is G'_1 and so $\varphi' = G'_1$.

  By the induction hypothesis for SLD-refutations of length 0, we obtain $G_1 \twoheadrightarrow \varphi$. It follows that

  $$G = G_1 \vee G_2 \to G_1 \twoheadrightarrow \varphi$$

  is an SLD-refutation.

- Let $G = \exists x.\, G_1$. Then, $G' = \exists x.\, G_1'$ for some $G_1' \in S_{G_1}$. In fact, G_1' must be a constraint, so $\varphi' = G_1'$. By the induction hypothesis for SLD-refutations of length 0, we obtain $G_1 \twoheadrightarrow \varphi$. Clearly, $G = \exists x.\, G_1 \to G_1 \twoheadrightarrow \varphi$ is an SLD-refutation.

- Let $G = G_1 \wedge G_2$, which is the inductive step in the proof for SLD-refutations of length 1. In this case, $G' = G_1' \wedge G_2'$ for some $G_1' \in S_{G_1}$ and $G_2' \in S_{G_2}$. By assumption, $G' = G_1' \wedge G_2' \to \varphi'$, so φ' must be a conjunction $\varphi_1' \wedge \varphi_2'$. One of two cases holds: $G_1' \to \varphi_1'$ and $G_2' = \varphi_2'$, or $G_2' \to \varphi_2'$ and $G_1' = \varphi_1'$.

  Suppose WLOG that the former holds. By the induction hypothesis for length 1 (on smaller goal terms), we obtain $G_1 \twoheadrightarrow \varphi_1$. By the induction hypothesis for length 0, we obtain $G_2 \twoheadrightarrow \varphi_2$. It follows that

  $$G = G_1 \wedge G_2 \twoheadrightarrow \varphi_1 \wedge G_2 \twoheadrightarrow \varphi_1 \wedge \varphi_2$$

  is an SLD-refutation.

Case $n > 1$**.** We again proceed by structural induction on G. Suppose that $G' \to G'' \to^k \varphi'$ and the claim holds for all $0 \leq k < n$.

- Let $G = y\, E_1 \ldots E_m$ with $y \in \Delta_{rel}$, so $G' = y\, E_1' \ldots E_m'$ or $G' = P(y)\, E_1' \ldots E_m'$, where either $E_i' = E_i$ (if $E_i : \iota$) or $E_i' \in S_{E_i}$, for all $i \in [m]$.

  Suppose the former. Then, $G'' = P(y)\, E_1' \ldots E_m'$. Then, $G'' \in S_{P(y)\, E_1 \ldots E_m}$. By the induction hypothesis, $P(y)\, E_1 \ldots E_m \twoheadrightarrow \varphi$. Clearly,

  $$G = y\, E_1 \ldots E_m \to P(y)\, E_1 \ldots E_m \twoheadrightarrow \varphi$$

  is an SLD-refutation.

  Suppose the latter. If $y : o$, then $G \to G'$ and the claim holds. Otherwise, $P(y)$ is of the form $\lambda x_1 \ldots x_m.\, G_1$ for $m > 0$, with $G'' = G_1[E_1'/x_1, \ldots E_m'/x_m]$. Then, $G'' \in S_{G_1[\overline{E}/\overline{x}]}$. By the induction hypothesis, $G_1[\overline{E}/\overline{x}] \twoheadrightarrow \varphi$. Clearly,

  $$G = y\, E_1 \ldots E_m \to P(y)\, E_1 \ldots E_m \to G_1[E_1/x_1, \ldots E_m/x_m] \twoheadrightarrow \varphi$$

  is an SLD-refutation.

- Let $G = (\lambda x_1 \ldots x_m.\, G_1)\, E_1 \ldots E_m$ for some $m > 0$. Then, G' is of the form $(\lambda x_1 \ldots x_m.\, G_1')\, E_1' \ldots E_m'$ for some $G_1' \in S_{G_1}$ and, for all $i \in [m]$, either $E_i' = E_i$

(if $E_i : \iota$) or $E'_i \in S_{E_i}$. Furthermore, $G'' = G'_1[E'_1/x_1, \ldots, E'_m/x_m]$. Note that $G'' \in S_{G_1[\overline{E}/\overline{x}]}$. By the induction hypothesis, $G_1[\overline{E}/\overline{x}] \twoheadrightarrow \varphi$. Clearly,

$$G = (\lambda x_1 \ldots x_m.\, G_1)\, E_1 \ldots E_m \to G_1[E_1/x_1, \ldots, E_m/x_m] \twoheadrightarrow \varphi$$

is an SLD-refutation.

- Let $G = G_1 \vee G_2$. Then, $G' = G'_1 \vee G'_2$ for some $G'_1 \in S_{G_1}$ and $G'_2 \in S_{G_2}$. Suppose WLOG that $G'' = G'_1$. By the induction hypothesis, $G_1 \twoheadrightarrow \varphi$. Clearly,

  $$G = G_1 \vee G_2 \to G_1 \twoheadrightarrow \varphi$$

  is an SLD-refutation.

- Let $G = \exists x.\, G_1$. Then, $G' = \exists x.\, G'_1$ for some $G'_1 \in S_{G_1}$, and $G'' = G'_1$. By the induction hypothesis, $G_1 \twoheadrightarrow \varphi$. Clearly,

  $$G = \exists x.\, G_1 \to G_1 \twoheadrightarrow \varphi$$

  is an SLD-refutation.

- Let $G = G_1 \wedge G_2$, which is the inductive step in the proof for SLD-refutations of length greater than 1. In this case, $G' = G'_1 \wedge G'_2$ for some $G'_1 \in S_{G_1}$ and $G'_2 \in S_{G_2}$. Then, $G'' = G''_1 \wedge G''_2$ and one of two cases holds: $G'_1 \to G''_1$ and $G'_2 = G''_2$, or $G'_2 \to G''_2$ and $G'_1 = G''_1$.

  Suppose WLOG that the former holds. Note that φ' must be a conjunction $\varphi'_1 \wedge \varphi'_2$ such that $G'_1 \to G''_1 \twoheadrightarrow \varphi'_1$ and $G'_2 = G''_2 \twoheadrightarrow \varphi'_2$. By the induction hypothesis for length $k+1$ (on smaller goal terms), we obtain $G_1 \twoheadrightarrow \varphi_1$. By the induction hypothesis for length k, we obtain $G_2 \twoheadrightarrow \varphi_2$. It follows that

  $$G = G_1 \wedge G_2 \twoheadrightarrow \varphi_1 \wedge G_2 \twoheadrightarrow \varphi_1 \wedge \varphi_2$$

  is an SLD-refutation.

□

Corollary 5.16. *Let $\vdash P : \Delta_{rel}$ be a program and $\Delta \vdash G : o$ a goal term. If there exists an SLD-refutation $\widehat{G} \twoheadrightarrow \varphi'$, then there exists an SLD-refutation $G \twoheadrightarrow \varphi$.*

We now present three auxiliary lemmas to support the proof of Theorem 5.20, which is the main technical result for completeness (Theorem 5.21).

Lemma 5.17. *Let $\vdash P : \Delta_{rel}$ be a program, and $\Delta \vdash G : o$ a goal term such that $\mathcal{C}[\![\Delta \vdash G : o]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. Then, there exists an SLD-refutation $B \twoheadrightarrow \varphi$ such that φ is satisfiable over α_{var}, where B is the β-normal form of G—written $G \Rightarrow_\beta B$.*

Proof. Let $G = G_1 \wedge \cdots \wedge G_m$, where each G_i is not a conjunction. We proceed by induction on m. We assume WLOG that constraint formulas are in β-normal form.

Base case $m = 1$. We use induction on length n of the β-reduction of G to B, within which we use structural induction on G. However, only the last subcase for G differs between $n = 0$ and $n > 0$ (in fact, $n = 0$ does not occur there), so we present the induction out of order to prevent duplication.

- Let $G = \varphi : o$ and $\mathcal{C}[\![\Delta \vdash \varphi : o]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. Clearly, φ is satisfiable, and G is in β-normal form, so $G = \varphi \twoheadrightarrow \varphi$ is an SLD-refutation with φ satisfiable over α_{var}.
- Let $G = y\, t_1 \ldots t_k$ with $y \in \Delta_{rel}$. This case does not occur, since the goal term $\mathcal{C}[\![\Delta \vdash y\, t_1 \ldots t_k]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var})$ would valuate to 0, because y is assigned the least element.
- Let $G = G_1 \vee G_2$. Suppose $\mathcal{C}[\![\Delta \vdash G_1 \vee G_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. This means that $\mathcal{C}[\![\Delta \vdash G_i]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$, for some $i \in \{1, 2\}$. Assume WLOG that $i = 1$.

  Let B_1 and B_2 denote the β-normal forms of G_1 and G_2, resp. By the induction hypothesis on smaller goal terms, $B_1 \twoheadrightarrow \varphi$ is an SLD-refutation with φ satisfiable over α_{var}. Clearly,

  $$G = G_1 \vee G_2 \Rightarrow_\beta B_1 \vee B_2 \to B_1 \twoheadrightarrow \varphi$$

  is an SLD-refutation that fits the criteria.
- Let $G = \exists x.\, H$. Suppose $\mathcal{C}[\![\Delta \vdash \exists x.\, H]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. Thus, there exists $t \in \mathcal{C}[\![\iota]\!]$ such that $\mathcal{C}[\![\Delta, x : \iota \vdash H]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}[x \mapsto t]) = 1$.

  Let B denote the β-normal form of H. By the induction hypothesis on smaller goal terms, $B \twoheadrightarrow \varphi$ is an SLD-refutation for some φ satisfiable over $\alpha_{var}[x \mapsto t]$. It follows that

  $$G = \exists x.\, H \Rightarrow_\beta \exists x.\, B \to B \twoheadrightarrow \varphi$$

is an SLD-refutation that fits the criteria.

- Let $G = (\lambda x_1 \dots x_k.\, H)\, t_1 \dots t_k$ and $\mathcal{C}[\![\Delta \vdash (\lambda x_1 \dots x_k.\, H)\, t_1 \dots t_k]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. Thus, $\mathcal{C}[\![\Delta \vdash H[t_1/x_1, \dots, t_k/x_k]]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$. Note that it must be that $n > 0$, because G is not in β-normal form.

  Let B be the β-normal form of $H[t_1/x_1, \dots, t_k/x_k]$ and G. By the induction hypothesis on length n of the β-reduction, there exists an SLD-refutation $B \twoheadrightarrow \varphi$ for some φ satisfiable over α_{var}. Clearly,

  $$G = (\lambda x_1 \dots x_k.\, H)\, t_1 \dots t_k \Rightarrow_\beta B \twoheadrightarrow \varphi$$

  is an SLD-refutation that fits the criteria.

Inductive case $m > 1$**.** Let $G = G_1 \wedge G_2$ and $\mathcal{C}[\![\Delta \vdash G_1 \wedge G_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. This means that $\mathcal{C}[\![\Delta \vdash G_i]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$, for all $i \in \{1, 2\}$.

Let B_1 and B_2 be the β-normal forms of G_1 and G_2, resp. By the induction hypothesis on m, $B_1 \twoheadrightarrow \varphi_1$ and $B_2 \twoheadrightarrow \varphi_2$ are SLD-refutations for some φ_1, φ_2 satisfiable over α_{var}. Let k_1 be the length of SLD-derivation $B_1 \twoheadrightarrow \varphi_1$ and k_2 the length of $B_2 \twoheadrightarrow \varphi_2$.

Note that the SLD-derivations may yield some additional free variables: $\Delta \vdash B_i$ but $\Delta, \Delta_i \vdash \varphi_i$, for $i \in \{1, 2\}$. If $k_1 = 0$ ($k_2 = 0$ analogous), then $\Delta_1 = \emptyset$. This means that $\varphi_1 \wedge \varphi_2$ is satisfiable over α_{var}. It follows that

$$G = G_1 \wedge G_2 \Rightarrow_\beta B_1 \wedge B_2 \twoheadrightarrow \varphi_1 \wedge \varphi_2$$

is an SLD-refutation that fits the criteria.

Now suppose the claim holds for pairs (k_1, k_2) up to (ℓ_1, ℓ_2). We prove that the claim holds for $k_1 = \ell_1 + 1$ and $k_2 = \ell_2$ (WLOG). We may assume that $k_1 > 0$ and $k_2 > 0$. We proceed by induction on the structure of B_1.

- Let $B_1 = \varphi : o$, $B_1 = y\, t_1 \dots t_k$, or $B_1 = (\lambda x_1 \dots x_k.\, H)\, t_1 \dots t_k$. This is a contradiction of $k_1 > 0$, $\mathcal{C}[\![\Delta \vdash G_1]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$, and B_1 being β-normal form, resp.

- Let $B_1 = B_{11} \vee B_{12}$, so that $\mathcal{C}[\![\Delta \vdash G_1 \wedge G_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = \mathcal{C}[\![\Delta \vdash B_1 \wedge B_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = \mathcal{C}[\![\Delta \vdash B_1]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = \mathcal{C}[\![\Delta \vdash B_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$. Thus, $\mathcal{C}[\![\Delta \vdash B_{1j}]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $j \in \{1, 2\}$. Assume WLOG that $j = 1$.

It follows that $\mathcal{C}[\![\Delta \vdash B_{11} \wedge B_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$. By the induction hypothesis on shorter SLD-refutations, $B_{11} \wedge B_2 \twoheadrightarrow \varphi$ is an SLD-refutation for some φ satisfiable over α_{var}. It follows that

$$G = G_1 \wedge G_2 \Rightarrow_\beta B_1 \wedge B_2 \to B_{11} \wedge B_2 \twoheadrightarrow \varphi$$

is an SLD-refutation that fits the criteria.

- Let $B_1 = \exists x.\, B'$, so that $\mathcal{C}[\![\Delta \vdash \exists x.\, B']\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$. Thus, there exists $t \in \mathcal{C}[\![\iota]\!]$ such that $\mathcal{C}[\![\Delta, x : \iota \vdash B']\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}[x \mapsto t]) = 1$.

  It follows that $\mathcal{C}[\![\Delta, x : \iota \vdash B' \wedge B_2]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}[x \mapsto t]) = 1$. By the induction hypothesis on shorter SLD-refutations, $B' \wedge B_2 \twoheadrightarrow \varphi$ for some φ satisfiable over α_{var}. It follows that

$$G = G_1 \wedge G_2 \Rightarrow_\beta B_1 \wedge B_2 \to B' \wedge B_2 \twoheadrightarrow \varphi$$

  is an SLD-refutation that fits the criteria.

□

Lemma 5.18. *If B is the β-normal form of goal term $G : o$—written $G \Rightarrow_\beta B$—and does not contain relational variables, and $B \twoheadrightarrow \varphi$ is an SLD-refutation, then $G \twoheadrightarrow \varphi$ is also an SLD-refutation.*

Proof. Our outer induction is on length n of β-reduction $G \Rightarrow_\beta B$. If $n = 0$, the result is immediate, so suppose $n > 0$.

Let $G = G_1 \wedge \cdots \wedge G_m$, where each G_i is not a conjunction. We proceed by induction on m. Again, we assume WLOG that constraint formulas are in β-normal form.

Base case $m = 1$**.** By case analysis on G.

- Let $G = \varphi : o$. This case does not occur, because it contradicts $n > 0$.
- Let $G = y\, t_1 \ldots t_k$ with $y \in \Delta_{rel}$. This case does not occur, since B is free of relational variables but β-reduction does not get rid of head variable $y \in \Delta_{rel}$.
- Let $G = G_1 \vee G_2$. Suppose $B_1 \vee B_2 \twoheadrightarrow \varphi$ is an SLD-refutation for B_1, B_2 such that $G_1 \Rightarrow_\beta B_1$ and $G_2 \Rightarrow_\beta B_2$. Note that $B_1 \vee B_2 \twoheadrightarrow \varphi$ is of the form $B_1 \vee B_2 \to B_i \twoheadrightarrow \varphi$, for some $i \in \{1, 2\}$.

By the induction hypothesis on smaller goal terms, $B_i \twoheadrightarrow \varphi$ gives us $G_i \twoheadrightarrow \varphi$. It follows that

$$G = G_1 \vee G_2 \to G_i \twoheadrightarrow \varphi$$

is an SLD-refutation.

- Let $G = \exists x.\, H$. Suppose $\exists x.\, B \twoheadrightarrow \varphi$ is an SLD-refutation for B such that $H \Rightarrow_\beta B$. Note that $\exists x.\, B \twoheadrightarrow \varphi$ is of the form $\exists x.\, B \to B \twoheadrightarrow \varphi$.

  By the induction hypothesis on smaller goal terms, $B \twoheadrightarrow \varphi$ gives us $H \twoheadrightarrow \varphi$. It follows that

  $$G = \exists x.\, H \to H \twoheadrightarrow \varphi$$

  is an SLD-refutation.

- Let $G = (\lambda x_1 \ldots x_k.\, H)\, t_1 \ldots t_k$. Suppose $B \twoheadrightarrow \varphi$ is an SLD-refutation for B such that $G \Rightarrow_\beta B$. Note that also $H[t_1/x_1, \ldots, t_k/x_k] \Rightarrow_\beta B$. Because the latter requires one fewer reduction step than G, we can invoke the induction hypothesis on n to give us $H[t_1/x_1, \ldots, t_k/x_k] \twoheadrightarrow \varphi$. It follows that

  $$G = (\lambda x_1 \ldots x_k.\, H)\, t_1 \ldots t_k \to H[t_1/x_1, \ldots, t_k/x_k] \twoheadrightarrow \varphi$$

  is an SLD-refutation.

Inductive case $m > 1$**.** Let $G = G_1 \wedge G_2$. Suppose $B_1 \wedge B_2 \twoheadrightarrow \varphi$ is an SLD-refutation for B_1, B_2 such that $G_1 \Rightarrow_\beta B_1$ and $G_2 \Rightarrow_\beta B_2$.

Suppose that the β-reduction $G_1 \Rightarrow_\beta B_1$ has length $n_1 > 0$ (the case for G_2 is analogous). We proceed by case analysis on G_1.

- Let $G_1 = \varphi : o$. This case does not occur, because it contradicts $n_1 > 0$.
- Let $G_1 = y\, t_1 \ldots t_k$ with $y \in \Delta_{rel}$. This case does not occur, since B_1 is free of relational variables but β-reduction does not get rid of head variable $y \in \Delta_{rel}$.
- Let $G_1 = G_{11} \vee G_{12}$. Note that we have $(B_{11} \vee B_{12}) \wedge B_2 \twoheadrightarrow \varphi$ for B_{11}, B_{12} such that $G_{11} \Rightarrow_\beta B_{11}$ and $G_{12} \Rightarrow_\beta B_{12}$. In particular, there exists an SLD-refutation of the form

  $$(B_{11} \vee B_{12}) \wedge B_2 \to B_{1j} \wedge B_2 \twoheadrightarrow \varphi$$

  for some $j \in \{1, 2\}$. The induction hypothesis on smaller goal terms gives us $G_{1j} \wedge G_2 \twoheadrightarrow \varphi$. Finally,

  $$G = G_1 \wedge G_2 = (G_{11} \vee G_{12}) \wedge G_2 \to G_{1j} \wedge G_2 \twoheadrightarrow \varphi.$$

- Let $G_1 = \exists x.\, H$. We obtain $H \Rightarrow_\beta B_1'$ from $G_1 \Rightarrow_\beta B_1 = \exists x.\, B_1'$. Thus,

$$H \wedge G_2 \Rightarrow_\beta B_1' \wedge B_2 \twoheadrightarrow \varphi.$$

  The induction hypothesis on smaller goal terms gives us

$$G = G_1 \wedge G_2 = \exists x.\, H \wedge G_2 \to H \wedge G_2 \twoheadrightarrow \varphi,$$

  as required.

- Let $G_1 = (\lambda x_1 \dots x_k.\, H)\, t_1 \dots t_k$. We obtain $H[t_1/x_1, \dots, t_k/x_k] \Rightarrow_\beta B_1$ from $G_1 \Rightarrow_\beta B_1$. Thus,

$$H[t_1/x_1, \dots, t_k/x_k] \wedge G_2 \Rightarrow_\beta B_1 \wedge B_2 \twoheadrightarrow \varphi.$$

  The induction hypothesis on shorter β-reductions gives us

$$G = G_1 \wedge G_2 = (\lambda x_1 \dots x_k.\, H)\, t_1 \dots t_k \wedge G_2 \to H[t_1/x_1, \dots, t_k/x_k] \wedge G_2 \twoheadrightarrow \varphi,$$

  as required.

□

Lemma 5.19. *Let $\vdash P : \Delta_{rel}$ be a program, and $\Delta \vdash G : o$ a goal term such that $\mathcal{C}[\![\Delta \vdash G : o]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$. Then, there exists an SLD-refutation $G \twoheadrightarrow \varphi$.*

Proof. Suppose $\mathcal{C}[\![\Delta \vdash G : o]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$. By Lemma 5.17, there exists an SLD-refutation $B \twoheadrightarrow \varphi$ for φ satisfiable over α_{var} and B the β-normal form of G.

Before we can apply Lemma 5.18, we need to prove that B does not contain relational variables. We proceed by induction on the structure of B.

- Let $B = \varphi : o$, which is a formula of the background theory that trivially does not contain relational variables.
- Let $B = y\, t_1 \dots t_k$ with $y \in \Delta_{rel}$. The semantics is invariant over β-reduction, so that we obtain $\mathcal{C}[\![\Delta \vdash B : o]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 1$ from our assumption. However, y is assigned the least relation under valuation $\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}$, so that $\mathcal{C}[\![\Delta \vdash B : o]\!](\bot_{\mathcal{C}[\![\Delta_{rel}]\!]} \cup \alpha_{var}) = 0$. This is a contradiction.
- Let $B = (\lambda x_1 \dots x_k.\, B')\, t_1 \dots t_k$ for $k > 0$. This case does not occur, because B is not in β-normal form.

The remaining cases follow directly from the induction hypothesis. We conclude that B does not contain relational variables.

Finally, we apply Lemma 5.18 to obtain that $G \twoheadrightarrow \varphi$ is also an SLD-refutation. □

Theorem 5.20. *Let $\vdash P : \Delta_{rel}$ be a program and $\Delta \vdash G : o$ a goal term. Suppose that $\mathcal{C}[\![\Delta \vdash G : o]\!](M_P \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta_{var}]\!]$, where M_P is the least model of P. Then, there exists an SLD-refutation $G \twoheadrightarrow \varphi$.*

Proof. Let $T^n := T^{\mathcal{C}\,n}_{P:\Delta_{rel}}(\bot_{\mathcal{C}[\![\Delta_{rel}]\!]}) \in \mathcal{C}[\![\Delta_{rel}]\!]$ denote the nth iteration the one-step consequence operator, for all $n \geq 0$. Note that continuity of our semantics guarantees that $\mathcal{C}[\![\Delta \vdash G : o]\!](M_P \cup \alpha_{var}) = 1$ implies $\mathcal{C}[\![\Delta \vdash G : o]\!](T^n \cup \alpha_{var}) = 1$ for some $n \geq 0$.

We prove by induction on n that $\mathcal{C}[\![\Delta \vdash G]\!](T^n \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta]\!]$ and $n \geq 0$ guarantees the existence of an SLD-refutation $G \twoheadrightarrow \varphi$. Lemma 5.19 corresponds to the base case $n = 0$.

Suppose the claim holds for some $n \geq 0$. Assume that $\mathcal{C}[\![\Delta \vdash G]\!](T^{n+1} \cup \alpha_{var}) = 1$ for some $\alpha_{var} \in \mathcal{C}[\![\Delta]\!]$. By Lemma 5.12, $\mathcal{C}[\![\Delta \vdash \widehat{G}]\!](T^n \cup \alpha_{var}) = 1$.

By the induction hypothesis, there exists an SLD-refutation $\widehat{G} \twoheadrightarrow \varphi'$. Finally, Corollary 5.16 implies that there is also an SLD-refutation $G \twoheadrightarrow \varphi$. □

Theorem 5.21 (Completeness)**.** *If HoCHC problem $\langle \Delta_{rel}, P, G \rangle$ is unsolvable, then $G \twoheadrightarrow \varphi$ for some satisfiable background formula φ.*

Proof. Let M_P be the least model of P. Suppose $\langle \Delta_{rel}, P, G \rangle$ is unsolvable. This means that $\mathcal{C}[\![\Delta_{rel} \vdash G]\!](M_P) = 1$, so that completeness follows from Theorem 5.20 (with Δ_{var} empty). □

5.5 Termination

Theorem 5.22 (Reduction)**.** *A HoCHC problem $\langle \Delta_{rel}, P, G \rangle$ is unsolvable if and only if there exists an SLD-refutation $G \twoheadrightarrow \varphi$ for some constraint formula φ.*

Proof. A consequence of soundness (Theorem 5.10) and completeness (Theorem 5.21). □

The question remains whether we can always find an SLD-refutation in finite time, given an unsolvable HoCHC instance over a semi-decidable background theory. This matter comes down to countability; do we have a countable number of potential SLD-refutations that we can iterate over in some clever order?

Proposition 5.23. *There are countably many SLD-refutations for each* $\langle \Delta_{rel}, P, G \rangle$.

Proof. We proceed by induction on the length n of SLD-refutations to show that there are countably many SLD-refutation $G \twoheadrightarrow \varphi$ (for some φ) of length at most n. Recall that SLD-refutations have finite length.

If $n = 0$, then $G = \varphi : o$ is satisfiable, and we are done. There is one SLD-refutation of G.

Suppose the claim holds for SLD-refutations up to some length $n \geq 0$. Suppose $G \to G'$ and there are countably many SLD-refutations $G' \twoheadrightarrow \varphi$ of length up to n. We show that there are countably many SLD-refutations $G \twoheadrightarrow \varphi$ of length up to $n+1$ by case analysis on G.

- If $G = y\, E_1 \ldots E_m$, $G = (\lambda x_1 \ldots x_m.\, H)\, E_1 \ldots E_m$, or $G = \exists x.\, H$, then there is only one goal term G' such that $G \to G'$, and the claim is trivial.
- If $G = G_1 \vee G_2$, then there are two G' such that $G \to G'$, so there is still a countable number of SLD-refutations.
- Otherwise, if $G = G_1 \wedge G_2$, then $G' = G'_1 \wedge G'_2$ where either $G_1 \to G'_1$ and $G_2 = G'_2$, or $G_2 \to G'_2$ and $G_1 = G'_1$. We invoke the induction hypothesis on the smaller goal terms, from which the claim then follows.

□

Theorem 5.24 (Termination)**.** *If a HoCHC problem* $\langle \Delta_{rel}, P, G \rangle$ *is unsolvable over a semi-decidable background theory, then an SLD-refutation of* $\langle \Delta_{rel}, P, G \rangle$ *can be found in finite time.*

Proof. A direct consequence of semi-decidability of the background theory and dovetailing the countably many higher-order SLD-refutations of Proposition 5.23. □

5.6 Examples

We provide two examples of unsolvable HoCHC instances for which a counterexample can be found using SLD-resolution. The first uses the background theory of Linear Integer Arithmetic with its standard model and the second Maher's theory of trees (from Chapter 4.6.1) with the set of all finite and infinite trees as designated model.

Example 5.25 (HoCHC over Linear Integer Arithmetic)**.** Consider HoCHC problem $\langle \{Sum : \iota \to \iota \to o\}, P, G\rangle$ over Linear Integer Arithmetic with program P:

$$Sum = \lambda n\, m.\, (n \leq 0 \wedge m = 0) \vee (n > 0 \wedge \exists p.\, Sum\, (n-1)\, p \wedge m = n + p)$$

Goal term G is given by:

$$G := \exists k\, l.\, Sum\, k\, l \wedge k < l.$$

Note that $Sum\, n\, m$ holds of precisely those pairs n, m where $m = 1 + 2 + 3 + \cdots + n$. Before we proceed with an SLD-refutation of $\langle \{Sum : \iota \to \iota \to o\}, P, G\rangle$, let us first consider two subrefutations. We label the first subrefutation ‡:

$$\begin{aligned}
&\exists p_2.\, Sum\, (k-2)\, p_2 \\
&\to Sum\, (k-2)\, p_2 \\
&\to (\lambda n\, m.\, (n \leq 0 \wedge m = 0) \vee (n > 0 \wedge \exists p_3.\, Sum\, (n-1)\, p_3 \wedge m = n + p_3))\, (k-2)\, p_2 \\
&\to (k-2 \leq 0 \wedge p_2 = 0) \vee (k-2 > 0 \wedge \exists p_3.\, Sum\, (k-3)\, p_3 \wedge p_2 = k - 2 + p_3) \\
&\to k - 2 \leq 0 \wedge p_2 = 0
\end{aligned}$$

The second subderivation is labelled † and relies on ‡ in the final step:

$$\begin{aligned}
&\exists p_1.\, Sum\, (k-1)\, p_1 \\
&\to Sum\, (k-1)\, p_1 \\
&\to (\lambda n\, m.\, (n \leq 0 \wedge m = 0) \vee (n > 0 \wedge \exists p_2.\, Sum\, (n-1)\, p_2 \wedge m = n + p_2))\, (k-1)\, p_1 \\
&\to (k-1 \leq 0 \wedge p_1 = 0) \vee (k-1 > 0 \wedge \exists p_2.\, Sum\, (k-2)\, p_2 \wedge p_1 = k - 1 + p_2) \\
&\to k - 1 > 0 \wedge \exists p_2.\, Sum\, (k-2)\, p_2 \wedge p_1 = k - 1 + p_2 \\
&\to k - 1 > 0 \wedge k - 2 \leq 0 \wedge p_2 = 0 \wedge p_1 = k - 1 + p_2
\end{aligned}$$

Finally, we obtain an SLD-refutation of $\langle \{Sum : \iota \to \iota \to o\}, P, G\rangle$, using † in the final step:

$$
\begin{aligned}
&\exists k\, l.\, Sum\, k\, l \wedge k < l \\
&\to \exists l.\, Sum\, k\, l \wedge k < l \\
&\to Sum\, k\, l \wedge k < l \\
&\to (\lambda n\, m.\, (n \leq 0 \wedge m = 0) \vee (n > 0 \wedge \exists p_1.\, Sum\, (n-1)\, p_1 \wedge m = n + p_1))\, k\, l \\
&\qquad\qquad \wedge\ k < l \\
&\to ((k \leq 0 \wedge l = 0) \vee (k > 0 \wedge \exists p_1.\, Sum\, (k-1)\, p_1 \wedge l = k + p_1)) \wedge k < l \\
&\to k > 0 \wedge \exists p_1.\, Sum\, (k-1)\, p_1 \wedge l = k + p_1 \wedge k < l \\
&\to k > 0 \wedge k - 1 > 0 \wedge k - 2 \leq 0 \wedge p_2 = 0 \wedge p_1 = k - 1 + p_2 \wedge l = k + p_1 \wedge k < l
\end{aligned}
$$

We have now derived a formula of the background theory of Linear Integer Arithmetic. This formula is satisfied for $k = 2$ and $l = 3$. Thus, the HoCHC problem $\langle \{Sum : \iota \to \iota \to o\}, P, G\rangle$ is unsolvable.

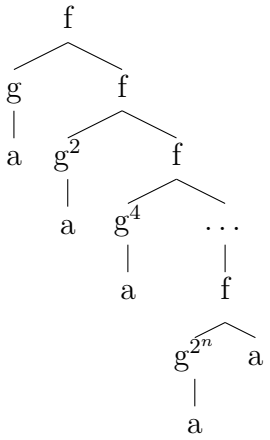
Example 5.26 (HoCHC over finite trees)**.** Let us consider HoCHC problem $\langle \Delta, P, G\rangle$ over Maher's decidable theory of trees with equality from Chapter 4.6.1 (Maher, 1988), where Δ is the set consisting of

$$
\begin{aligned}
S &: \iota \to o \\
B &: (\iota \to \iota \to o) \to (\iota \to \iota \to o) \to \iota \to \iota \to o \\
F &: (\iota \to \iota \to o) \to \iota \to o \\
Sub &: \iota \to \iota \to o
\end{aligned}
$$

and $\vdash P : \Delta$ is given by:

$$
\begin{aligned}
S &= \lambda r.\, F\, (\lambda y\, r_1.\, g\, y = r_1)\, r \\
B &= \lambda \varphi\, \psi\, x\, r_2.\, \exists r_3.\, \varphi\, r_3\, r_2 \wedge \psi\, x\, r_3 \\
F &= \lambda \varphi\, r_5.\, \exists r_6\, r_7.\, (f\, r_6\, r_7 = r_5) \wedge \varphi\, a\, r_6 \wedge (r_7 = a \vee F\, (B\, \varphi\, \varphi)\, r_7) \\
Sub &= \lambda s\, t.\, (s = t) \vee \exists s'\, s''.\, (t = g\, s' \wedge Sub\, s\, s') \vee (t = f\, s'\, s'' \wedge (Sub\, s\, s' \vee Sub\, s\, s''))
\end{aligned}
$$

Note that $Sub\, s\, t$ holds if and only if s is a subtree of t—assuming terminal symbols drawn from $\Sigma = \{f, g, a\}$. Furthermore, $S\, t$ holds for precisely those (finite) trees of the form



for some n.

Goal term G is defined by:

$$G := \exists t.\, S\, t \wedge Sub\,(g^2\, a)\, t$$

Intuitively, this goal term states that there exists a tree in the language of S that has subtree $g\,(g\,a)$. We can now use SLD-resolution to demonstrate unsolvability of $\langle \Delta, P, G \rangle$. We combine pairs of relational variable expansions and lambda reductions in one step to save space. Let us write X as a shorthand for $\lambda y\, r_1.\, g\, y = r_1$.

$$\begin{aligned}
\exists t.\, S\, t \wedge Sub\,(g^2\, a)\, t &\rightarrow S\, t \wedge Sub\,(g^2\, a)\, t \\
&\rightarrow F\, X\, t \wedge Sub\,(g^2\, a)\, t \\
&\rightarrow \exists r_6\, r_7.\, (f\, r_6\, r_7 = t) \wedge X\, a\, r_6 \\
&\qquad\qquad \wedge\ (r_7 = a \vee F\,(B\, X\, X)\, r_7) \wedge Sub\,(g^2\, a)\, t \\
&\rightarrow \exists r_7.\, (f\, r_6\, r_7 = t) \wedge X\, a\, r_6 \\
&\qquad\qquad \wedge\ (r_7 = a \vee F\,(B\, X\, X)\, r_7) \wedge Sub\,(g^2\, a)\, t \\
&\rightarrow (f\, r_6\, r_7 = t) \wedge X\, a\, r_6 \\
&\qquad\qquad \wedge\ (r_7 = a \vee F\,(B\, X\, X)\, r_7) \wedge Sub\,(g^2\, a)\, t \\
&\rightarrow (f\, r_6\, r_7 = t) \wedge (\lambda r_1.\, g\, a = r_1)\, r_6 \\
&\qquad\qquad \wedge\ (r_7 = a \vee F\,(B\, X\, X)\, r_7) \wedge Sub\,(g^2\, a)\, t \\
&\rightarrow (f\, r_6\, r_7 = t) \wedge (g\, a = r_6) \\
&\qquad\qquad \wedge\ (r_7 = a \vee F\,(B\, X\, X)\, r_7) \wedge Sub\,(g^2\, a)\, t
\end{aligned}$$

Let us reduce $F\,(B\,X\,X)\,r_7$ separately:

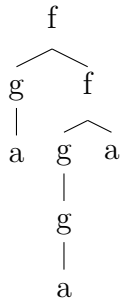
$$\begin{aligned}
F\,(B\,X\,X)\,r_7 &\to \exists r_6\,r_8.\,(f\,r_6\,r_8 = r_7) \wedge B\,X\,X\,a\,r_6 \\
&\qquad\qquad \wedge\ (r_8 = a \vee F\,(B\,(B\,X\,X)\,(B\,X\,X))\,r_8) \\
&\to \exists r_8.\,(f\,r_6\,r_8 = r_7) \wedge B\,X\,X\,a\,r_6 \\
&\qquad\qquad \wedge\ (r_8 = a \vee F\,(B\,(B\,X\,X)\,(B\,X\,X))\,r_8) \\
&\to (f\,r_6\,r_8 = r_7) \wedge B\,X\,X\,a\,r_6 \\
&\qquad\qquad \wedge\ (r_8 = a \vee F\,(B\,(B\,X\,X)\,(B\,X\,X))\,r_8) \\
&\to (f\,r_6\,r_8 = r_7) \wedge B\,X\,X\,a\,r_6 \wedge (r_8 = a) \\
&\to (f\,r_6\,r_8 = r_7) \wedge \exists r_3.\,X\,r_3\,r_6 \wedge X\,a\,r_3 \wedge (r_8 = a) \\
&\to (f\,r_6\,r_8 = r_7) \wedge X\,r_3\,r_6 \wedge X\,a\,r_3 \wedge (r_8 = a) \\
&\to (f\,r_6\,r_8 = r_7) \wedge (\lambda r_1.\,g\,r_3 = r_1)\,r_6 \wedge X\,a\,r_3 \wedge (r_8 = a) \\
&\to (f\,r_6\,r_8 = r_7) \wedge (g\,r_3 = r_6) \wedge X\,a\,r_3 \wedge (r_8 = a) \\
&\to (f\,r_6\,r_8 = r_7) \wedge (g\,r_3 = r_6) \wedge (\lambda r_1.\,g\,a = r_1)\,r_3 \wedge (r_8 = a) \\
&\to (f\,r_6\,r_8 = r_7) \wedge (g\,r_3 = r_6) \wedge (g\,a = r_3) \wedge (r_8 = a)
\end{aligned}$$

Note that all variables but r_7 are free, so that this simplifies to $r_7 = f\,(g\,(g\,a))\,a$. We continue our SLD-refutation of G, in-lining the above:

$$\begin{aligned}
&(f\,r_6\,r_7 = t) \wedge (g\,a = r_6) \wedge (r_7 = a \vee F\,(B\,X\,X)\,r_7) \wedge Sub\,(g^2\,a)\,t \\
&\to^* (f\,r_6\,r_7 = t) \wedge (g\,a = r_6) \wedge F\,(B\,X\,X)\,r_7 \wedge Sub\,(g^2\,a)\,t \\
&\to^* (f\,r_6\,r_7 = t) \wedge (g\,a = r_6) \wedge (r_7 = f\,(g\,(g\,a))\,a) \wedge Sub\,(g^2\,a)\,t
\end{aligned}$$

It is not hard to see that Sub behaves as expected, and that this will derive a witness

$t =$



to the unsolvability of $\langle \Delta, P, G \rangle$.

5.7 Solution sets

Recall that the HoCHC problem is not interested in solution sets of all witnesses to unsolvability. In fact, the goal term $G : o$ in $\langle \Delta_{rel}, P, G \rangle$ takes its variables only from Δ_{rel}, so strictly speaking there are no witnesses. However, in practice, a goal term contains internal existential quantifiers, so we do end up with witnesses to unsolvability (provided the HoCHC problem is indeed unsolvable).

Because we do not care about solution sets, we have eliminated higher-order existential quantification from goal terms and logic programs, as described at the start of this chapter. This simplifies the proof system greatly, while still preserving some witness-finding, just not all.

Finding the full solution set to a HoCHC problem requires several (five) sacrifices of simplicity and genericity with respect to the background theory to achieve completeness.

The completeness proof by Charalambidis et al. (2013)—who use full solution sets—relies on a one-to-one correspondence between enumerable compact elements of argument sort (i.e. either ι or a relational sort)—called *basic elements*—and terms of a particular form—*basic expressions*. To define these basic expressions and ensure there are countably many, they impose the following restrictions on the background theory:

1. Assume the poset of individuals is definable and countable
2. Assume equality is part of the background theory

Definition 5.27 (Definable poset of individuals)**.** *The (discretely ordered) poset A_ι of individuals is* definable *in background theory Th if for every element $b \in A_\iota$ there exists a closed applicative term $B : \iota$ over Σ such that $b = \mathcal{S}[\![B]\!]$.*

Example 5.28 (Undefinable poset of individuals)**.** Consider $\Sigma = \{f, a\}$ with f unary and a nullary, and A_ι the set of all finite and infinite trees over Σ. There exists a basic element that does not have a syntactic equivalent of sort ι; the infinite tree $f\ f\ f \ldots$ cannot be captured by the meaning of some basic expression, which is a finite term by definition.

The Maher theory of trees (Maher, 1988) is an example of a background theory that does not allow this modified SLD-resolution-with-solution-sets; although tree equality

is part of the theory and recursive definitions are allowed, infinite trees (in particular, irregular ones) are not definable as per Definition 5.27.

Furthermore, the resolution procedure would be more complicated if we were to compute solution sets. SLD-refutations would 'guess' values for higher-order relations and, thus, be annotated with a substitution:

3. Use substitutions in SLD-refutations
4. Add a proof rule $x\,E_1 \ldots E_n \to (x\,E_1 \ldots E_n)[B/x]$ for higher-order variables

Finally, to establish completeness of SLD-resolution-with-solution-sets for HoCHC, we would need to demonstrate that these basic expressions capture the full semantic domain. The intuition is that, because resolution uses syntactic substitutions and rewrites, finite terms must encompass the entire semantic domain for completeness. In a continuous semantics, this is straightforward from definability of the individuals. After all, nothing is 'invented' at infinity. For any function $f \in \mathcal{C}[\![\rho_1 \to \rho_2]\!]$, the value of $f(x)$ for an infinite element $x \in \mathcal{C}[\![\rho_1]\!]$ is uniquely determined by $f(y)$ of all its smaller, finite/compact elements $y \in \mathcal{C}[\![\rho_1]\!]$.

Charalambidis et al. (2013) use a variant of the continuous semantics for which this finite-sufficiency is more apparent: an *effectively continuous* interpretation in which function domains are restricted to finite elements. We show this interpretation to be isomorphic with the continuous interpretation in Chapter 3.5. Although it may not be strictly necessary to adopt the effectively continuous interpretation of SLD-resolution-with-solution-sets for HoCHC, it would simplify correctness proofs, particularly completeness.

5. Use an *effectively continuous* interpretation (optional)

Chapter 6

An axiomatic basis for relations as higher-order program invariants

Hoare logic is a deductive system for reasoning about the correctness of (first-order imperative) programs first introduced by Hoare (1969), building on work by Floyd (1967). The original paper by Hoare provides axioms and inference rules for the constructs of a simple imperative programming language containing iteration in the form of a while loop.

A formula of Hoare logic is an asserted program $\{\varphi\}\,\Pi\,\{\psi\}$ where Π is a program and φ, ψ are assertions from an *assertion logic*. The intuition is that, given a state in which φ holds (the *precondition*), every terminating execution of the program Π results in a state in which ψ holds (the *postcondition*).

We also call $\{\varphi\}\,\Pi\,\{\psi\}$ a *Hoare triple* or a *partial correctness* statement. If, in addition to partial correctness, we require the execution of Π in a state φ to terminate, we obtain *total correctness*:

$$\text{total correctness} = \text{partial correctness} + \text{termination}$$

Although the language of programs varies across different applications of Hoare logic—with the inference rules adapted accordingly—assertions are typically expressed in first-order logic or an extension thereof.

Hoare logic has found applications in program verification as well as contract programming, which is natively supported by languages such as Eiffel, Ada, and Scala. Due to the *soundness* of Hoare logic, i.e. only valid triples can be proved, the existence of a proof implies the correctness of a program with respect to a pre- and

postcondition. In contract programming (Meyer, 1992; Findler and Felleisen, 2002), the designer must formally specify a precondition, postcondition, and invariant for each method, analogous to Hoare logic, which enforces a priori program correctness.

For verifying higher-order programs, and specifically for compositional verification where a program may not be of ground sort, it is natural to consider assertions expressed in a higher-order logic to capture a wider range of correctness properties of higher-order programs.

In this chapter, we develop an axiomatic basis for relations as higher-order program invariants, which is a form of higher-order Hoare logic. As before, we work in a presentation of higher-order logic as a simply sorted (typed) lambda calculus.

We choose call-by-value PCF as our source language. PCF is a prototype functional language that lies at the core of Haskell and ML but is syntactically and—especially—semantically simpler than those languages. It has simple sorts over a single base sort. The lack of polymorphism (compared to e.g. System F) allows us to relate it to HoCHC in a natural way.

First, we present call-by-value PCF in Section 6.1. Then, we introduce the proof system in Section 6.2, as well as the higher-order Hoare triples lying at its core. Correctness of the proof system is established in Section 6.3. Finally, we relate our higher-order Hoare logic to the HoCHC fragment and to refinement types (Section 6.4 and 6.5, resp.).

6.1 Call-by-value PCF

We assume that our programs are expressed in PCF (*Programming Computable Functions*), which is a simple, typed functional language first introduced by Plotkin (1977) based on Dana Scott's LCF (*Logic of Computable Functions*, written in 1969 but unpublished until Scott, 1993). In particular, we use a call-by-value variant of this language, denoted by $\mathrm{PCF}_{\mathrm{v}}$, that was studied in detail by Sieber (1990).

The syntax of $\mathrm{PCF}_{\mathrm{v}}$ is straightforward. It is a simply sorted lambda calculus endowed with conditional and fixpoint operators and some first-order arithmetic operators.

We refer to $\mathrm{PCF}_{\mathrm{v}}$ programs exclusively as *(program) expressions* to distinguish them from HoCHC logic programs.

Expressions. Expressions are defined by the following grammar, where e denotes an expression and v a value:

$$v ::= n \in \mathbb{N} \mid \lambda x : \sigma.\, e$$
$$e ::= x \mid v \mid e_1\, e_2 \mid \mathsf{if}(e_1, e_2, e_3) \mid \mathsf{fix}\, e \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \leq e_2 \mid \mathsf{pif}(e_1, e_2, e_3)$$

We define the free variables $\mathsf{FV}(e)$ of an expression e as usual. An expression with no free variables is said to be *closed*. If an expression $e : \sigma$ is well-sorted over some sort environment $\Delta \supseteq \mathsf{FV}(e)$, we write $\Delta \vdash e : \sigma$.

To be precise, $\mathrm{PCF_v}$ generates natural numbers from constant 0 through successor and predecessor functions, i.e. from finite signature $\{0 : \iota, \mathsf{succ} : \iota \to \iota, \mathsf{pred} : \iota \to \iota\}$. For convenience, however, we regard each natural number n as a constant of sort ι.

Sorts. Sorts are defined by the following grammar:

$$\sigma ::= \iota \mid \sigma_1 \to \sigma_2$$

Note that $\mathrm{PCF_v}$ is often assumed to have simple sorts over boolean sort o as well as over individual sort ι. However, we choose to consider booleans as integers to not conflict with the relations in the preconditions and postconditions of our Hoare triples, defined in Section 6.2. This difference arises only in the conditional, where we treat all nonzero guards as if they were true; the interpretations are otherwise identical.

6.1.1 Semantics

There are several different semantics for $\mathrm{PCF_v}$ expressions. We present three equivalent ones and move between them at our convenience. They share the following interpretation of sorts.

Sort interpretation. For each sort σ, we distinguish a dcpo of values V^σ and a dcpo of computations C^σ that differ in the addition of a *divergent* (non-terminating) element $\bot$ to the dcpo of computations. For a dcpo D, we write $D_\bot$ for D with the addition of least element $\bot$. Thus, for all σ, $C^\sigma = V^\sigma_\bot$.

$\mathrm{PCF_v}$ sorts are interpreted inductively as

$$\begin{aligned} V^\iota &:= \langle \mathbb{N}, = \rangle \\ C^\iota &:= \langle \mathbb{N}, = \rangle_\perp \\ V^{\sigma\to\tau} &:= V^\sigma \Rightarrow_c C^\tau \\ C^{\sigma\to\tau} &:= [V^\sigma \Rightarrow_c C^\tau]_\perp \end{aligned}$$

where $A \Rightarrow_c B$ denotes the continuous function space between dcpos A and B. Note that $\langle \mathbb{N}, = \rangle$ is a discrete poset, which we may write as simply $\mathbb{N}$, so that $\langle \mathbb{N}, = \rangle_\perp$ is a *flat* dcpo of incomparable natural numbers with least element $\perp$.

We shall write $\perp_{V^\sigma}$ for the (unique) least element of V^σ, the constant function that maps all inputs to $\perp$ in the codomain, not to be confused with divergent element $\perp \notin V^\sigma$.

These definitions extend pointwise to sort environments. Thus, we write V^Δ for $\Pi_{x:\sigma\in\Delta} V^\sigma$, where Δ is a sort environment.

The meaning of a closed expression $\vdash e : \sigma$ is an element of C^σ. The meaning of a general expression $\Delta \vdash e : \sigma$ is a function from V^Δ to C^σ.

6.1.1.1 Operational semantics

One way of defining the meaning of an expression is through its computational behaviour. The *operational semantics* $e \Downarrow v$ of closed expression e is defined in Figure 6.1.

$$\frac{}{v \Downarrow v} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 \le e_2 \Downarrow 1}(n_1 \le n_2) \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 \le e_2 \Downarrow 0}(n_1 > n_2)$$

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 + e_2 \Downarrow n}(n = n_1 + n_2) \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 - e_2 \Downarrow n}(n = n_1 \dot{-} n_2)$$

$$\frac{e_1 \Downarrow n \qquad e_2 \Downarrow v}{\mathsf{if}(e_1, e_2, e_3) \Downarrow v}\,(n > 0) \qquad \frac{e_1 \Downarrow 0 \qquad e_3 \Downarrow w}{\mathsf{if}(e_1, e_2, e_3) \Downarrow w}$$

$$\frac{e_1 \Downarrow \lambda x.\, e \qquad e_2 \Downarrow w \qquad e[w/x] \Downarrow v}{e_1\, e_2 \Downarrow v} \qquad \frac{e_1 \Downarrow v}{\mathsf{fix}\, e_1 \Downarrow \lambda x.\, v\, (\mathsf{fix}\, v)\, x}$$

$$\frac{e_1 \Downarrow n \qquad e_2 \Downarrow v}{\mathsf{pif}(e_1, e_2, e_3) \Downarrow v}\,(n > 0) \qquad \frac{e_1 \Downarrow 0 \qquad e_3 \Downarrow w}{\mathsf{pif}(e_1, e_2, e_3) \Downarrow w} \qquad \frac{e_2 \Downarrow n \qquad e_3 \Downarrow n}{\mathsf{pif}(e_1, e_2, e_3) \Downarrow n}$$

Figure 6.1: Operational semantics of $\mathrm{PCF_v}$ program expressions

Note that subtraction is interpreted as a *monus* operation, written $\dot{-}$, to account for the lack of negative integers:

$$n_1 \dot{-} n_2 := \begin{cases} 0 & \text{if } n_1 < n_2 \\ n_1 - n_2 & \text{if } n_1 \geq n_2 \end{cases}$$

We define the short-hands:

$$e{\Downarrow} := \exists v.\, e \Downarrow v \qquad e{\Uparrow} := \neg\exists v.\, e \Downarrow v$$

A closed expression e is said to *converge* just if $e{\Downarrow}$ and *diverge* just if $e{\Uparrow}$.

The call-by-value nature of $\mathrm{PCF_v}$ is apparent in the application rule, where the arguments are evaluated before the application itself is evaluated. In particular, this means that if e_2 diverges so does $e_1\, e_2$, regardless of whether e_1 uses its argument.

6.1.1.2 One-step semantics

Another formalism of meaning is given through a *one-step (operational) semantics*. The one-step semantics $e \to e'$ of closed expression e is defined in Figure 6.2.

(OSIf_1)	$\mathsf{if}(n, e_1, e_0)$	$\to$	e_1	$(n > 0)$
(OSIf_0)	$\mathsf{if}(0, e_1, e_0)$	$\to$	e_0	
(OSIf)	$\mathsf{if}(e, e_1, e_0)$	$\to$	$\mathsf{if}(e', e_1, e_0)$	$(e \to e')$
(OSPlus)	$n_1 + n_2$	$\to$	n'	$(n' = n_1 + n_2)$
(OSMin)	$n_1 - n_2$	$\to$	n'	$(n' = n_1 \dot{-} n_2)$
(OSLeq_1)	$n_1 \leq n_2$	$\to$	1	$(n_1 \leq n_2)$
(OSLeq_0)	$n_1 \leq n_2$	$\to$	0	$(n_1 > n_2)$
$(\mathsf{OSConsL})$	$e_1\, c\, e_2$	$\to$	$e_1'\, c\, e_2$	$(e_1 \to e_1', c \in \{+, -, \leq\})$
$(\mathsf{OSConsR})$	$e_1\, c\, e_2$	$\to$	$e_1\, c\, e_2'$	$(e_2 \to e_2', c \in \{+, -, \leq\})$
$(\mathsf{OS}\beta_v)$	$(\lambda x.\, e)\, v$	$\to$	$e[v/x]$	
$(\mathsf{OSFixEv})$	$\mathsf{fix}_\sigma(v)$	$\to$	$\lambda x.\, v\, (\mathsf{fix}_\sigma(v))\, x$	(fresh x)
$(\mathsf{OSFixRed})$	$\mathsf{fix}_\sigma(e)$	$\to$	$\mathsf{fix}_\sigma(e')$	$(e \to e')$
(OSAppL)	$e_1\, e_2$	$\to$	$e_1'\, e_2$	$(e_1 \to e_1')$
(OSAppR)	$v\, e_2$	$\to$	$v\, e_2'$	$(e_2 \to e_2')$
(OSPifL)	$\mathsf{pif}(e, e_1, e_0)$	$\to$	$\mathsf{pif}(e', e_1, e_0)$	$(e \to e')$
(OSPifM)	$\mathsf{pif}(e, e_1, e_0)$	$\to$	$\mathsf{pif}(e, e_1', e_0)$	$(e_1 \to e_1')$
(OSPifR)	$\mathsf{pif}(e, e_1, e_0)$	$\to$	$\mathsf{pif}(e, e_1, e_0')$	$(e_0 \to e_0')$
(OSPif_1)	$\mathsf{pif}(n, e_1, e_0)$	$\to$	e_1	$(n > 0)$
(OSPif_0)	$\mathsf{pif}(0, e_1, e_0)$	$\to$	e_0	
$(\mathsf{OSPif}_\bot)$	$\mathsf{pif}(e, n, n)$	$\to$	n	

Figure 6.2: One-step semantics of $\mathrm{PCF_v}$ program expressions

The relation $e_1 \to e_2$ on closed expressions e_1, e_2 denotes that e_1 reduces to e_2 in a single step. Again, call-by-value demands that all arguments in an application are evaluated as far as they can—until they are values—before the application itself can be evaluated.

Many of these one-step semantics rules could be replaced by a single reduction-in-context rule. Compared to Sieber (1990), we have added (OSFixRed), because we do not treat fix_σ like a value (and thus (OSAppR) cannot apply to reduction-in-context of the argument of fix_σ). Even with this addition, the reflexive, transitive closure $e \to^* v$ captures precisely $e \Downarrow v$, as demonstrated by Lemma 6.6 and 6.7.

6.1.1.3 Denotational semantics

The last $\mathrm{PCF_v}$ semantics that we use is a *denotational semantics*, where meaning is inductively assigned to subexpressions.

Let $\eta^\sigma : V^\sigma \to C^\sigma$ be an inclusion map, for all sorts σ. Let $\mathrm{app}^{\sigma,\tau} : C^{\sigma\to\tau} \times C^\sigma \to C^\tau$ be defined by

$$\mathrm{app}^{\sigma,\tau}(x, y) := \begin{cases} \bot & \text{if } x = \bot \text{ or } y = \bot \\ x\, y & \text{O/W} \end{cases}$$

for all $x \in C^{\sigma\to\tau}$ and $y \in C^\sigma$.

The denotational semantics of expression $\Delta \vdash e : \sigma$ under some valuation $\rho \in V^\Delta$, written $\|e : \sigma\|(\rho) \in C^\sigma$, is given in Figure 6.3, where $c \in \{+, 1, \leq\}$, $\sigma' = \sigma \to \tau$, and $\mathrm{lfp}_{\sigma'}(h)$ is a shorthand for the least fixpoint of h, i.e. $\bigsqcup\{h^n(\bot_{V^{\sigma'}}) \mid n \in \omega\}$. Note that $\mathrm{lfp}_{\sigma'}$ exists in the pointed dcpo $V^{\sigma'} \Rightarrow_c V^{\sigma'}$.

Lemma 6.1. *For all sorts σ, τ, the maps $\eta^\sigma : V^\sigma \to C^\sigma$ and $\mathrm{app}^{\sigma,\tau} : C^{\sigma\to\tau} \times C^\sigma \to C^\tau$ are continuous.*

Proof. Continuity of η^σ is straightforward from the fact that C^σ is V^σ augmented with a least element.

Let $x_1, x_2 \in C^{\sigma\to\tau}$ such that $x_1 \sqsubseteq x_2$. Let $y_1, y_2 \in C^\sigma$ such that $y_1 \sqsubseteq y_2$. If $x_1 = \bot$ or $y_1 = \bot$, then $\mathrm{app}^{\sigma,\tau}(x_1, y_1) = \bot$ and the claim is trivial, so suppose both x_1 and y_1

$$
\begin{aligned}
\|n\|(\rho) &= \eta^{\iota}(n) \\
\|x : \sigma\|(\rho) &= \eta^{\sigma}(\rho(x)) \\
\|c^{\iota\to\iota\to\iota}\|(\rho) &= \eta^{\iota\to\iota\to\iota}(\lambda x \in V^{\iota}.\, \lambda y \in V^{\iota}.\, x\, c\, y) \\
\|e_1^{\sigma\to\tau}\, e_2^{\sigma}\|(\rho) &= \mathrm{app}^{\sigma,\tau}(\|e_1\|(\rho), \|e_2\|(\rho)) \\
\|\lambda x : \sigma.\, e : \tau\|(\rho) &= \eta^{\sigma\to\tau}(\lambda d \in V^{\sigma}.\, \|e : \tau\|(\rho[x \mapsto d])) \\
\|\mathsf{fix}_{\sigma'}\|(\rho) &= \eta^{(\sigma'\to\sigma')\to\sigma'}(\lambda f \in V^{\sigma'\to\sigma'}.\, \eta^{\sigma'}(\mathrm{lfp}_{\sigma'}(\lambda g \in V^{\sigma'}.\, \lambda v \in V^{\sigma}. \\
&\qquad\qquad \mathrm{app}^{\sigma,\tau}(f\, g, \eta^{\sigma}(v)))))
\end{aligned}
$$

$$
\|\mathsf{if}_{\sigma}(s, t_1, t_0)\|(\rho) = \begin{cases} \|t_1 : \sigma\|(\rho) & \text{if } \|s\|(\rho) > 0 \\ \|t_0 : \sigma\|(\rho) & \text{if } \|s\|(\rho) = 0 \\ \bot & \text{if } \|s\|(\rho) = \bot \end{cases}
$$

$$
\|\mathsf{pif}(s, t_1, t_0)\|(\rho) = \begin{cases} \|t_1 : \iota\|(\rho) & \text{if } \|s\|(\rho) > 0 \\ \|t_0 : \iota\|(\rho) & \text{if } \|s\|(\rho) = 0 \\ n & \text{if } \|t_1 : \iota\|(\rho) = \|t_0 : \iota\|(\rho) = n \\ \bot & \text{O/W} \end{cases}
$$

Figure 6.3: Denotational semantics of $\mathrm{PCF_v}$ program expressions

are convergent elements.

$$
\begin{aligned}
\mathrm{app}^{\sigma,\tau}(x_1, y_1) &= x_1\, y_1 \\
&\sqsubseteq x_1\, y_2 \qquad x_1 \text{ mon} \\
&\sqsubseteq x_2\, y_2 \qquad x_1 \sqsubseteq x_2 \\
&= \mathrm{app}^{\sigma,\tau}(x_2, y_2)
\end{aligned}
$$

Let $X \subseteq C^{\sigma\to\tau}$ and $Y \subseteq C^{\sigma}$ be directed sets. Similar to before, the claim is trivial if $\bigsqcup X = \bot$ or $\bigsqcup Y = \bot$. And, because a directed set with at least two elements is still directed with the same lub if we remove its least argument, we assume WLOG that neither X nor Y contains $\bot$.

$$
\begin{aligned}
\mathrm{app}^{\sigma,\tau}\left(\bigsqcup X, \bigsqcup Y\right) &= \left(\bigsqcup X\right)\left(\bigsqcup Y\right) \\
&= \bigsqcup_{x\in X} x \left(\bigsqcup Y\right) \qquad \text{Prop 2.8} \\
&= \bigsqcup_{x\in X} \bigsqcup_{y\in Y} x\, y \qquad \text{each } x \text{ cont} \\
&= \bigsqcup_{x\in X} \bigsqcup_{y\in Y} \mathrm{app}^{\sigma,\tau}(x, y)
\end{aligned}
$$

□

The following lemma shows that the denotational semantics are well-defined with respect to the semantic domain.

Lemma 6.2. *For all expressions* $\Delta \vdash e : \sigma$ *and* $\rho \in V^\Delta$,

(i) $\|e : \sigma\|(\rho) \in C^\sigma$, *and*

(ii) $\|e : \sigma\|$ *is continuous.*

Proof. By induction on the structure of $e : \sigma$.

Case $e = n$**.** Because $n \in V^\iota$, $\|n\|(\rho) = \eta^\iota(n) \in C^\iota$. The semantics is invariant over valuations, so (ii) is trivially satisfied.

Case $e = x$**.** Because $\rho(x : \sigma) \in V^\sigma$, $\|x : \sigma\|(\rho) = \eta^\sigma(\rho(x : \sigma)) \in C^\sigma$. Let $\rho_1, \rho_2 \in V^\Delta$ such that $\rho_1 \sqsubseteq \rho_2$.

$$\|x : \sigma\|(\rho_1) = \eta^\sigma(\rho_1(x : \sigma)) \sqsubseteq \eta^\sigma(\rho_2(x : \sigma)) = \|x : \sigma\|(\rho_2) \qquad \text{Lem 6.1}$$

Let $\mathrm{P} \subseteq V^\Delta$ be a directed set.

$$\begin{aligned}
\|x : \sigma\| \left(\bigsqcup \mathrm{P}\right) &= \eta^\sigma \left(\left(\bigsqcup \mathrm{P}\right)(x : \sigma)\right) \\
&= \eta^\sigma \left(\bigsqcup_{\rho \in \mathrm{P}} \rho(x : \sigma)\right) && \text{Prop 2.8} \\
&= \eta^\sigma \left(\bigsqcup_{\rho \in \mathrm{P}} \|x : \sigma\|(\rho)\right) \\
&= \bigsqcup_{\rho \in \mathrm{P}} \eta^\sigma \left(\|x : \sigma\|(\rho)\right) && \text{Lem 6.1}
\end{aligned}$$

Case $e = c^{\iota \to \iota \to \iota}$**.** It is clear that $\lambda x\, y.\, x\, c\, y \in V^{\iota \to \iota \to \iota}$, due to its inputs from V^ι (which is a discrete poset, so the continuous function space $V^\iota \Rightarrow_c B$ coincides with the full function space $V^\iota \Rightarrow B$ for all dcpos B). It follows that $\|c^{\iota \to \iota \to \iota}\|(\rho) = \eta^{\iota \to \iota \to \iota}(\lambda x y.\, x\, c\, y)$ is an element of $C^{\iota \to \iota \to \iota}$. The semantics is invariant over valuations, so (ii) is trivially satisfied.

Case $e = e_1\, e_2$**.** By the IH, $\|e_1 : \sigma \to \tau\|(\rho) \in C^{\sigma \to \tau}$ and $\|e_2 : \sigma\|(\rho) \in C^\sigma$, so:

$$\|e_1\, e_2\|(\rho) = \mathrm{app}^{\sigma \to \tau}(\|e_1\|(\rho), \|e_2\|(\rho)) \in C^\tau.$$

Let $\rho_1, \rho_2 \in V^\Delta$ such that $\rho_1 \sqsubseteq \rho_2$.

$$\begin{aligned}
\|e_1\, e_2\|(\rho_1) &= \mathrm{app}^{\sigma\to\tau}(\|e_1\|(\rho_1), \|e_2\|(\rho_1)) \\
&\sqsubseteq \mathrm{app}^{\sigma\to\tau}(\|e_1\|(\rho_2), \|e_2\|(\rho_2)) && \text{IH, Lem 6.1} \\
&= \|e_1\, e_2\|(\rho_2)
\end{aligned}$$

Let $\mathrm{P} \subseteq V^\Delta$ be a directed set.

$$\begin{aligned}
\|e_1\, e_2\| \left(\bigsqcup \mathrm{P}\right) &= \mathrm{app}^{\sigma\to\tau} \left(\|e_1\| \left(\bigsqcup \mathrm{P}\right), \|e_2\| \left(\bigsqcup \mathrm{P}\right)\right) \\
&= \mathrm{app}^{\sigma\to\tau} \left(\bigsqcup_{\rho'\in\mathrm{P}} \|e_1\|(\rho'), \bigsqcup_{\rho\in\mathrm{P}} \|e_2\|(\rho)\right) && \text{IH} \\
&= \bigsqcup_{\rho,\rho'\in\mathrm{P}} \mathrm{app}^{\sigma\to\tau}(\|e_1\|(\rho'), \|e_2\|(\rho)) && \text{Lem 6.1} \\
&= \bigsqcup_{\rho\in\mathrm{P}} \mathrm{app}^{\sigma\to\tau}(\|e_1\|(\rho), \|e_2\|(\rho)) && \text{Prop 2.10} \\
&= \bigsqcup_{\rho\in\mathrm{P}} \|e_1\, e_2\|(\rho)
\end{aligned}$$

Case $e = \lambda x : \sigma.\, e' : \tau$**.** By the induction hypothesis, $\|e' : \tau\|(\rho[x \mapsto d]) \in C^\tau$ for all $d \in V^\tau$. Clearly, $\lambda d \in V^\sigma.\, \|e' : \tau\|(\rho[x \mapsto d])$ is an element of $V^\sigma \Rightarrow C^\tau$. It remains to show that it is continuous. Let $x_1, x_2 \in V^\sigma$ such that $x_1 \sqsubseteq x_2$.

$$\begin{aligned}
(\lambda d \in V^\sigma.\, \|e' : \tau\|(\rho[x \mapsto d]))\, x_1 &= \|e' : \tau\|(\rho[x \mapsto x_1]) \\
&\sqsubseteq \|e' : \tau\|(\rho[x \mapsto x_2]) && \text{IH} \\
&= (\lambda d \in V^\sigma.\, \|e' : \tau\|(\rho[x \mapsto d]))\, x_2
\end{aligned}$$

Let $D \subseteq V^\sigma$ be a directed set.

$$\begin{aligned}
(\lambda d \in V^\sigma.\, \|e' : \tau\|(\rho[x \mapsto d])) \left(\bigsqcup D\right) &= \|e' : \tau\| \left(\rho \left[x \mapsto \bigsqcup D\right]\right) \\
&= \bigsqcup_{d'\in D} \|e' : \tau\|(\rho[x \mapsto d']) && \text{IH} \\
&= \bigsqcup_{d'\in D} (\lambda d \in V^\sigma.\, \|e' : \tau\|(\rho[x \mapsto d]))\, d'
\end{aligned}$$

We conclude that $\lambda d \in V^\sigma.\, \|e' : \tau\|(\rho[x \mapsto d]) \in V^{\sigma\to\tau}$, and thus $\|\lambda x : \sigma.\, e' : \tau\|(\rho) \in C^{\sigma\to\tau}$. Let $\rho_1, \rho_2 \in V^\Delta$ such that $\rho_1 \sqsubseteq \rho_2$.

$$\begin{aligned}
\|\lambda x : \sigma.\, e' : \tau\|(\rho_1) &= \eta^{\sigma\to\tau}(\lambda d \in V^\sigma.\, \|e : \tau\|(\rho_1[x \mapsto d])) \\
&\sqsubseteq \eta^{\sigma\to\tau}(\lambda d \in V^\sigma.\, \|e : \tau\|(\rho_2[x \mapsto d])) && \text{IH, Lem 6.1} \\
&= \|\lambda x : \sigma.\, e' : \tau\|(\rho_2)
\end{aligned}$$

Let $\mathrm{P} \subseteq V^{\Delta}$ be a directed set.

$$\begin{aligned}
\|\lambda x : \sigma.\, e' : \tau\| \left(\bigsqcup \mathrm{P}\right) &= \eta^{\sigma\to\tau}\left(\lambda d \in V^{\sigma}.\, \|e'\|\left(\left(\bigsqcup \mathrm{P}\right)[x \mapsto d]\right)\right) \\
&= \eta^{\sigma\to\tau}\left(\lambda d \in V^{\sigma}.\, \|e'\|\left(\bigsqcup_{\rho\in\mathrm{P}} \rho[x \mapsto d]\right)\right) \\
&= \eta^{\sigma\to\tau}\left(\lambda d \in V^{\sigma}.\, \bigsqcup_{\rho\in\mathrm{P}} \|e'\|(\rho[x \mapsto d])\right) && \text{IH} \\
&= \eta^{\sigma\to\tau}\left(\bigsqcup_{\rho\in\mathrm{P}} \lambda d \in V^{\sigma}.\, \|e'\|(\rho[x \mapsto d])\right) \\
&= \bigsqcup_{\rho\in\mathrm{P}} \eta^{\sigma\to\tau}(\lambda d \in V^{\sigma}.\, \|e'\|(\rho[x \mapsto d])) && \text{Lem 6.1} \\
&= \bigsqcup_{\rho\in\mathrm{P}} \|\lambda x : \sigma.\, e' : \tau\|(\rho)
\end{aligned}$$

Case $e = \mathsf{fix}_{\sigma\to\tau}$**.** Recall that:

$$\|\mathsf{fix}_{\sigma'}\|(\rho) = \eta^{(\sigma'\to\sigma')\to\sigma'}(\lambda f \in V^{\sigma'\to\sigma'}.\, \eta^{\sigma'}(\mathrm{lfp}_{\sigma'}(\lambda g \in V^{\sigma'}.\, \lambda v \in V^{\sigma}.\, \mathrm{app}^{\sigma,\tau}(f\, g, \eta^{\sigma}(v)))))$$

For (i), we first prove that the argument to $\mathrm{lfp}_{\sigma'}$, let us call it †, is an element of $V^{\sigma'} \Rightarrow_c V^{\sigma'}$ for all $f \in V^{\sigma'\to\sigma'}$. It is straightforward to verify that it is of the right type, since $\mathrm{app}^{\sigma,\tau}(f\, g, \eta^{\sigma}(v)) \in C^{\tau}$ and $V^{\sigma'} = [V^{\sigma} \Rightarrow_c C^{\tau}]$. Continuity follows from f, η^{σ}, and $\mathrm{app}^{\sigma,\tau}$ being continuous (Lemma 6.1).

Thus, $\mathrm{lfp}_{\sigma'}(\dagger) \in V^{\sigma'}$ and $\eta^{\sigma'}(\mathrm{lfp}_{\sigma'}(\dagger)) \in C^{\sigma'}$. It remains to show that $\lambda f.\, \eta^{\sigma'}(\mathrm{lfp}_{\sigma'}(\dagger)))$ is continuous. This follows from continuity of $\eta^{\sigma'}$, $\mathrm{lfp}_{\sigma'}$, $\mathrm{app}^{\sigma,\tau}$, and η^{σ}. We conclude that $\|\mathsf{fix}_{\sigma'}\|(\rho) \in C^{(\sigma'\to\sigma')\to\sigma'}$.

Part (ii) of the claim is trivial, because $\|\mathsf{fix}_{\sigma'}\|(\rho)$ does not depend on ρ.

Case $e = \mathsf{if}_{\sigma}(s, t_1, t_0)$**.** All three cases are elements of C^{σ}, partly thanks to the IH. Let $\rho_1, \rho_2 \in V^{\Delta}$ such that $\rho_1 \sqsubseteq \rho_2$. By the IH, $\|s\|$ is monotone, so $\|s\|(\rho_1) \sqsubseteq \|s\|(\rho_2)$, which means that either $\|s\|(\rho_1) = \bot$ or $\|s\|(\rho_1) = \|s\|(\rho_2) = n$. In the former case:

$$\|\mathsf{if}_{\sigma}(s, t_1, t_0)\|(\rho_1) = \bot \sqsubseteq \|\mathsf{if}_{\sigma}(s, t_1, t_0)\|(\rho_2)$$

In the latter case:

$$\begin{aligned}
\|\mathsf{if}_{\sigma}(s, t_1, t_0)\|(\rho_1) &= \|t_i\|(\rho_1) && \text{for } i = (n > 0) \\
&\sqsubseteq \|t_i\|(\rho_2) && \text{IH} \\
&= \|\mathsf{if}_{\sigma}(s, t_1, t_0)\|(\rho_2)
\end{aligned}$$

Let $\mathrm{P} \subseteq V^{\Delta}$ be a directed set. Thanks to flat dcpo C^{ι}, the set

$$\{\|s\|(\rho) \mid \rho \in \mathrm{P}\}$$

is either a singleton set or $\{\bot, n\}$ for some n. Either $\|s\|\left(\bigsqcup \mathrm{P}\right) = \bigsqcup_{\rho\in\mathrm{P}}\|s\|(\rho) = \bot$ or $\|s\|\left(\bigsqcup \mathrm{P}\right) = \bigsqcup_{\rho\in\mathrm{P}}\|s\|(\rho) = n$. In the former case:

$$\|\mathsf{if}_{\sigma}(s,t_1,t_0)\|\left(\bigsqcup \mathrm{P}\right) = \bot = \bigsqcup_{\rho\in\mathrm{P}}\|\mathsf{if}_{\sigma}(s,t_1,t_0)\|(\rho)$$

In the latter case:

$$\begin{aligned}
\|\mathsf{if}_{\sigma}(s,t_1,t_0)\|\left(\bigsqcup \mathrm{P}\right) &= \|t_i : \sigma\|\left(\bigsqcup \mathrm{P}\right) \qquad \text{for } i = (n > 0)\\
&= \bigsqcup_{\rho\in\mathrm{P}}\|t_i : \sigma\|(\rho) \qquad \text{IH}\\
&= \bigsqcup_{\rho\in\mathrm{P}}\|\mathsf{if}_{\sigma}(s,t_1,t_0)\|(\rho)
\end{aligned}$$

Case $e = \mathsf{pif}(s,t_1,t_0)$**.** All four cases are elements of C^{ι}, partly thanks to the IH. Let $\rho_1, \rho_2 \in V^{\Delta}$ such that $\rho_1 \sqsubseteq \rho_2$. By the IH, $\|s\|$ is monotone, so $\|s\|(\rho_1) \sqsubseteq \|s\|(\rho_2)$, which means that either $\|s\|(\rho_1) = \bot$ or $\|s\|(\rho_1) = \|s\|(\rho_2) = n$. The former case is the one that differs w.r.t. $\mathsf{if}_{\iota}(s,t_1,t_0)$, so suppose that $\|s\|(\rho_1) = \bot$. If $\|s\|(\rho_2) = \bot$, then:

$$\begin{aligned}
\|\mathsf{pif}(s,t_1,t_0)\|(\rho_1) &= \begin{cases} \|t_1 : \iota\|(\rho_1) & \text{if } \|t_1 : \iota\|(\rho_1) = \|t_0 : \iota\|(\rho_1) = m \\ \bot & \text{O/W} \end{cases}\\
&\sqsubseteq \begin{cases} \|t_1 : \iota\|(\rho_1) & \text{if } \|t_1 : \iota\|(\rho_2) = \|t_0 : \iota\|(\rho_2) = m \\ \bot & \text{O/W} \end{cases} \qquad \text{IH}\\
&\sqsubseteq \begin{cases} \|t_1 : \iota\|(\rho_2) & \text{if } \|t_1 : \iota\|(\rho_2) = \|t_0 : \iota\|(\rho_2) = m \\ \bot & \text{O/W} \end{cases} \qquad \text{IH}\\
&= \|\mathsf{pif}(s,t_1,t_0)\|(\rho_2)
\end{aligned}$$

If $\|s\|(\rho_2) = n$, then:

$$\begin{aligned}
\|\mathsf{pif}(s,t_1,t_0)\|(\rho_1) &= \begin{cases} \|t_1 : \iota\|(\rho_1) & \text{if } \|t_1 : \iota\|(\rho_1) = \|t_0 : \iota\|(\rho_1) = m \\ \bot & \text{O/W} \end{cases}\\
&\sqsubseteq \begin{cases} \|t_1 : \iota\|(\rho_1) & \text{if } \|t_1 : \iota\|(\rho_2) = \|t_0 : \iota\|(\rho_2) = m \\ \bot & \text{O/W} \end{cases} \qquad \text{IH}\\
&\sqsubseteq \begin{cases} \|t_1 : \iota\|(\rho_2) & \text{if } \|t_1 : \iota\|(\rho_2) = \|t_0 : \iota\|(\rho_2) = m \\ \bot & \text{O/W} \end{cases} \qquad \text{IH}\\
&\sqsubseteq \begin{cases} \|t_i : \iota\|(\rho_2) & \text{if } \|s\|(\rho_2) = n \text{ and } i = (n > 0) \\ \|t_1 : \iota\|(\rho_2) & \text{if } \|t_1 : \iota\|(\rho_2) = \|t_0 : \iota\|(\rho_2) = m \\ \bot & \text{O/W} \end{cases}\\
&= \|\mathsf{pif}(s,t_1,t_0)\|(\rho_2)
\end{aligned}$$

Let $\mathrm{P} \subseteq V^{\Delta}$ be a directed set. Thanks to flat dcpo C^{ι}, the set

$$\{\|s\|(\rho) \mid \rho \in \mathrm{P}\}$$

is either a singleton set or $\{\bot, n\}$ for some n, and similarly for t_1 and t_0. The claim then follows straightforwardly from the IH and monotonicity. □

6.1.1.4 Equivalence

The following theorems and lemma establish an equivalence between the three semantics for $\mathrm{PCF_v}$ that we have presented.

Theorem 6.3 (Adequacy, Plotkin, 1977, Sieber, 1990)**.** *Given* $\vdash s : \sigma$*,*

(1) if $s \Downarrow v$ *then* $\|\vdash s : \sigma\| = \|\vdash v : \sigma\|$*, and*

(2) $s\Downarrow$ *if and only if* $\|s\| \neq \bot$*.*

The converse of adequacy is *computational soundness*, which does not hold for $\mathrm{PCF_v}$ without the *parallel if* operator (under the standard Scott-continuous model).

Theorem 6.4 (Computational soundness, Sieber, 1990)**.** *If* $\|\vdash e : \iota\| = \|\vdash n : \iota\|$*, then* $e \Downarrow n$*.*

Without the parallel if operator, pif, there exist elements in the semantic domain that cannot be captured by an expression. Plotkin (1977) first established the relationship between computational soundness and definability of finite elements: for a model to be fully abstract (i.e. for both adequacy and computational soundness to hold, Hyland and Ong, 2000), we require definability of finite elements.

Proposition 6.5 (Definability of finite elements, Sieber, 1990)**.** *For each finite/compact element* $t \in V^{\sigma}$*, there exists a closed fixpoint-free* PCF_v *expression* e *such that* $\|e\| = t$*.*

Computational soundness holds only at ground sort ι, not at higher sorts. To see that this is the case, consider closed values $\lambda n.\, 5+1$ and $\lambda n.\, 6$. Both expressions have $\lambda n.\, 6$ as denotational semantics, but converge to themselves operationally.

The following two lemmas establish an equivalence between the (big-step) operational semantics and the one-step semantics.

Lemma 6.6. *For closed expression e and value v,*

$$e \Downarrow v \quad \Rightarrow \quad e \to^* v.$$

Proof. By induction on the operational semantics.

Case v. Then, $v \Downarrow v$ and $v \to^* v$.

Case $e_1 \leq e_2$. Let $e_1 \leq e_2 \Downarrow n$. This means that $e_1 \Downarrow n_1$ and $e_2 \Downarrow n_2$ for some $n_1, n_2 \in \mathbb{N}$, and $n \in [0, 1]$. By the induction hypothesis, $e_1 \to^* n_1$ and $e_2 \to^* n_2$. Combined with (OSConsL), (OSConsR), and (OSLeq_1) if $n_1 \leq n_2$ and (OSLeq_0) otherwise, this gives us

$$e_1 \leq e_2 \to^* e_1 \leq n_2 \to^* n_1 \leq n_2 \to n.$$

Case $e_1 + e_2$. Let $e_1 + e_2 \Downarrow n$. This means that $e_1 \Downarrow n_1$ and $e_2 \Downarrow n_2$ for some $n_1, n_2 \in \mathbb{N}$ such that $n_1 + n_2 = n$. By the induction hypothesis, $e_1 \to^* n_1$ and $e_2 \to^* n_2$. Combined with (OSConsL), (OSConsR), and (OSPlus), this gives us

$$e_1 + e_2 \to^* e_1 + n_2 \to^* n_1 + n_2 \to n.$$

The case for $e_1 - e_2$ is analogous using (OSMin).

Case $\mathsf{if}(e_1, e_2, e_3)$. Let $\mathsf{if}(e_1, e_2, e_3) \Downarrow v$. This means that $e_1 \Downarrow n$ and $e_i \Downarrow v$ for some $i \in \{2, 3\}$. By the induction hypothesis, $e_1 \to^* n$ and $e_i \to^* v$. Combined with (OSIf), and (OSIf_1) if $n > 0$ and (OSIf_0) otherwise, this gives us

$$\mathsf{if}(e_1, e_2, e_3) \to^* \mathsf{if}(n, e_2, e_3) \to e_i \to^* v.$$

Case $e_1\, e_2$. Let $e_1\, e_2 \Downarrow v$. This means that $e_1 \Downarrow \lambda x.\, e$, $e_2 \Downarrow w$, and $e[w/x] \Downarrow v$. By the induction hypothesis, $e_1 \to^* \lambda x.\, e$, $e_2 \to^* w$, and $e[w/x] \to^* v$. Combined with (OSAppL), (OSAppR), and ($\mathsf{OS}\beta_v$), this gives us

$$e_1\, e_2 \to^* (\lambda x.\, e)\, e_2 \to^* (\lambda x.\, e)\, w \to e[w/x] \to^* v.$$

Case $\mathsf{fix}\, e_1$. Let $\mathsf{fix}\, e_1 \Downarrow \lambda x.\, v\, (\mathsf{fix}\, v)\, x$. This means that $e_1 \Downarrow v$. By the induction hypothesis, $e_1 \to^* v$. Combined with (OSFixRed) and (OSFixEv), this gives us

$$\mathsf{fix}\, e_1 \to^* \mathsf{fix}\, v \to \lambda x.\, v\, (\mathsf{fix}\, v)\, x.$$

Case $\mathsf{pif}(e_1, e_2, e_3)$. Let $\mathsf{pif}(e_1, e_2, e_3) \Downarrow n$. We proceed by case analysis on e_1. If $e_1 \Downarrow m$, then $e_i \Downarrow n$ with $i = 2$ if $m > 0$ and $i = 3$ if $m = 0$. By the IH, $e_1 \to^* m$

and $e_i \to^* n$. Combined with (OSPifL), (OSPif_1) if $m > 0$ and (OSPif_0) if $m = 0$, this gives us

$$\mathsf{pif}(e_1, e_2, e_3) \to^* \mathsf{pif}(m, e_2, e_3) \to e_i \to^* n.$$

If $e \Uparrow$, then $e_2 \Downarrow n$ and $e_3 \Downarrow n$. By the IH, $e_2 \to^* n$ and $e_3 \to^* n$. Combined with (OSPifM), (OSPifR), and ($\mathsf{OSPif}_\bot$), this gives us

$$\mathsf{pif}(e_1, e_2, e_3) \to^* \mathsf{pif}(e_1, n, e_3) \to^* \mathsf{pif}(e_1, n, n) \to n.$$

□

Lemma 6.7. *For closed expression e and value v,*

$$e \to^* v \quad \Rightarrow \quad e \Downarrow v.$$

Proof. It suffices to show that $e \to e'$ and $e' \Downarrow v$ implies $e \Downarrow v$. We proceed by case analysis on $e \to e'$, where we distinguish between base case rules (those that do not depend on any $e'' \to e'''$) and inductive rules (those that do). Let us start with the base case rules.

Case (OSIf_1). Suppose that $e_1 \Downarrow v$. Trivially, $n \Downarrow n$. The operational semantics then gives us $\mathsf{if}(n, e_1, e_0) \Downarrow v$.

The case for (OSIf_0) is similar.

Case (OSPlus). The RHS is a value, so $n' \Downarrow n' = v$. Trivially, $n_1 \Downarrow n_1$ and $n_2 \Downarrow n_2$. The operational semantics then gives us $n_1 + n_2 \Downarrow v$, as required.

The case for (OSMin) is similar.

Case (OSLeq_1). The RHS is a value, so $1 \Downarrow 1 = v$. Trivially, $n_1 \Downarrow n_1$ and $n_2 \Downarrow n_2$. The operational semantics then gives us $n_1 \leq n_2 \Downarrow v$, as required.

The case for (OSLeq_0) is similar.

Case ($\mathsf{OS}\beta_v$). Suppose that $e[v'/x] \Downarrow v$. Trivially, $\lambda x.\, e \Downarrow \lambda x.\, e$ and $v' \Downarrow v'$. The operational semantics then gives us $(\lambda x.\, e)\, v' \Downarrow v$, as required.

Case (OSFixEv). The RHS is a value, so $\lambda x.\, v'\, (\mathsf{fix}_\sigma(v'))\, x \Downarrow \lambda x.\, v'\, (\mathsf{fix}_\sigma(v'))\, x$. Trivially, $v' \Downarrow v'$. The operational semantics then gives us $\mathsf{fix}_\sigma(v') \Downarrow v$, as required.

Case (OSPif_1). Suppose that $e_2 \Downarrow v$. Trivially, $n \Downarrow n$. The operational semantics then gives us $\mathsf{pif}(n, e_2, e_3) \Downarrow v$, as required.

The case for (OSPif_0) is similar.

Case ($\mathsf{OSPif}_\perp$). The RHS is a value, so $n \Downarrow n = v$. The operational semantics then gives us $\mathsf{pif}(e_1, n, n) \Downarrow v$, as required.

For the inductive rules $e \to e'$, suppose that the claim holds for dependent rules $e'' \to e'''$. These cases are straightforward reductions-in-context.

Case (OSIf). Suppose that $\mathsf{if}(e', e_1, e_0) \Downarrow v$. This means that $e' \Downarrow n$ for some n. Since we have $e \to e'$, the IH gives us $e \Downarrow n$. It follows from the operational semantics that $\mathsf{if}(e, e_1, e_0) \Downarrow v$.

Case (OSConsL). Suppose that $e_1' \, c \, e_2 \Downarrow n = v$. This means that $e_1' \Downarrow n_1$ and $e_2 \Downarrow n_2$. Since we have $e_1 \to e_1'$, the IH gives us $e_1 \Downarrow n_1$. It follows from the operational semantics that $e_1 \, c \, e_2 \Downarrow v$.

The case for (OSConsR) is similar.

Case (OSFixRed). Suppose that $\mathsf{fix}_\sigma(e') \Downarrow v$. This means that $e' \Downarrow v'$. Since we have $e \to e'$, the IH gives us $e \Downarrow v'$. It follows from the operational semantics that $\mathsf{fix}_\sigma(e) \Downarrow v$.

Case (OSAppL). Suppose that $e_1' \, e_2 \Downarrow v$. This means that $e_1' \Downarrow \lambda x.\, e$, $e_2 \Downarrow w$, and $e[w/x] \Downarrow v$. Since we have $e_1 \to e_1'$, the IH gives us $e_1 \Downarrow \lambda x.\, e$. It follows from the operational semantics that $e_1 \, e_2 \Downarrow v$.

The case for (OSAppR) is similar.

Case (OSPifL). Suppose that $\mathsf{pif}(e_1', e_2, e_3) \Downarrow n' = v$. Consider the subcase where there exists n such that $e_1' \Downarrow n$. Since we have $e_1 \to e_1'$, the IH gives us $e_1 \Downarrow n$. It then follows from the operational semantics that $\mathsf{pif}(e_1, e_2, e_3) \Downarrow v$. Now consider the subcase where $e_1' \Uparrow$. Then, $e_2 \Downarrow n'$ and $e_3 \Downarrow n'$. Again, this gives us $\mathsf{pif}(e_1, e_2, e_3) \Downarrow v$ by the operational semantics.

Case (OSPifM). Suppose that $\mathsf{pif}(e_1, e_2', e_3) \Downarrow n' = v$. We distinguish three exhaustive subcases.

Suppose that $e_1 \Downarrow n > 0$ and $e_2' \Downarrow n'$. Since we have $e_2 \to e_2'$, the IH gives us $e_2 \Downarrow n'$. The operational semantics then give us $\mathsf{pif}(e_1, e_2, e_3) \Downarrow v$.

Suppose that $e_1 \Downarrow 0$ and $e_3 \Downarrow n'$. The result is immediate.

Suppose that $e_1 \Uparrow$, $e_2' \Downarrow n'$, and $e_3 \Downarrow n'$. Since we have $e_2 \to e_2'$, the IH gives us $e_2 \Downarrow n'$. The result follows from the operational semantics.

The case for (OSPifR) is similar. □

Adequacy and computational soundness provide an equivalence between the denotational semantics and the operational semantics for sort ι (Theorem 6.3 and 6.4). Lemma 6.6 and 6.7 establish an equivalence between the operational semantics and the one-step semantics of $\mathrm{PCF_v}$. By transitivity, this gives us the following result.

Corollary 6.8 (Equivalence of $\mathrm{PCF_v}$ semantics)**.** *For closed expression $e : \iota$ and value $n : \iota$,*

$$\|e\| = \|n\| \quad \Leftrightarrow \quad e \Downarrow n \quad \Leftrightarrow \quad e \to^* n.$$

6.2 Higher-order Hoare triples

We make our usual assumptions about higher-order logic (HoL). Please refer to Chapter 2.2 for definitions. Notably, we assume that all sorts are generated by:

$$\sigma ::= \iota \mid o \mid \sigma_1 \to \sigma_2$$

These sorts are interpreted by the usual continuous sort frame of (Scott-)continuous functions, where the sort ι of individuals is interpreted as the set of natural numbers (regarded as a discrete poset) and the sort o of propositions as the complete lattice of boolean truth values:

$$\mathcal{C}[\![\iota]\!] := \langle \mathbb{N}, = \rangle, \qquad \mathcal{C}[\![o]\!] := \mathbb{B}, \qquad \mathcal{C}[\![\sigma_1 \to \sigma_2]\!] := \mathcal{C}[\![\sigma_1]\!] \Rightarrow_c \mathcal{C}[\![\sigma_2]\!].$$

We may write $\top$ for the HoL formula true to save space.

The choice of the continuous interpretation suits the continuous semantics of $\mathrm{PCF_v}$; we need continuity to make sense of the fixpoint operator. Furthermore, finite elements can be captured by the meaning of a finite HoL term (Theorem 6.20). By continuity, this allows us to restrict our proofs to HoL terms rather semantic elements, e.g. in Theorem 6.21.

6.2.1 Syntax

A *higher-order Hoare triple* (or just *Hoare triple*) has the form

$$\{A\}\, e\, \{P\}$$

where

- $\Delta \vdash e : \sigma$ is a PCF_v expression,
- $\Gamma \cup \mathsf{Rel}^-(\Delta) \vdash A : o$ is a HoL term,
- $\Gamma \cup \mathsf{Rel}^-(\Delta) \vdash P : \mathsf{Rel}^+(\sigma)$ is a HoL term,

and $\mathsf{Rel}^-(\sigma)$ and $\mathsf{Rel}^+(\sigma)$ are defined as in Chapter 4.3, which is by induction over:

$$\mathsf{Rel}^-(\iota) := \iota \qquad \mathsf{Rel}^-(\sigma_1 \to \sigma_2) := \mathsf{Rel}^-(\sigma_1) \to \mathsf{Rel}^+(\sigma_2)$$
$$\mathsf{Rel}^+(\iota) := \iota \to o \qquad \mathsf{Rel}^+(\sigma_1 \to \sigma_2) := \mathsf{Rel}^-(\sigma_1) \to \mathsf{Rel}^+(\sigma_2)$$

This *relational lift* of sorts extends pointwise to sort environments Δ.

Note that Δ is a sort environment of *program variables*, i.e. $\Delta \supseteq \mathsf{FV}(e)$. The (relational) sort environment Γ, on the other hand, contains *ghost variables* that may occur in the pre- and postcondition but not in the program expression e. Thus, $\mathsf{dom}(\Delta) \cap \mathsf{dom}(\Gamma) = \emptyset$.

We assume the existence of *relational clone* $x' : \mathsf{Rel}^-(\sigma) \in \mathsf{Rel}^-(\Delta)$ for each $x : \sigma \in \Delta$.

Useful program logics can be obtained from this general system through restrictions on the syntax. For instance, if we restrict the precondition A to definite clauses and the postcondition P to goal terms, then we recover the higher-order constrained Horn clause fragment of HoL, as shown in Section 6.4.

6.2.2 Lift from program expressions to property terms

We define a transformation of PCF_v program expressions $\Delta \vdash e : \sigma$ to *property terms* $\mathsf{Rel}^-(\Delta) \vdash \lceil e \rceil : \mathsf{Rel}^+(\sigma)$ of higher-order logic in Figure 6.4. This transformation essentially captures the *strongest postcondition* of a program expression.

In this definition, x' is the relational clone of x, the sequence $\overline{z}$ of elements denotes $z_1 \dots z_n\, r$ for $z_i : \mathsf{Rel}^-(\tau_i)$ and $r : \iota$ with $\tau = \tau_1 \to \dots \to \tau_n \to \iota$, and c' denotes a HoL term relationally equivalent to c.

Our PCF_v expressions contain three constants c of sort $\iota \to \iota \to \iota$. We define relationally equivalent HoL terms $c' : \mathsf{Rel}^+(\iota \to \iota \to \iota) = \iota \to \iota \to \iota \to o$ by:

$$+' := \lambda xyz.\, (x + y = z)$$
$$-' := \lambda xyz.\, (x \dot{-} y = z)$$
$$\leq' := \lambda xyz.\, (x \leq y \wedge z = 1) \vee (x > y \wedge z = 0)$$

$$
\begin{aligned}
\lceil n \rceil &:= \lambda m.\, m = n \\
\lceil c^{\sigma_1 \to \sigma_2} \rceil &:= c' \\
\lceil x^{\iota} \rceil &:= \lambda m.\, m = x' \\
\lceil x^{\sigma_1 \to \sigma_2} \rceil &:= x' \\
\lceil s^{\iota \to \tau}\, t^{\iota} \rceil &:= \lambda \overline{z}.\, \exists x.\, \lceil t \rceil\, x \wedge \lceil s \rceil\, x\, \overline{z} \\
\lceil s^{(\sigma \to \sigma') \to \tau}\, t^{\sigma \to \sigma'} \rceil &:= \lceil s \rceil\, \lceil t \rceil \\
\lceil \lambda x^{\sigma}.\, s^{\tau} \rceil &:= \lambda x'.\, \lceil s^{\tau} \rceil \\
\lceil \mathsf{if}_{\tau}(s, t_1, t_0) \rceil &:= \lambda \overline{z}.\, (\exists x.\, x > 0 \wedge \lceil s \rceil\, x \wedge \lceil t_1 \rceil\, \overline{z}) \vee (\lceil s \rceil\, 0 \wedge \lceil t_0 \rceil\, \overline{z}) \\
\lceil \mathsf{pif}(s, t_1, t_0) \rceil &:= \lambda r.\, (\exists x.\, x > 0 \wedge \lceil s \rceil\, x \wedge \lceil t_1 \rceil\, r) \vee (\lceil s \rceil\, 0 \wedge \lceil t_0 \rceil\, r) \vee (\lceil t_1 \rceil\, r \wedge \lceil t_0 \rceil\, r) \\
\lceil \mathsf{fix}_{\sigma \to \tau}(s) \rceil &:= \mathbf{Y}_{\mathsf{Rel}^{+}(\sigma \to \tau)}\, (\lceil s \rceil)
\end{aligned}
$$

Figure 6.4: Lift from $\mathrm{PCF_v}$ program expressions to property terms of HoL

Well-sortedness of the lift, i.e. that $\lceil e : \sigma \rceil : \mathsf{Rel}^{+}(\sigma)$ for all expressions $e : \sigma$, is established in Lemma 6.14 in Section 6.3.

6.2.3 Semantics

Intuitively, a Hoare triple $\{A\}\, e\, \{P\}$ is *valid*, written $\vDash \{A\}\, e\, \{P\}$, if postcondition P is a (relational) overapproximation of the program in the presence of A.

Closed program expressions. The meaning of a Hoare triple with closed program expression $\vdash e : \sigma$ is given by

$$\vDash \{A\}\, e\, \{P\} := A \vDash \lceil e \rceil \sqsubseteq P.$$

A valid triple $\vDash \{A\}\, e\, \{P\}$ is implicitly universally quantified over its free program variables. Therefore, we consider a triple with a non-closed program expression valid if and only if it is valid under all (closed value) substitutions.

Similar to a valid triple $\vDash \{A\}\, e\, \{P\}$, we write $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$ just if the relational substitution $\mathbb{S}$ is an overapproximation of the substitution θ of the program variables.

Closed value substitutions. We extend this judgement to pairs $\langle \theta, \mathbb{S} \rangle$ of closed value substitutions θ over Δ and relational substitutions $\mathbb{S}$ over $\Gamma \cup \mathsf{Rel}^{-}(\Delta)$ in the following way. We say that $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$ just if:

- for each $x : \iota \in \Delta$, $\theta(x) = \mathbb{S}(x')$, and

- for each $x : \sigma_1 \to \sigma_2 \in \Delta$, $\vDash \{\mathbb{S}(A)\}\,\theta(x)\,\{\mathbb{S}(x')\}$, i.e. $\mathbb{S}(A) \vDash \lceil\theta(x)\rceil \sqsubseteq \mathbb{S}(x')$.

We often refer to such a pair $\langle\theta, \mathbb{S}\rangle$ as a closed value substitution (over Δ, if this is unclear from the context).

These closed values substitutions allows us to define the meaning of a Hoare triple for a general program expression.

General program expressions. Finally, we define $\vDash \{A\}\,e\,\{P\}$ for general program expression e just if, for all $\langle\theta, \mathbb{S}\rangle$,

$$\vDash \{A\}\,\theta\,\{\mathbb{S}\} \quad \Rightarrow \quad \mathbb{S}(A) \vDash \mathbb{S}(\lceil e\rceil) \sqsubseteq \mathbb{S}(P).$$

Lemma 6.9. *For all expressions $\Delta \vdash e$ and closed value substitutions $\langle\theta, \mathbb{S}\rangle$ over Δ such that $\vDash \{A\}\,\theta\,\{\mathbb{S}\}$,*

$$\mathbb{S}, A \vDash \lceil\theta(e)\rceil \sqsubseteq \lceil e\rceil.$$

Proof. Suppose $\Delta \vdash e$ and $\vDash \{A\}\,\theta\,\{\mathbb{S}\}$ for a closed value substitution $\langle\theta, \mathbb{S}\rangle$ over Δ.

Recall that this means that

- for each $x : \iota \in \Delta$, $\theta(x) = \mathbb{S}(x')$, and
- for each $x : \sigma_1 \to \sigma_2 \in \Delta$, $\vDash \{\mathbb{S}(A)\}\,\theta(x)\,\{\mathbb{S}(x')\}$, i.e. $\mathbb{S}(A) \vDash \lceil\theta(x)\rceil \sqsubseteq \mathbb{S}(x')$.

We proceed by a straightforward inductive argument on e. Suppose $\mathbb{S}(A)$. It suffices to show that $\mathcal{C}[\![\lceil\theta(e)\rceil]\!] \sqsubseteq \mathcal{C}[\![\mathbb{S}(\lceil e\rceil)]\!]$, relying on $\lceil\theta(e)\rceil$ being closed, so $\mathbb{S}(\lceil\theta(e)\rceil) = \lceil\theta(e)\rceil$.

Case $e = n$ or $e = c^{\iota\to\iota\to\iota}$. These are constant, so the result is trivial.

Case $e = x^{\iota}$. This follows from $\theta(x^\iota) = n = \mathbb{S}(x')$ for some $n \in \mathbb{N}$:

$$\begin{aligned}
\mathcal{C}[\![\lceil\theta(x^\iota)\rceil]\!] &= \mathcal{C}[\![\lceil n\rceil]\!] \\
&= \mathcal{C}[\![\lambda m.\, m = n]\!] \\
&= \mathcal{C}[\![\lambda m.\, m = \mathbb{S}(x')]\!] \\
&= \mathcal{C}[\![\mathbb{S}(\lambda m.\, m = x')]\!] \\
&= \mathcal{C}[\![\mathbb{S}(\lceil x^\iota\rceil)]\!]
\end{aligned}$$

Case $e = x^{\sigma_1\to\sigma_2}$. We obtain $\mathbb{S}(A) \vDash \lceil\theta(x)\rceil \sqsubseteq \mathbb{S}(x')$ as a consequence of $\vDash \{A\}\,\theta\,\{\mathbb{S}\}$. The result follows from $\lceil x\rceil = x'$.

The remaining (inductive) cases are simple substitutions. For proof of concept, we present the fixpoint case.

Case $e = \mathsf{fix}_\sigma(s)$**.**

$$\begin{aligned}
\mathcal{C}[\![\lceil\theta(\mathsf{fix}_\sigma(s))\rceil]\!] &= \mathcal{C}[\![\lceil\mathsf{fix}_\sigma(\theta(s))\rceil]\!] \\
&= \mathcal{C}[\![\mathbf{Y}_{\mathsf{Rel}^+(\sigma)}(\lceil\theta(s)\rceil)]\!] \\
&= \mathrm{lfp}(\mathcal{C}[\![\lceil\theta(s)\rceil)]\!]) \\
&\sqsubseteq \mathrm{lfp}(\mathcal{C}[\![\mathbb{S}(\lceil s\rceil)]\!]) \qquad \text{IH, mon of lfp} \\
&= \mathcal{C}[\![\mathbf{Y}_{\mathsf{Rel}^+(\sigma)}(\mathbb{S}(\lceil s\rceil))]\!] \\
&= \mathcal{C}[\![\mathbb{S}(\mathbf{Y}_{\mathsf{Rel}^+(\sigma)}(\lceil s\rceil))]\!] \\
&= \mathcal{C}[\![\mathbb{S}(\lceil\mathsf{fix}_\sigma(s)\rceil)]\!]
\end{aligned}$$

□

Corollary 6.10. *For all program expressions e,*

$$\vDash \{\top\}\, e\, \{\lceil e\rceil\}.$$

6.2.4 Proof system

A Hoare triple is *derivable*, written $\vdash \{A\}\, e\, \{P\}$, just if there exists a proof with $\{A\}\, e\, \{P\}$ as root, in the proof system presented in Figure 6.5.

Lemma 6.11 (Left-monotonicity)**.**

(1) If $\vdash \{B\}\, s\, \{P\}$ *and* $\vDash A \Rightarrow B$*, then* $\vdash \{A\}\, s\, \{P\}$*.*

(2) If $\vDash \{B\}\, s\, \{P\}$ *and* $\vDash A \Rightarrow B$*, then* $\vDash \{A\}\, s\, \{P\}$*.*

Proof. The former follows directly from the Conseq rule for $\vdash$. For the latter, suppose $\vDash \{B\}\, s\, \{P\}$ and $\vDash A \Rightarrow B$.

Let Γ_A, Γ_B be the ghost variables of A and B, resp. It holds that $\Gamma_B \subseteq \Gamma_A$. Take a substitution pair $\langle\theta_A, \mathbb{S}_A\rangle$ satisfying $\vDash \{A\}\, \theta_A\, \{\mathbb{S}_A\}$. It suffices to show $\mathbb{S}_A(A) \vDash \mathbb{S}_A(\lceil s\rceil) \sqsubseteq \mathbb{S}_A(P)$.

It follows from $\vDash \{A\}\, \theta_A\, \{\mathbb{S}_A\}$ that $\vDash \{B\}\, \theta_B\, \{\mathbb{S}_B\}$, where $\langle\theta_B, \mathbb{S}_B\rangle$ is the restriction of $\langle\theta_A, \mathbb{S}_A\rangle$ to Γ_B (and the program variables). Since $\vDash \{B\}\, s\, \{P\}$ by assumption, $\vDash \{B\}\, \theta_B\, \{\mathbb{S}_B\}$ implies $\mathbb{S}_B(B) \vDash \mathbb{S}_B(\lceil s\rceil) \sqsubseteq \mathbb{S}_B(P)$.

$$(\mathsf{VarI})\ \frac{}{\{A\}\, x^{\iota}\, \{P\}}\ (\, A \vDash P\, x'\,) \qquad (\mathsf{VarH})\ \frac{}{\{A\}\, x^{\sigma_1 \to \sigma_2}\, \{P\}}\ (\, A \vDash x' \sqsubseteq P\,)$$

$$(\mathsf{Const})\ \frac{}{\{A\}\, c^{\sigma}\, \{P\}}\ (\, A \vDash \lceil c \rceil \sqsubseteq P\,) \qquad (\mathsf{Fix})\ \frac{\{A\}\, s\, \{P\}}{\{A\}\, \mathsf{fix}\,(s)\, \{\mathbf{Y}\,(P)\}}$$

$$(\mathsf{AppH})\ \frac{\{A\}\, s^{(\sigma_1 \to \sigma_2) \to \tau}\, \{P\} \qquad \{A\}\, t^{\sigma_1 \to \sigma_2}\, \{Q\}}{\{A\}\, s\, t\, \{P\, Q\}}$$

$$(\mathsf{AppI})\ \frac{\{A\}\, s^{\iota \to \tau}\, \{P\} \qquad \{A\}\, t^{\iota}\, \{Q\}}{\{A\}\, s\, t\, \{\lambda \bar{z}.\, \exists x.\, (Q\, x \wedge P\, x\, \bar{z})\}}$$

$$(\mathsf{Abs})\ \frac{\{A\}\, s\, \{P\}}{\{A\}\, \lambda x.\, s\, \{\lambda x'.\, P\}}\ (\, x, x' \notin \mathsf{FV}(A)\,)$$

$$(\mathsf{Cond})\ \frac{\{A\}\, s\, \{Q\} \qquad \{A \wedge \exists x.\, x > 0 \wedge Q\, x\}\, t_1\, \{P\} \qquad \{A \wedge Q\, 0\}\, t_0\, \{P\}}{\{A\}\, \mathsf{if}(s, t_1, t_0)\, \{P\}}$$

$$(\mathsf{PCond})\ \frac{\{A\}\, s\, \{Q\} \qquad \{A \wedge ((\exists x.\, x > 0 \wedge Q\, x) \vee \neg \exists x.\, Q\, x)\}\, t_1\, \{P\} \qquad \{A \wedge (Q\, 0 \vee \neg \exists x.\, Q\, x)\}\, t_0\, \{P\}}{\{A\}\, \mathsf{pif}(s, t_1, t_0)\, \{P\}}$$

$$(\mathsf{Conseq})\ \frac{\vDash A \Rightarrow B \qquad \{B\}\, s\, \{P\} \qquad B \vDash P \sqsubseteq Q}{\{A\}\, s\, \{Q\}}$$

Figure 6.5: Proof rules for higher-order Hoare triples.

Because $\langle \theta_B, \mathbb{S}_B \rangle$ is the restriction of $\langle \theta_A, \mathbb{S}_A \rangle$, $\mathbb{S}_A(B) \vDash \mathbb{S}_A(\lceil s \rceil) \sqsubseteq \mathbb{S}_A(P)$. Finally, $\vDash A \Rightarrow B$ (which implies $\mathbb{S}_A \vDash A \Rightarrow B$) gives us $\mathbb{S}_A(A) \vDash \mathbb{S}_A(\lceil s \rceil) \sqsubseteq \mathbb{S}_A(P)$, as required. □

Lemma 6.12 (Right-monotonicity)**.**

(1) If $\vdash \{A\}\, s\, \{P\}$ *and* $A \vDash P \sqsubseteq Q$*, then* $\vdash \{A\}\, s\, \{Q\}$*.*

(2) If $\vDash \{A\}\, s\, \{P\}$ *and* $A \vDash P \sqsubseteq Q$*, then* $\vDash \{A\}\, s\, \{Q\}$*.*

Proof. The former follows directly from the Conseq rule for $\vdash$. For the latter, suppose that $\vDash \{A\}\, s\, \{P\}$ and $A \vDash P \sqsubseteq Q$.

Let Γ_P, Γ_Q be the ghost variables of P and Q, resp. It holds that $\Gamma_Q \subseteq \Gamma_P$. Take a substitution pair $\langle \theta_P, \mathbb{S}_P \rangle$ satisfying $\vDash \{A\}\, \theta_P\, \{\mathbb{S}_P\}$. Let $\langle \theta_Q, \mathbb{S}_Q \rangle$ be the restriction of $\langle \theta_P, \mathbb{S}_P \rangle$ to Γ_Q (and the program variables). Note that every $\langle \theta_Q, \mathbb{S}_Q \rangle$ is the image of such a $\langle \theta_P, \mathbb{S}_P \rangle$ and that $\vDash \{A\}\, \theta_Q\, \{\mathbb{S}_Q\}$. It suffices to show $\mathbb{S}_Q(A) \vDash \mathbb{S}_Q(\lceil s \rceil) \sqsubseteq \mathbb{S}_Q(Q)$.

Because $\vDash \{A\}\, s\, \{P\}$ by assumption, we have $\mathbb{S}_P(A) \vDash \mathbb{S}_P(\lceil s \rceil) \sqsubseteq \mathbb{S}_P(P)$. We obtain $\mathbb{S}_P(A) \vDash \mathbb{S}_P(\lceil s \rceil) \sqsubseteq \mathbb{S}_P(Q)$ from $A \vDash P \sqsubseteq Q$ (which implies $\mathbb{S}_P, A \vDash P \sqsubseteq Q$). Finally, because $\langle \theta_Q, \mathbb{S}_Q \rangle$ is a restriction of $\langle \theta_P, \mathbb{S}_P \rangle$ and $\mathbb{S}_Q(A) \vDash \mathbb{S}_Q(\lceil s \rceil) \sqsubseteq \mathbb{S}_Q(Q)$, as required. □

Lemma 6.13. *For all program expressions e,*

$$\vdash \{\top\}\, e\, \{\lceil e \rceil\}.$$

Proof. The constant and variable cases follow directly from the proof rules. So do the application, abstraction, and fixpoint cases.

For the conditional, $\lceil \mathsf{if}(s, t_1, t_0) \rceil = \lambda \overline{z}.\, (\exists x.\, x > 0 \wedge \lceil s \rceil\, x \wedge \lceil t_1 \rceil\, \overline{z}) \vee (\lceil s \rceil\, 0 \wedge \lceil t_0 \rceil\, \overline{z})$. By the IH, there exists a proof Δ_s of $\{\top\}\, s\, \{\lceil s \rceil\}$, a proof Δ_{t_1} of $\{\top\}\, t_1\, \{\lceil t_1 \rceil\}$, and a proof Δ_{t_0} of $\{\top\}\, t_0\, \{\lceil t_0 \rceil\}$, so that we can prove $\vdash \{\top\}\, \mathsf{if}(s, t_1, t_0)\, \{\lceil \mathsf{if}(s, t_1, t_0) \rceil\}$:

$$(\mathsf{Cond})\ \frac{\Delta_s \qquad \dfrac{(\mathsf{Conseq})\ \dfrac{\Delta_{t_1}}{\{\exists x.\, x > 0 \wedge \lceil s \rceil\, x\}\, t_1\, \{\lceil t_1 \rceil\}}}{(\mathsf{Conseq})\ \dfrac{\{\exists x.\, x > 0 \wedge \lceil s \rceil\, x\}\, t_1\, \{\lceil \mathsf{if}(s, t_1, t_0) \rceil\}}{}} \qquad \dfrac{(\mathsf{Conseq})\ \dfrac{\Delta_{t_0}}{\{\lceil s \rceil\, 0\}\, t_0\, \{\lceil t_0 \rceil\}}}{(\mathsf{Conseq})\ \dfrac{\{\lceil s \rceil\, 0\}\, t_0\, \{\lceil \mathsf{if}(s, t_1, t_0) \rceil\}}{}}}{\{\top\}\, \mathsf{if}(s, t_1, t_0)\, \{\lceil \mathsf{if}(s, t_1, t_0) \rceil\}}$$

The parallel conditional is similar, but with $\mathsf{pif}(s, t_1, t_0)$ instead of $\mathsf{if}(s, t_1, t_0)$ and $\vee \neg \exists x.\, \lceil s \rceil\, x$ tagged on to the non-trivial preconditions. □

6.3 Correctness

The correctness of our system relies on a correspondence between provable Hoare triples and valid Hoare triples. If all provable triples are valid, then our system is sound (i.e. if $\vdash \{A\}\, e\, \{P\}$ implies $\vDash \{A\}\, e\, \{P\}$). If the converse holds, and all valid triples are provable, then the system is complete. We establish soundness in Section 6.3.2.

Recall that Hoare logic is incomplete even for first-order logic. In particular, formulas involving implications tend to be undecidable. The use of the consequence rule, which relies on implications, then renders the Hoare logic incomplete. Since higher-order logic subsumes first-order logic, we too can only hope for a *relativised completeness*, which is with respect to some oracle for deciding implications/inclusions. We prove relativised completeness of our proof system in Section 6.3.3.

Because the lift $\lceil e \rceil$ of an expression e serves as an intermediate step in our proofs of correctness, Section 6.3.1 performs a sanity check on this lift. It shows that it is well-sorted (Lemma 6.14) and, therefore, compatible with our definition of Hoare triples. Furthermore, we show in Proposition 6.18 that the lift is invariant over one-step reduction. That is, that $e \to e'$ implies $\mathcal{C}[\![\lceil e \rceil]\!] = \mathcal{C}[\![\lceil e' \rceil]\!]$.

6.3.1 Lift

Lemma 6.14 (Well-sortedness of the lift)**.** *If $e : \sigma$, then $\lceil e : \sigma \rceil : \mathsf{Rel}^+(\sigma)$.*

Proof. By a straightforward structural induction on e.

Case $e = n : \iota$**.** Note that m has sort ι too, so that $\lceil n \rceil = \lambda m.\, (m = n) : \iota \to o = \mathsf{Rel}^+(\iota)$.

Case $e = c : \iota \to \iota \to \iota$**.** The formal parameters x, y, z in c' all have sort ι, so that $\lceil c \rceil = c' : \iota \to \iota \to \iota \to o = \mathsf{Rel}^+(\iota \to \iota \to \iota)$.

Case $e = x : \iota$**.** As case $n : \iota$ above, relying on $x' : \mathsf{Rel}^-(\iota) = \iota$.

Case $e = x : \sigma_1 \to \sigma_2$**.** This follows from $x' : \mathsf{Rel}^-(\sigma_1 \to \sigma_2) = \mathsf{Rel}^+(\sigma_1 \to \sigma_2)$.

Case $e = s^{\iota \to \tau}\, t^{\iota}$**.** Let $\tau = \tau_1 \to \ldots \tau_n \to \iota$. This means that $\mathsf{Rel}^+(\tau) = \mathsf{Rel}^-(\tau_1) \to \cdots \to \mathsf{Rel}^-(\tau_n) \to \iota \to o$. By the IH, $\lceil s \rceil : \mathsf{Rel}^-(\iota \to \tau) = \iota \to \mathsf{Rel}^+(\tau)$ and $\lceil t \rceil : \mathsf{Rel}^+(\iota)$. Since $x : \iota$ and $\overline{z} = z_1 \ldots z_n\, r$ for $z_i : \mathsf{Rel}^-(\tau_i)$ and $r : \iota$, the terms $\lceil s \rceil\, x\, \overline{z}$ and $\lceil t \rceil\, x$ have sort o. So does $\exists x.\, \lceil t \rceil\, x \wedge \lceil s \rceil\, x\, \overline{z}$. The claim follows straightforwardly from the formal parameters $\overline{z}$.

Case $e = s^{(\sigma \to \sigma') \to \tau}\, t^{\sigma \to \sigma'}$**.** By the IH, $\lceil s \rceil : \mathsf{Rel}^+((\sigma \to \sigma') \to \tau) = \mathsf{Rel}^+(\sigma \to \sigma') \to \mathsf{Rel}^+(\tau)$ and $\lceil t \rceil : \mathsf{Rel}^+(\sigma \to \sigma')$, so the claim follows directly.

Case $e = \lambda x^{\sigma}.\, s^{\tau}$**.** By the IH, $\lceil s \rceil : \mathsf{Rel}^+(\tau)$. The relational clone x' has sort $\mathsf{Rel}^-(\sigma)$, so that $\lceil \lambda x.\, s \rceil = \lambda x'.\, \lceil s \rceil : \mathsf{Rel}^-(\sigma) \to \mathsf{Rel}^+(\tau) = \mathsf{Rel}^+(\sigma \to \tau)$, as required.

Case $e = \mathsf{if}_{\tau}(s, t_1, t_0)$**.** Let $\tau = \tau_1 \to \ldots \tau_n \to \iota$. This means that $\mathsf{Rel}^+(\tau) = \mathsf{Rel}^-(\tau_1) \to \cdots \to \mathsf{Rel}^-(\tau_n) \to \iota \to o$. By the IH, $\lceil s \rceil : \mathsf{Rel}^+(\iota)$ and $\lceil t_j \rceil : \mathsf{Rel}^+(\tau)$, for $j \in \{0, 1\}$. Since $x : \iota$ and $\overline{z} = z_1 \ldots z_n\, r$ for $z_i : \mathsf{Rel}^-(\tau_i)$ and $r : \iota$, the terms $\lceil s \rceil\, x$, $\lceil t_1 \rceil\, \overline{z}$, $\lceil s \rceil\, 0$, and $\lceil t_0 \rceil\, \overline{z}$ have sort o. We conclude that $\mathsf{if}_{\tau}(s, t_1, t_0) : \mathsf{Rel}^+(\tau)$, as required.

Case $e = \mathsf{pif}(s, t_1, t_0)$**.** Similar to the previous case.

Case $e = \mathsf{fix}_{\sigma\to\tau}(s)$**.** By the IH, $\lceil s\rceil : \mathsf{Rel}^+((\sigma\to\tau)\to\sigma\to\tau) = \mathsf{Rel}^+(\sigma\to\tau)\to\mathsf{Rel}^+(\sigma\to\tau)$. The claim follows directly from the fixpoint construction. □

Lemma 6.15. *For all closed expressions* $\vdash e : \iota\to\tau$*, and* $\vdash n : \iota$*,*

$$\mathcal{C}[\![\lceil e^{\iota\to\tau}\, n\rceil]\!] = \mathcal{C}[\![\lceil e^{\iota\to\tau}\rceil\, n]\!]$$

Proof.

$$\begin{aligned}\mathcal{C}[\![\lceil e^{\iota\to\tau}\, n\rceil]\!] &= \mathcal{C}[\![\lambda\overline{z}.\,\exists x.\,\lceil n\rceil\, x \wedge \lceil e\rceil\, x\,\overline{z}]\!]\\ &= \mathcal{C}[\![\lambda\overline{z}.\,\exists x.\, n = x \wedge \lceil e\rceil\, x\,\overline{z}]\!]\\ &= \mathcal{C}[\![\lambda\overline{z}.\,\lceil e\rceil\, n\,\overline{z}]\!]\\ &= \mathcal{C}[\![\lceil e^{\iota\to\tau}\rceil\, n]\!]\end{aligned}$$

□

Definition 6.16. *Let us define:*

$$\lceil v^\sigma\rceil^- := \begin{cases} v & \text{if } \sigma = \iota \\ \lceil v^\sigma\rceil & O/W \end{cases}$$

Lemma 6.17. *For all* $\mathsf{Rel}^-(\Delta), x' : \mathsf{Rel}^+(\sigma) \vdash \lceil e\rceil$*, closed values* v *of sort* σ*, and valuations* $\alpha \in \mathcal{C}[\![\mathsf{Rel}^-(\Delta)]\!]$*,*

$$\mathcal{C}[\![\lceil e\rceil[x' \mapsto \lceil v\rceil^-]]\!](\alpha) = \mathcal{C}[\![\lceil e[x\mapsto v]\rceil]\!](\alpha).$$

Proof. By induction on the structure of e. That is, given some e, we prove that the claim holds for all sort environments $\mathsf{Rel}^-(\Delta)$, values v of sort σ, and $\alpha \in \mathcal{C}[\![\mathsf{Rel}^-(\Delta)]\!]$ such that $\mathsf{Rel}^-(\Delta), x : \sigma \vdash \lceil e\rceil$. Note that the latter implies that $\lceil e\rceil[x' \mapsto \lceil v\rceil^-]$ and $\lceil e[x\mapsto v]\rceil$ are well-sorted over $\mathsf{Rel}^-(\Delta)$. Furthermore, $\Delta \vdash e[x\mapsto v]$. We omit the sort environment for brevity.

Case $e = c \in \{+,-,\leq\}\cup\mathbb{N}$**.**

$$\begin{aligned}\mathcal{C}[\![\lceil c\rceil[x' \mapsto \lceil v\rceil^-]]\!](\alpha) &= \mathcal{C}[\![\lceil c\rceil]\!](\alpha)\\ &= \mathcal{C}[\![\lceil c[x\mapsto v]\rceil]\!](\alpha)\end{aligned}$$

Case $e = y \in \Delta$**.**

$$\begin{aligned}\mathcal{C}[\![\lceil y\rceil[x' \mapsto \lceil v\rceil^-]]\!](\alpha) &= \mathcal{C}[\![\lceil y\rceil]\!](\alpha)\\ &= \mathcal{C}[\![\lceil y[x\mapsto v]\rceil]\!](\alpha)\end{aligned}$$

Case $e = x^{\iota}$**.**

$$\begin{aligned}
\mathcal{C}[\![\lceil x \rceil[x' \mapsto \lceil v \rceil^-]]\!](\alpha) &= \mathcal{C}[\![(\lambda m.\, m = x')[x' \mapsto \lceil v \rceil^-]]\!](\alpha) \\
&= \mathcal{C}[\![\lambda m.\, m = v]\!](\alpha) \\
&= \mathcal{C}[\![\lceil v \rceil]\!](\alpha) \\
&= \mathcal{C}[\![\lceil x[x \mapsto v] \rceil]\!](\alpha)
\end{aligned}$$

Case $e = x^{\sigma_1 \to \sigma_2}$**.**

$$\begin{aligned}
\mathcal{C}[\![\lceil x \rceil[x' \mapsto \lceil v \rceil^-]]\!](\alpha) &= \mathcal{C}[\![x'[x' \mapsto \lceil v \rceil^-]]\!](\alpha) \\
&= \mathcal{C}[\![\lceil v \rceil]\!](\alpha) \\
&= \mathcal{C}[\![\lceil x[x \mapsto v] \rceil]\!](\alpha)
\end{aligned}$$

Now suppose the claim holds for all subexpressions. The arguably most interesting inductive case is abstraction.

Case $e = \lambda y.\, e'$**.**

$$\begin{aligned}
\mathcal{C}[\![\lceil \lambda y.\, e' \rceil[x' \mapsto \lceil v \rceil^-]]\!](\alpha) &= \mathcal{C}[\![(\lambda y'.\, \lceil e' \rceil)[x' \mapsto \lceil v \rceil^-]]\!](\alpha) \\
&= \mathcal{C}[\![\lambda y'.\, \lceil e' \rceil[x' \mapsto \lceil v \rceil^-]]\!](\alpha) \\
&= \lambda y''.\, \mathcal{C}[\![\lceil e' \rceil[x' \mapsto \lceil v \rceil^-]]\!](\alpha[y' \mapsto y'']) \\
&= \lambda y''.\, \mathcal{C}[\![\lceil e'[x \mapsto v] \rceil]\!](\alpha[y' \mapsto y'']) \qquad \text{IH} \\
&= \mathcal{C}[\![\lambda y'.\, \lceil e'[x \mapsto v] \rceil]\!](\alpha) \\
&= \mathcal{C}[\![\lceil \lambda y.\, e'[x \mapsto v] \rceil]\!](\alpha) \\
&= \mathcal{C}[\![\lceil (\lambda y.\, e')[x \mapsto v] \rceil]\!](\alpha)
\end{aligned}$$

Since the other inductive cases are straightforward substitutions, we provide one more case as a proof of concept. We abuse notation by using the same symbols for variables as for semantic elements.

Case $e = e_1\, e_2'$**.**

$$
\begin{aligned}
&\mathcal{C}[\![\ulcorner e_1\, e_2 \urcorner[x' \mapsto \ulcorner v \urcorner^-]]\!](\alpha) \\
&= \mathcal{C}[\![(\lambda \overline{z}\, r.\, \exists y.\, \ulcorner e_2 \urcorner\, y \wedge \ulcorner e_1 \urcorner\, y\, \overline{z}\, r)[x' \mapsto \ulcorner v \urcorner]]\!](\alpha) \\
&= \mathcal{C}[\![\lambda \overline{z}\, r.\, \exists y.\, \ulcorner e_2 \urcorner[x' \mapsto \ulcorner v \urcorner]\, y \wedge \ulcorner e_1 \urcorner[x' \mapsto \ulcorner v \urcorner]\, y\, \overline{z}\, r]\!](\alpha) \\
&= \lambda \overline{z}\, r.\, \exists y.\, \mathcal{C}[\![\ulcorner e_2 \urcorner[x' \mapsto \ulcorner v \urcorner]]\!](\alpha)\, y \wedge \mathcal{C}[\![\ulcorner e_1 \urcorner[x' \mapsto \ulcorner v \urcorner]]\!](\alpha)\, y\, \overline{z}\, r \\
&= \lambda \overline{z}\, r.\, \exists y.\, \mathcal{C}[\![\ulcorner e_2[x \mapsto v] \urcorner]\!](\alpha)\, y \wedge \mathcal{C}[\![\ulcorner e_1[x \mapsto v] \urcorner]\!](\alpha)\, y\, \overline{z}\, r \qquad \text{IH} \\
&= \mathcal{C}[\![\lambda \overline{z}\, r.\, \exists y.\, \ulcorner e_2[x \mapsto v] \urcorner\, y \wedge \ulcorner e_1[x \mapsto v] \urcorner\, y\, \overline{z}\, r]\!](\alpha) \\
&= \mathcal{C}[\![\ulcorner e_1[x \mapsto v]\, e_2[x \mapsto v] \urcorner]\!](\alpha) \\
&= \mathcal{C}[\![\ulcorner (e_1\, e_2)[x \mapsto v] \urcorner]\!](\alpha)
\end{aligned}
$$

□

Proposition 6.18. *For all closed expressions* $\vdash e$ *and values* v*,*

$$e \to^* v \quad \Rightarrow \quad \mathcal{C}[\![\ulcorner e \urcorner]\!] = \mathcal{C}[\![\ulcorner v \urcorner]\!].$$

Proof. It suffices to show that $e \to e'$ implies that $\mathcal{C}[\![\ulcorner e \urcorner]\!] = \mathcal{C}[\![\ulcorner e' \urcorner]\!]$. We proceed by induction on the $\to$-rules.

Case (OSIf$_1$). $\mathsf{if}(n, e_1, e_0) \to e_1$ with $n > 0$

$$
\begin{aligned}
\mathcal{C}[\![\ulcorner \mathsf{if}(n, e_1, e_0) \urcorner]\!] &= \mathcal{C}[\![\lambda \overline{z}.\, (\exists x.\, x > 0 \wedge \ulcorner n \urcorner\, x \wedge \ulcorner e_1 \urcorner\, \overline{z}) \vee (\ulcorner n \urcorner\, 0 \wedge \ulcorner e_0 \urcorner\, \overline{z})]\!] \\
&= \mathcal{C}[\![\lambda \overline{z}.\, \ulcorner e_1 \urcorner\, \overline{z}]\!] \\
&= \mathcal{C}[\![\ulcorner e_1 \urcorner]\!]
\end{aligned}
$$

The (OSIf$_0$) case is similar.

Case (OSPlus). $n_1 + n_2 \to n'$

$$
\begin{aligned}
\mathcal{C}[\![\ulcorner n_1 + n_2 \urcorner]\!] &= \mathcal{C}[\![\ulcorner + \urcorner\, n_1\, n_2]\!] \qquad \text{Lem 6.15} \\
&= \mathcal{C}[\![\lambda r.\, n_1 + n_2 = r]\!] \\
&= \mathcal{C}[\![\ulcorner n' \urcorner]\!]
\end{aligned}
$$

The (OSMin) case is similar.

Case (OSLeq$_1$). $n_1 \leq n_2 \to 1$

$$\begin{aligned}
\mathcal{C}[\![\ulcorner n_1 \leq n_2 \urcorner]\!] &= \mathcal{C}[\![\ulcorner \leq \urcorner\, n_1\, n_2]\!] && \text{Lem 6.15} \\
&= \mathcal{C}[\![\lambda r.\, (r = 1 \wedge n_1 \leq n_2) \vee (r = 0 \wedge n_1 > n_2)]\!] \\
&= \mathcal{C}[\![\lambda r.\, r = 1]\!] \\
&= \mathcal{C}[\![\ulcorner 1 \urcorner]\!]
\end{aligned}$$

The (OSLeq$_0$) case is similar.

Case (OSβ_v). $(\lambda x.\, e)\, v' \to e[v'/x]$

$$\begin{aligned}
\mathcal{C}[\![\ulcorner (\lambda x^{\iota}.\, e)\, n \urcorner]\!] &= \mathcal{C}[\![\ulcorner \lambda x.\, e \urcorner\, n]\!] && \text{Lem 6.15} \\
&= \mathcal{C}[\![(\lambda x'.\, \ulcorner e \urcorner)\, n]\!] \\
&= \mathcal{C}[\![\ulcorner e \urcorner [n/x']]\!] \\
&= \mathcal{C}[\![\ulcorner e[n/x] \urcorner]\!] && \text{Lem 6.17} \\
\mathcal{C}[\![\ulcorner (\lambda x^{\sigma_1 \to \sigma_2}.\, e)\, v' \urcorner]\!] &= \mathcal{C}[\![\ulcorner \lambda x.\, e \urcorner\, \ulcorner v' \urcorner]\!] \\
&= \mathcal{C}[\![(\lambda x'.\, \ulcorner e \urcorner)\, \ulcorner v' \urcorner]\!] \\
&= \mathcal{C}[\![\ulcorner e \urcorner [\ulcorner v' \urcorner / x']]\!] \\
&= \mathcal{C}[\![\ulcorner e[v'/x] \urcorner]\!] && \text{Lem 6.17}
\end{aligned}$$

Case (OSFixEv). $\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e) \to \lambda x.\, (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e))\, x$

$$\begin{aligned}
\mathcal{C}[\![\ulcorner \mathsf{fix}_{\sigma\to\tau}\, (\lambda y.\, e) \urcorner]\!] &= \mathcal{C}[\![\mathsf{Y}_{\mathsf{Rel}^+(\sigma\to\tau)}\, \ulcorner \lambda y.\, e \urcorner]\!] \\
&= \mathcal{C}[\![\ulcorner \lambda y.\, e \urcorner\, (\mathsf{Y}_{\mathsf{Rel}^+(\sigma\to\tau)}\, \ulcorner \lambda y.\, e \urcorner)]\!] \\
&= \mathcal{C}[\![\ulcorner \lambda y.\, e \urcorner\, \ulcorner \mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e) \urcorner]\!] \\
&= \mathcal{C}[\![\ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e)) \urcorner]\!]
\end{aligned}$$

Now, if $\sigma = \iota$, then we proceed with:

$$\begin{aligned}
\mathcal{C}[\![\ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e)) \urcorner]\!] &= \mathcal{C}[\![\lambda x'\, \overline{z}.\, \ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e)) \urcorner\, x'\, \overline{z}]\!] \\
&= \mathcal{C}[\![\lambda x'\, \overline{z}.\, \ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e)) \urcorner\, n\, \overline{z} \wedge \exists n.\, \ulcorner x \urcorner\, n]\!] \\
&= \mathcal{C}[\![\ulcorner \lambda x.\, (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e))\, x \urcorner]\!]
\end{aligned}$$

Otherwise:

$$\begin{aligned}
\mathcal{C}[\![\ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e)) \urcorner]\!] &= \mathcal{C}[\![\lambda x'.\, \ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e)) \urcorner\, x']\!] \\
&= \mathcal{C}[\![\lambda x'.\, \ulcorner (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e))\, x \urcorner]\!] \\
&= \mathcal{C}[\![\ulcorner \lambda x.\, (\lambda y.\, e)\, (\mathsf{fix}_{\sigma\to\tau}(\lambda y.\, e))\, x \urcorner]\!]
\end{aligned}$$

Case (OSPif$_1$). $\mathsf{pif}(n, e_2, e_3) \to e_2$ with $n > 0$

$$\begin{aligned}
&\mathcal{C}[\![\lceil \mathsf{pif}(n, e_2, e_3)\rceil]\!] \\
&= \mathcal{C}[\![\lambda r.\, (\exists x.\, x > 0 \wedge \lceil n\rceil\, x \wedge \lceil e_2\rceil\, r) \vee (\lceil n\rceil\, 0 \wedge \lceil e_3\rceil\, r) \vee (\lceil e_2\rceil\, r \wedge \lceil e_3\rceil\, r)]\!] \\
&= \mathcal{C}[\![\lambda r.\, \lceil e_2\rceil\, r]\!] \\
&= \mathcal{C}[\![\lceil e_2\rceil]\!]
\end{aligned}$$

The (OSPif$_0$) case is similar.

Case (OSPif$_\perp$). $\mathsf{pif}(e_1, n, n) \to n$

$$\begin{aligned}
&\mathcal{C}[\![\lceil \mathsf{pif}(e_1, n, n)\rceil]\!] \\
&= \mathcal{C}[\![\lambda r.\, (\exists x.\, x > 0 \wedge \lceil e_1\rceil\, x \wedge \lceil n\rceil\, r) \vee (\lceil e_1\rceil\, 0 \wedge \lceil n\rceil\, r) \vee (\lceil n\rceil\, r \wedge \lceil n\rceil\, r)]\!] \\
&= \mathcal{C}[\![\lambda r.\, \lceil n\rceil\, r]\!] \\
&= \mathcal{C}[\![\lceil n\rceil]\!]
\end{aligned}$$

The inductive cases are straightforward reductions-in-context that can be proved by simply writing out the $\lceil -\rceil$s and substituting $\mathcal{C}[\![\lceil e'\rceil]\!]$ for $\mathcal{C}[\![\lceil e\rceil]\!]$ whenever we rely on a rule $e \to e'$. For proof of concept, we present one case. We again abuse notation by using the same symbols for variables as for semantic elements.

Case (OSAppL). $e_1\, e_2 \to e_1'\, e_2$, provided that $e_1 \to e_1'$

$$\begin{aligned}
\mathcal{C}[\![\lceil e_1\, e_2^{\iota}\rceil]\!] &= \mathcal{C}[\![\lambda \overline{z}.\, \exists x.\, \lceil e_2\rceil\, x \wedge \lceil e_1\rceil\, x\, \overline{z}]\!] \\
&= \lambda \overline{z}.\, \exists x.\, \mathcal{C}[\![\lceil e_2\rceil]\!]\, x \wedge \mathcal{C}[\![\lceil e_1\rceil]\!]\, x\, \overline{z} \\
&= \lambda \overline{z}.\, \exists x.\, \mathcal{C}[\![\lceil e_2\rceil]\!]\, x \wedge \mathcal{C}[\![\lceil e_1'\rceil]\!]\, x\, \overline{z} \qquad \text{IH} \\
&= \mathcal{C}[\![\lceil e_1'\, e_2\rceil]\!] \\
\mathcal{C}[\![\lceil e_1\, e_2^{\sigma_1 \to \sigma_2}\rceil]\!] &= \mathcal{C}[\![\lceil e_1\rceil\ \lceil e_2\rceil]\!] \\
&= \mathcal{C}[\![\lceil e_1\rceil]\!]\, \mathcal{C}[\![\lceil e_2\rceil]\!] \\
&= \mathcal{C}[\![\lceil e_1'\rceil]\!]\, \mathcal{C}[\![\lceil e_2\rceil]\!] \qquad \text{IH} \\
&= \mathcal{C}[\![\lceil e_1'\, e_2\rceil]\!]
\end{aligned}$$

□

Lemma 6.19. *If* $\vdash e : \iota$ *and* $e \Downarrow$*, then* $\lambda m.\, (m = \|e\|) = \mathcal{C}[\![\lceil e\rceil]\!]$.

Proof. Let $\vdash e : \iota$ and $e \Downarrow$. Thus, there exists n such that $e \Downarrow n$. By Lemma 6.6, $e \to^* n$. Then, it follows that:

$$\begin{aligned}
\lambda r.\,(r = \|e\|) &= \lambda r.\,(r = \|n\|) && \text{Thm 6.3}\\
&= \lambda r.\,(r = n)\\
&= \mathcal{C}[\![\lambda m.\, m = n]\!]\\
&= \mathcal{C}[\![\ulcorner n \urcorner]\!]\\
&= \mathcal{C}[\![\ulcorner e \urcorner]\!] && \text{Prop 6.18}
\end{aligned}$$

□

6.3.2 Soundness

We show in Proposition 6.20 that any finite element in a complete lattice of relations $\mathcal{C}[\![\rho]\!]$ can be described by some term of higher-order logic. Intuitively, continuity plus *definability* of finite elements allows us to capture the entire semantic domain using terms rather than semantic elements.

This definability property simplifies the proof of soundness in Theorem 6.21, because we can prove inclusions $A \vDash P \sqsubseteq Q$ by showing that each $A \vDash P\,\overline{v} : o$ implies $A \vDash Q\,\overline{v} : o$, for all terms $\overline{v} = v_1 \ldots v_n$ of the appropriate sorts.

Proposition 6.20 (Definability)**.** *For all finite elements $b \in \mathcal{C}[\![\rho]\!]$, there exists a closed term $t : \rho$ of higher-order logic such that $\mathcal{C}[\![t : \rho]\!] = b$.*

Proof. We proceed by induction on sorts. The result is trivial for propositional sort o; the formulas false and true correspond to $0 \in \mathcal{C}[\![o]\!]$ and $1 \in \mathcal{C}[\![o]\!]$, respectively.

For higher-order sorts, recall $\mathcal{C}[\![\sigma \to \rho_2]\!]$ is an algebraic lattice (by Proposition 2.16) with basis

$$B_{\sigma\to\rho_2} := \left\{ \bigsqcup M \;\middle|\; M \text{ is a finite subset of } S_{\sigma\to\rho_2} \right\},$$

where

$$S_{\sigma\to\rho_2} := \{(a \searrow c) \mid a \in \mathit{fin}(\mathcal{C}[\![\sigma]\!]), c \in \mathit{fin}(\mathcal{C}[\![\rho_2]\!])\}$$

and the step function $(a \searrow c)$ is defined by, for all $x \in \mathcal{C}[\![\sigma]\!]$:

$$(a \searrow c)(x) := \begin{cases} c & \text{if } a \sqsubseteq x \\ \bot_{\mathcal{C}[\![\rho_2]\!]} & \text{O/W} \end{cases}$$

Now suppose the claim holds for all relational sorts smaller than ρ.

Case $\rho = \iota \to \rho_2$**.** Let $(a \searrow c) \in S_{\iota \to \rho_2}$. By the induction hypothesis, there exists a term $t : \rho_2$ such that $\mathcal{C}[\![t : \rho_2]\!] = c$, so that $(a \searrow c) = \mathcal{C}[\![\lambda n\, \overline{z}.\, n = a \wedge t\, \overline{z}]\!]$.

Now, let $M = \{m_1, \ldots, m_k\}$ such that $\bigsqcup M \in B_{\iota \to \rho_2}$, with terms $t_1, \ldots, t_k$ capturing its respective elements. Then, $\bigsqcup M = \mathcal{C}[\![\lambda n\, \overline{z}.\, t_1\, n\, \overline{z} \vee \cdots \vee t_k\, n\, \overline{z}]\!]$.

Case $\rho = \rho_1 \to \rho_2$**.** Let $(a \searrow c) \in S_{\rho_2 \to \rho_2}$. By the induction hypothesis, there exist terms $t_a : \rho_1$ and $t_c : \rho_2$ such that $\mathcal{C}[\![t_a : \rho_1]\!] = a$ and $\mathcal{C}[\![t_c : \rho_2]\!] = c$. Then,

$$(a \searrow c) = \mathcal{C}[\![\lambda x\, \overline{z}.\, (\forall \overline{y}.\, t_1\, \overline{y} \Rightarrow x\, \overline{y}) \wedge t_2\, \overline{z}]\!].$$

Note that this has a meaningful continuous interpretation despite the implication, because variable x occurs 'positively' (i.e. in the head). Of course, we already knew from Proposition 2.16 that $(a \searrow c)$ is continuous.

Now, let $M = \{m_1, \ldots, m_k\}$ such that $\bigsqcup M \in B_{\rho_1 \to \rho_2}$, with terms $t_1, \ldots, t_k$ capturing its respective elements. Then, $\bigsqcup M = \mathcal{C}[\![\lambda x\, \overline{z}.\, t_1\, x\, \overline{z} \vee \cdots \vee t_k\, x\, \overline{z}]\!]$. □

Theorem 6.21 (Soundness)**.** *If* $\vdash \{A\}\, s\, \{P\}$ *then* $\vDash \{A\}\, s\, \{P\}$.

Proof. We proceed by induction on the proof tree of $\vdash \{A\}\, s\, \{P\}$ (see Figure 6.5 for the proof rules). To prove $\vDash \{A\}\, s\, \{P\}$, it suffices to show that $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$ for all substitution pairs $\langle \theta, \mathbb{S} \rangle$ such that $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$.

Case Varl**.** Suppose that $\vdash \{A\}\, x^{\iota}\, \{P\}$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. By the side condition of the proof rule, $A \vDash P\, x'$, and thus $\mathbb{S}, A \vDash P\, x'$. Note that $\lceil x \rceil = \lambda m.\, (m = x')$ is the characteristic function of x' (and $\mathbb{S}(\lceil x \rceil)$ of $\mathbb{S}(x')$). It follows that $\mathbb{S}, A \vDash \lceil x \rceil \sqsubseteq P$, as required.

Case VarH**.** Suppose that $\vdash \{A\}\, x\, \{P\}$. By the side condition of the proof rule, $A \vDash x' \sqsubseteq P$. The definition of the semantics along with $x' = \lceil x \rceil$ gives us the result: $\vDash \{A\}\, x\, \{P\}$.

Case Const**.** This follows immediately from the side condition.

This concludes the base cases of the induction.

Case Appl**.** Suppose that $\vdash \{A\}\, s^{\iota \to \tau}\, t^{\iota}\, \{\lambda \bar{z}.\, \exists x.\, (Q\, x \wedge P\, x\, \bar{z})\}$. By the IH on the premises of the rule, $\vDash \{A\}\, s\, \{P\}$ and $\vDash \{A\}\, t\, \{Q\}$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. This gives us $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$ and $\mathbb{S}, A \vDash \lceil t \rceil \sqsubseteq Q$.

To prove that $\mathbb{S}, A \vDash \lceil s\, t \rceil \sqsubseteq \lambda \bar{z}.\, \exists x.\, (Q\, x \wedge P\, x\, \bar{z})$, assume $\mathbb{S}, A \vDash \lceil s\, t \rceil\, \overline{v}$ for some terms $\overline{v}$ of the appropriate sorts (which suffices by Proposition 6.20). That is, assume

$\mathbb{S}, A \vDash \exists x.\, \lceil t \rceil\, x \wedge \lceil s \rceil\, x\, \overline{v}$. It follows from $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$ and $\mathbb{S}, A \vDash \lceil t \rceil \sqsubseteq Q$ that $\mathbb{S}, A \vDash \exists x.\, Q\, x \wedge P\, x\, \overline{v}$. Finally, we conclude $\vDash \{A\}\, s\, t\, \{\lambda \overline{z}.\, \exists x.\, (Q\, x \wedge P\, x\, \overline{z})\}$.

Case AppH. Suppose that $\vdash \{A\}\, s^{\sigma \to \tau}\, t^{\sigma}\, \{P\, Q\}$ for $\sigma = \sigma_1 \to \sigma_2$. By the IH on the premises of the rule, $\vDash \{A\}\, s\, \{P\}$ and $\vDash \{A\}\, t\, \{Q\}$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. This gives us $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$ and $\mathbb{S}, A \vDash \lceil t \rceil \sqsubseteq Q$.

To prove that $\mathbb{S}, A \vDash \lceil s\, t \rceil \sqsubseteq P\, Q$, assume $\mathbb{S}, A \vDash \lceil s\, t \rceil\, \overline{v}$ for some terms $\overline{v}$ of the appropriate sorts. That is, $\mathbb{S}, A \vDash \lceil s \rceil\, \lceil t \rceil\, \overline{v}$. It follows from monotonicity and $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$ and $\mathbb{S}, A \vDash \lceil t \rceil \sqsubseteq Q$ that $\mathbb{S}, A \vDash P\, Q\, \overline{v}$. Finally we conclude $\vDash \{A\}\, s\, t\, \{P\, Q\}$.

Case Abs. Suppose that $\vdash \{A\}\, \lambda x.\, s\, \{\lambda x'.\, P\}$, where $x, x' \notin \mathsf{FV}(A)$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. By the IH on the premise of the rule, $\vDash \{A\}\, s\, \{P\}$. This gives us $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$. Clearly, also $\mathbb{S}, A \vDash \lambda x'.\, \lceil s \rceil \sqsubseteq \lambda x'.\, P$. The claim $\vDash \{A\}\, \lambda x.\, s\, \{\lambda x'.\, P\}$ follows from $\lceil \lambda x.\, s \rceil = \lambda x'.\, \lceil s \rceil$.

Case Cond. Suppose $\vdash \{A\}\, \mathsf{if}_\sigma(s, t_1, t_0)\, \{P\}$. By the IH on the premises of the rule, $\vDash \{A\}\, s\, \{Q\}$, $\vDash \{A \wedge \exists x.\, x > 0 \wedge Q\, x\}\, t_1\, \{P\}$, and $\vDash \{A \wedge Q\, 0\}\, t_0\, \{P\}$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. This gives us $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq Q$ and $\mathbb{S}, A \wedge Q\, 0 \vDash \lceil t_0 \rceil \sqsubseteq P$ and $\mathbb{S}, A \wedge \exists x.\, x > 0 \wedge Q\, x \vDash \lceil t_1 \rceil \sqsubseteq P$.

To prove that $\mathbb{S}, A \vDash \lceil \mathsf{if}_\sigma(s, t_1, t_0) \rceil \sqsubseteq P$, let $\mathbb{S}, A \vDash \lceil \mathsf{if}_\sigma(s, t_1, t_0) \rceil\, \overline{v}$ for some terms $\overline{v}$ of the appropriate sorts. This means that $\mathbb{S}, A \vDash \exists x.\, x > 0 \wedge \lceil s \rceil\, x \wedge \lceil t_1 \rceil\, \overline{v}$ or $\mathbb{S}, A \vDash \lceil s \rceil\, 0 \wedge \lceil t_0 \rceil\, \overline{v}$.

Consider the latter case (the former is analogous). It follows from $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq Q$ that $\mathbb{S}, A \vDash Q\, 0 \wedge \lceil t_0 \rceil\, \overline{v}$. This implies that $\mathbb{S}, A \wedge Q\, 0 \vDash \lceil t_0 \rceil\, \overline{v}$, which gives us $\mathbb{S}, A \wedge Q\, 0 \vDash P\, \overline{v}$. However, $Q\, 0$ was already modelled by $\mathbb{S}, A$, so $\mathbb{S}, A \vDash P\, \overline{v}$.

The former case also results in $\mathbb{S}, A \vDash P\, \overline{v}$ using similar reasoning, so we conclude $\mathbb{S}, A \vDash \lceil \mathsf{if}_\sigma(s, t_1, t_0) \rceil \sqsubseteq P$ and $\vDash \{A\}\, \mathsf{if}_\sigma(s, t_1, t_0)\, \{P\}$.

Case PCond. Suppose $\vdash \{A\}\, \mathsf{pif}(s, t_1, t_0)\, \{P\}$. By the IH on the premises of the rule, $\vDash \{A \wedge ((\exists x.\, x > 0 \wedge Q\, x) \vee \neg \exists x.\, Q\, x)\}\, t_1\, \{P\}$, $\vDash \{A \wedge (Q\, 0 \vee \neg \exists x.\, Q\, x)\}\, t_0\, \{P\}$, and $\vDash \{A\}\, s\, \{Q\}$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. This gives us $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq Q$ and $\mathbb{S}, A \wedge ((\exists x.\, x > 0 \wedge Q\, x) \vee \neg \exists x.\, Q\, x) \vDash \lceil t_1 \rceil \sqsubseteq P$ and $\mathbb{S}, A \wedge (Q\, 0 \vee \neg \exists x.\, Q\, x) \vDash \lceil t_0 \rceil \sqsubseteq P$.

To prove that $\mathbb{S}, A \vDash \lceil \mathsf{pif}(s, t_1, t_0) \rceil \sqsubseteq P$, let $\mathbb{S}, A \vDash \lceil \mathsf{pif}(s, t_1, t_0) \rceil\, v$ for some value $v : \iota$. This means that $\mathbb{S}, A \vDash \exists x.\, x > 0 \wedge \lceil s \rceil\, x \wedge \lceil t_1 \rceil\, v$ or $\mathbb{S}, A \vDash \lceil s \rceil\, 0 \wedge \lceil t_0 \rceil\, v$ or $\mathbb{S}, A \vDash \lceil t_1 \rceil\, v \wedge \lceil t_0 \rceil\, v$.

The first two cases are the same as for Cond. The first results in $\mathbb{S}, A \wedge \exists x.\, x > 0 \wedge Q\, x \vDash P\, \overline{v}$ and the second in $\mathbb{S}, A \wedge Q\, 0 \vDash P\, \overline{v}$.

Note that $(\exists x.\, x > 0 \wedge Q\, x) \vee Q\, 0 \vee \neg \exists x.\, Q\, x = \mathsf{true}$. If one of the first two disjuncts holds, then we already have $P\, v$, so consider $\mathbb{S}, A \wedge \neg \exists x.\, Q\, x \vDash \lceil t_1 \rceil\, v \wedge \lceil t_0 \rceil\, v$ for case three. This immediately gives us $\mathbb{S}, A \wedge \neg \exists x.\, Q\, x \vDash P\, v$.

We have now covered all cases of Q, so that $\mathbb{S}, A \vDash P\, v$. We conclude that $\mathbb{S}, A \vDash \lceil \mathsf{pif}(s, t_1, t_0) \rceil \sqsubseteq P$ and $\vDash \{A\}\, \mathsf{pif}(s, t_1, t_0)\, \{P\}$, as required.

Case Fix. Suppose that $\vdash \{A\}\, \mathsf{fix}_\sigma\, (s)\, \{\mathsf{Y}_{\mathsf{Rel}^+(\sigma)}\, (P)\}$ for $\sigma = \sigma_1 \to \sigma_2$. Take a substitution pair $\langle \theta, \mathbb{S} \rangle$ satisfying $\vDash \{A\}\, \theta\, \{\mathbb{S}\}$. By the IH on the premise of the proof rule, $\vDash \{A\}\, s\, \{P\}$ and $\mathbb{S}, A \vDash \lceil s \rceil \sqsubseteq P$. By monotonicity of $\mathsf{Y}_{\mathsf{Rel}^+(\sigma)}$, it holds that $\mathbb{S}, A \vDash \mathsf{Y}_{\mathsf{Rel}^+(\sigma)}(\lceil s \rceil) \sqsubseteq \mathsf{Y}_{\mathsf{Rel}^+(\sigma)}(P)$. We conclude that $\vDash \{A\}\, \mathsf{fix}_\sigma\, (s)\, \{\mathsf{Y}_{\mathsf{Rel}^+(\sigma)}\, (P)\}$.

Case Conseq. Suppose $\vdash \{A\}\, s\, \{Q\}$. Note that we have $\vDash A \Rightarrow B$ and $B \vDash P \sqsubseteq Q$. By the IH on the remaining premise of the rule, $\vDash \{B\}\, s\, \{P\}$ and $B \vDash \lceil s \rceil \sqsubseteq P$. It follows from $B \vDash P \sqsubseteq Q$ that $B \vDash \lceil s \rceil \sqsubseteq Q$. Finally, $\vDash A \Rightarrow B$ gives us the result, $A \vDash \lceil s \rceil \sqsubseteq Q$ and thus $\vDash \{A\}\, s\, \{Q\}$. □

6.3.3 Completeness

Definition 6.22 (Cook expressivity, Cook, 1978)**.** *We say that a higher-order logic is* Cook expressive *if, the strongest postcondition $\lceil e \rceil$ is expressible by an assertion in the logic for each program expression e.*

Note that higher-order logic is Cook expressive under the continuous interpretation; the strongest postconditions do not contain implication or negation.

Theorem 6.23 (Relativised completeness)**.** *If a higher-order logic is Cook expressive, then $\vDash \{A\}\, e\, \{P\}$ implies $\vdash \{A\}\, e\, \{P\}$, for all closed expressions e.*

Proof. We prove the following sequence:

1. $\vDash \{A\}\, e\, \{P\}$ by assumption
2. $A \vDash \lceil e \rceil \sqsubseteq P$ by the definition of the semantics (thanks to Cook expressivity)
3. $\vdash \{\top\}\, e\, \{\lceil e \rceil\}$ by Lemma 6.13
4. $\vdash \{A\}\, e\, \{\lceil e \rceil\}$ by left-monotonicity
5. $\vdash \{A\}\, e\, \{P\}$ by right-monotonicity

□

6.4 The HoCHC fragment

If we restrict the precondition to HoCHC definite clauses and the postcondition to goal clauses, then we obtain a HoCHC-like fragment of higher-order Hoare logic.

Example 6.24. Let us consider the following $\mathrm{PCF_v}$ program:

$$\textsc{Sum} := \mathsf{fix}\,(\lambda f.\,\lambda n.\,\mathsf{if}\; n \leq 0\;\mathsf{then}\;0\;\mathsf{else}\;n + f\,(n-1))$$

In our syntax, this is:

$$\textsc{Sum} := \mathsf{fix}\,(\lambda f.\,\lambda n.\,\mathsf{if}(n \leq 0, 0, n + f\,(n-1)))$$

Let A be the set of definite clauses in the unknown $\mathit{Sum} : \mathsf{Rel}^+(\iota \to \iota)$:

$$\begin{aligned} \forall n\,m.\,n \leq 0 \wedge m = 0 \quad &\Rightarrow \quad \mathit{Sum}\,n\,m \\ \forall n\,m.\,n > 0 \wedge \exists p.\,\mathit{Sum}\,(n-1)\,p \wedge m = n + p \quad &\Rightarrow \quad \mathit{Sum}\,n\,m \end{aligned}$$

Let B be the set of definite clauses in the unknown $F : \mathsf{Rel}^+((\iota \to \iota) \to \iota \to \iota)$:

$$\begin{aligned} \forall f'\,n\,m.\,n \leq 0 \wedge m = 0 \quad &\Rightarrow \quad F\,f'\,n\,m \\ \forall f'\,n\,m.\,n > 0 \wedge \exists p.\,f'\,(n-1)\,p \wedge m = n + p \quad &\Rightarrow \quad F\,f'\,n\,m \end{aligned}$$

The following proof shows that any solution of $\textsc{Sum}$ is an invariant of Sum:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta_1 \qquad \dfrac{}{\{A \wedge B \wedge n \leq 0\}\,0\,\{\lambda z.\,z = 0\}}\,(\mathsf{Const}) \qquad \Delta_2}{\{A \wedge B\}\,\mathsf{if}\; n \leq 0\;\mathsf{then}\;0\;\mathsf{else}\;n + f\,(n-1)\,\{\lambda m.\,(n \leq 0 \wedge m = 0) \vee (n < 0 \wedge \exists p.\,f'\,(n-1)\,p \wedge m = n + p)\}}\,(\mathsf{Cond})(\mathsf{Conseq})}{\{A \wedge B\}\,\mathsf{if}\; n \leq 0\;\mathsf{then}\;0\;\mathsf{else}\;n + f\,(n-1)\,\{F\,f'\,n\}}\,(\mathsf{Abs})}{\{A \wedge B\}\,\lambda n.\,\mathsf{if}\; n \leq 0\;\mathsf{then}\;0\;\mathsf{else}\;n + f\,(n-1)\,\{F\,f'\}}\,(\mathsf{Abs})}{\{A \wedge B\}\,\lambda f.\,\lambda n.\,\mathsf{if}\; n \leq 0\;\mathsf{then}\;0\;\mathsf{else}\;n + f\,(n-1)\,\{F\}}\,(\mathsf{Fix})$$

where we use $\mathsf{Y}(F) = \mathit{Sum}$, and in the Conseq rule that

$$A \wedge B \vDash (\lambda m.\,(n \leq 0 \wedge m = 0) \vee (n < 0 \wedge \exists p.\,f'\,(n-1)\,p \wedge m = n + p) \sqsubseteq F\,f'\,n.$$

Subproof Δ_1 proves $\{A \wedge B\}\,n \leq 0\,\{\lambda m.\,m = (n \leq 0)\}$ as follows:

$$\dfrac{\dfrac{\dfrac{}{\{A \wedge B\}\,\_ \leq \_\,\{\lambda x\,y\,m.\,m = (x \leq y)\}}\,(\mathsf{Const}) \qquad \dfrac{}{\{A \wedge B\}\,n\,\{\lambda m.\,m = n\}}\,(\mathsf{Varl})}{\{A \wedge B\}\,n \leq \_\,\{\lambda y\,m.\,m = (n \leq y)\}}\,(\mathsf{Appl}) \qquad \dfrac{}{\{A \wedge B\}\,0\,\{\lceil 0 \rceil\}}\,(\mathsf{Const})}{\{A \wedge B\}\,n \leq 0\,\{\lambda m.\,m = (n \leq 0)\}}\,(\mathsf{Appl})$$

Subproof Δ_2 proves $\{A \wedge B \wedge n > 0\}\,n + f\,(n-1)\,\{\lambda m.\,\exists p.\,f'\,(n-1)\,p \wedge m = n + p\}$, where we write C as shorthand for $A \wedge B \wedge n > 0$:

$$\dfrac{\dfrac{\dfrac{}{\{C\}\,\_+\_\,\{\lambda x\,y\,m.\,m = x + y\}}\,(\mathsf{Const})\quad \dfrac{}{\{C\}\,n\,\{\lambda m.\,m = n\}}\,(\mathsf{VarI})}{\{C\}\,n + \_\,\{\lambda y\,m.\,m = n + y\}}\,(\mathsf{Appl})\quad \dfrac{\dfrac{}{\{C\}\,f\,\{f'\}}\,(\mathsf{VarH})\quad \Delta_3}{\{C\}\,f\,(n-1)\,\{f'\,(n-1)\}}\,(\mathsf{Appl})}{\{C\}\,n + f\,(n-1)\,\{\lambda m.\,\exists p.\,f'\,(n-1)\,p \wedge m = n + p\}}\,(\mathsf{Appl})$$

Subproof Δ_3 proves $\{A \wedge B \wedge n > 0\}\,n - 1\,\{\lambda m.\,m = n - 1\}$, again with shorthand C for $A \wedge B \wedge n > 0$:

$$\dfrac{\dfrac{\dfrac{}{\{C\}\,\_-\_\,\{\lambda x\,y\,m.\,m = x - y\}}\,(\mathsf{Const})\quad \dfrac{}{\{C\}\,n\,\{\lambda m.\,m = n\}}\,(\mathsf{VarI})}{\{C\}\,n - \_\,\{\lambda y\,m.\,m = n - y\}}\,(\mathsf{Appl})\quad \dfrac{}{\{C\}\,1\,\{\lambda m.\,m = 1\}}\,(\mathsf{Const})}{\{C\}\,n - 1\,\{\lambda m.\,m = n - 1\}}\,(\mathsf{Appl})$$

Valid postconditions of $\mathrm{PCF_v}$ expressions can be modelled in HoCHC over the background theory of Linear Integer Arithmetic. For fixpoint-free expression e, we define a relational variable P_e through

$$P_e := \lceil e \rceil,$$

which is a well-defined HoCHC goal term. For fixpoint expression $\mathsf{fix}(\lambda f.\,e)$, we define

$$P_{\mathsf{fix}(\lambda f.\,e)} := \lceil e \rceil[f' \mapsto P_{\mathsf{fix}(\lambda f.\,e)}].$$

Models of a definite clause headed by P_e correspond to valid postconditions of $\mathrm{PCF_v}$ expression e. In particular, the strongest postcondition $\lceil e \rceil$ is the least model of P_e.

Example 6.25. Consider $\textsc{Sum} := \mathsf{fix}\,(\lambda f.\,\lambda n.\,\mathsf{if}\,n \leq 0\,\mathsf{then}\,0\,\mathsf{else}\,n + f\,(n-1))$ from our previous example. The above encoding of postconditions into HoCHC yields

$$P_{\textsc{Sum}} = \lambda n'\,r.\,(n' \leq 0 \wedge r = 0) \vee (n' > 0 \wedge \exists x.\,P_{\textsc{Sum}}\,(n' - 1)\,x \wedge r = n' + x)$$

after some simplification. Clearly, the least fixpoint of $P_{\textsc{Sum}}$ coincides with $\lceil \textsc{Sum} \rceil$.

6.5 Refinement types

There is a natural crossover between Hoare logic and *refinement types*, which are types endowed with predicates that further restrict the elements inhabiting a type (Freeman and Pfenning, 1991); a refinement type corresponds to a precondition when passed as function argument, while a return type resembles a postcondition. This relation between Hoare logic and refinement types goes back to Back and Wright's work on Refinement Calculus (1998).

It is not surprising that we can obtain refinement type systems within our higher-order Hoare proof system through a relational interpretation of the dependent arrow.

Define the following family of predicates $\mathbin{\wr}_\sigma : \mathsf{Rel}^-(\sigma) \to \mathsf{Rel}^+(\sigma) \to o$ by induction on the sort σ, written as infix between its arguments of sort $\mathsf{Rel}^-(\sigma)$ and $\mathsf{Rel}^+(\sigma)$, resp.:

- When $\sigma = \iota$, $M \mathbin{\wr} R := R\,M$.
- When $\sigma = \sigma_1 \to \cdots \to \sigma_m \to \iota$, $M \mathbin{\wr} R := \forall x_1 \ldots x_m.\, M\,x_1 \ldots x_m \Rightarrow R\,x_1 \ldots x_m$.

Assume the operator $\mathbin{\wr}$ binds its arguments more tightly than implication. Note that $M \mathbin{\wr} R$ is read as R *refines* M, and the subscript is omitted for brevity.

We define the family of terms $\mathbb{T}$ (corresponding to types, $\mathbb{T}_\sigma$ for the subset of sort $\mathsf{Rel}^+(\sigma)$), writing T, T_i for elements of $\mathbb{T}$:

$$\frac{}{\Delta \vdash \lambda v.\, \varphi : \mathsf{Rel}^+(\iota) \in \mathbb{T}}\ \Delta, v : \iota \vdash \varphi : o \in Fm$$

$$\frac{\Delta \vdash T_1 : \mathsf{Rel}^+(\sigma_1) \in \mathbb{T} \qquad \Delta, x : \mathsf{Rel}^-(\sigma_1) \vdash T_2 : \mathsf{Rel}^+(\sigma_2)}{\Delta \vdash \lambda x\,\overline{z}.\, x \mathbin{\wr} T_1 \Rightarrow T_2\,\overline{z} : \mathsf{Rel}^+(\sigma_1 \to \sigma_2) \in \mathbb{T}}$$

We use the following notation/shorthands:

$$\begin{aligned} \{v : \iota \mid \varphi\} &:= \lambda v.\, \varphi \\ \Pi x : T_1.\, T_2 &:= \lambda x\,\overline{z}.\, x \mathbin{\wr} T_1 \Rightarrow T_2\,\overline{z} \\ A_1, A_2 &:= A_1 \wedge A_2 \end{aligned}$$

Suppose we have a constant fail for every sort that denotes failure (divergence). Furthermore, we define a refinement type $\mathbb{T}(c) : \mathsf{Rel}^+(\sigma)$ for all other constants $c : \sigma$:

$$\begin{aligned} \mathbb{T}(n) &:= \{v : \iota \mid v = n\} \\ \mathbb{T}(\leq) &:= \Pi v_1 : \mathbb{N}.\, \{\Pi v_2 : \mathbb{N}.\, \{v_3 : \iota \mid (v_1 \leq v_2 \wedge v_3 = 1) \vee (v_1 > v_2 \wedge v_3 = 0)\}\} \\ \mathbb{T}(+) &:= \Pi v_1 : \mathbb{N}.\, \{\Pi v_2 : \mathbb{N}.\, \{v_3 : \iota \mid v_1 + v_2 = v_3\}\} \\ \mathbb{T}(-) &:= \Pi v_1 : \mathbb{N}.\, \{\Pi v_2 : \mathbb{N}.\, \{v_3 : \iota \mid v_1 \dot{-} v_2 = v_3\}\} \end{aligned}$$

We write $\mathbb{N}$ for $\{v : \iota \mid \mathsf{true}\}$. Note that this definition resembles the respective relational lifts of these constants.

It is not immediately clear these types have (meaningful) continuous interpretations, due to the implications. However, variables only occur 'positively' in implications, so refinement type do have continuous interpretations. It follows that they are suitable postconditions.

The following rules are admissible in the proof system:

$$\frac{}{\{A\}\, c\, \{\mathbb{T}(c)\}}$$

$$\frac{}{\{A, x \mathbin{:} \{v : \iota \mid \varphi\}\}\, x\, \{\{v : \iota \mid v = x\}\}}$$

$$\frac{}{\{A, x \mathbin{:} \Pi y : T_1.\, T_2\}\, x\, \{\Pi y : T_1.\, T_2\}}$$

$$\frac{\{A, x \mathbin{:} T_1\}\, e\, \{T_2\}}{\{A\}\, \lambda x.\, e\, \{\Pi x : T_1.\, T_2\}}\ x \notin \mathsf{FV}(A)$$

$$\frac{\{A\}\, e_1\, \{\Pi x : T_1.\, T_2\} \qquad \{A\}\, e_2\, \{T_1\}}{\{A\}\, e_1\, e_2\, \{T_2\}}\ x \notin \mathsf{FV}(T_2)$$

$$\frac{\{A, x \mathbin{:} T\}\, e\, \{T\}}{\{A\}\, \mathsf{fix}\, (\lambda x.\, e)\, \{T\}}\ x \notin \mathsf{FV}(T)$$

$$\frac{\{A, x > 0\}\, e_1\, \{T\} \qquad \{A, x = 0\}\, e_0\, \{T\}}{\{A\}\, \mathsf{if}\, x\, \mathsf{then}\, e_1\, \mathsf{else}\, e_0\, \{T\}}$$

$$\frac{\vDash A \Rightarrow \mathsf{false}}{\{A\}\, \mathsf{fail}\, \{T\}}$$

$$\frac{\{A\}\, e\, \{T_1\} \qquad A \vDash T_1 \sqsubseteq T_2}{\{A\}\, e\, \{T_2\}}$$

These rules are essentially the type system by Unno and Kobayashi (2009), from which the following example is adapted.

Example 6.26. Consider the incremental function $\mathsf{inc} : \iota \to \iota$, which can be defined in $\mathrm{PCF_v}$ as $\lambda x.\, x + 1$. This function inhabits refined type $T_{\mathsf{inc}} := \Pi v_1 : \mathbb{N}.\, \{v_2 : \iota \mid v_2 = v_1 + 1\}$. Now, we can prove that term $\mathsf{inc}\, 3$ inhabits type $\{v : \iota \mid v = 4\}$:

$$\dfrac{\dfrac{(\mathsf{RConst})\ \dfrac{}{\{\mathsf{true}\}\, \mathsf{inc}\, \{\Pi x : \mathbb{N}.\{v : \iota \mid v = x + 1\}\}}}{(\mathsf{RSub})\ \dfrac{}{\{\mathsf{true}\}\, \mathsf{inc}\, \{\Pi x : \{v' : \iota \mid v' = 3\}.\{v : \iota \mid v = 4\}\}}} \quad (\mathsf{RConst})\ \dfrac{}{\{\mathsf{true}\}\, 3\, \{\{v : \iota \mid v = 3\}\}}}{(\mathsf{RApp})\ \{\mathsf{true}\}\, \mathsf{inc}\, 3\, \{\{v : \iota \mid v = 4\}\}}$$

Chapter 7

Conclusion

The introduction posed two questions pertaining to the adequacy of HoCHC as a setting for higher-order program verification:

(i) Is the fragment sufficiently expressive to capture the problems typically studied by the higher-order program verification community?

(ii) Are there (relatively) complete solution methods (e.g. algorithms that are sound and complete in the presence of an oracle for first-order constraint solving)?

We first addressed (i) in Chapter 3, where we have seen that there exists a continuous semantics for HoCHC that is well-defined and well-behaved. On top of that, the associated continuous HoCHC problem is equivalent to the standard problem. The continuous semantics closely aligns with the operational semantics of higher-order programs and is, thus, a natural choice of semantics for higher-order program verification.

Again addressing (i), Chapter 4 established connections between HoCHC and higher-order model checking—an archetypical representative of the problems studied by the higher-order verification community. We indicate that the HoMC safety problem can be reduced to coinductive HoCHC—as well as to (inductive) HoCHC—through an encoding of HoRS into HoCHC logic programs. Furthermore, we show that this encoding can be used to reduce the open HoRS equivalence problem to coinductive HoCHC over Maher (1988)'s complete and decidable theory of trees. The most interesting of further directions following from this chapter is whether coinductive HoCHC over a decidable background theory is semi-decidable. If it is, then this chapter describes a full decision procedure for the HoRS equivalence problem and the recursively equivalent $\lambda\mathbf{Y}$-calculus Böhm tree equivalence problem.

Even though there seems to be a disparity between HoMC and inductive HoCHC, coinductive HoCHC provides a promising alternative that may allow full incorporation of HoMC into HoCHC. The relationship of HoCHC and coinductive HoCHC to higher-order fixpoint logic—another notable approach to higher-order program verification—remains to be formalised.

For our reduction of HoRS equivalence to coinductive HoCHC, we proved the existence of an algorithm that generates the $\bot$-free transform of a HoRS. This algorithm allows us to decide productivity of HoRS terms; the presence of b in the value tree of the $\bot$-free transform signals unproductivity and its absence productivity. This result subsumes a first-order result by Fu (2017), who developed a second-order type system to represent nontermination in (first-order) rewrite systems. Productivity checking in their system is decidable via a mapping to the $\lambda\mathbf{Y}$-calculus.

Our continuous semantics of HoCHC admits a sound and refutation-complete solution method for the HoCHC problem over a semi-decidable background theory (Chapter 5), which addresses (ii). Our proof system based on SLD-resolution was the first resolution-based solution method at the time of writing. The next step in this line of work is the implementation of a HoCHC solver, building on Charalambidis et al. (2013)'s solver for a positive existential fragment of higher-order Horn clauses without constraints. We also aim to identify decidable fragments of HoCHC for which a full solver can be obtained.

Finally, Chapter 6 developed an axiomatic basis for relations as higher-order program invariants in the form of a Hoare logic. Our approach to Hoare logic for higher-order programs is new by virtue of a combination of a higher-order assertion logic and a relatively complete proof system. Restrictions on the syntax of preconditions and postconditions allow us to obtain useful program logics like HoCHC. The strongest postconditions that are key to relative completeness of our Hoare logic are expressible in HoCHC; we expect valid Hoare triples to correspond to HoCHC invariants (overapproximations of a program).

Our main motivation for studying HoCHC was to provide a unifying framework for higher-order program verification. After relating higher-order model checking to HoCHC in Chapter 4, we show in Chapter 6 that Unno and Kobayashi (2009)'s refinement type system can be modelled in our Hoare logic, which allows us to relate refinement type inference to HoCHC via the above correspondence.

This thesis makes a case for the HoCHC as a framework for higher-order program verification. It is semantically and algorithmically robust; our continuous interpretation is equivalent to the standard and monotone interpretations, and there exist sound and refutation-complete solution methods. On top of that, HoCHC allows for the reduction of certain HoMC and RTI problems and has applications in Hoare logic, which testifies to its versatility.

We conclude that the HoCHC fragment of higher-order logic is well worth studying in the context of higher-order program verification.

7.1 Further directions and related work

7.1.1 Coinductive HoCHC

Our work on coinductive HoCHC in Chapter 4 prompts several directions of further research, many of which pertain to decidability of (monotone) coinductive HoCHC over a decidable background theory.

7.1.1.1 Decidability

Refutation-complete resolution proof systems that semi-decide (inductive) HoCHC over a decidable background theory—like the proof system presented in Chapter 5—do not simply carry over to coinductive HoCHC, as proofs for coinductive programs may have infinite length. This is already apparent in the HoRS determined by single rewrite rule $S = a\,S$, whose logic program $\{R_S = \lambda r.\, \exists r_1.\, (a\,r_1 = r) \wedge R_S\,r_1\}$ contains no 'base cases', so that any proof must be infinite.

For first-order Horn clauses, CoLP (Coinductive Logic Programming, Gupta et al., 2007; Simon et al., 2007) provides an approach to computing solutions for infinite sequences of reductions. Resolution for coinductive logic programs relies on loop detection (see e.g. Komendantskaya et al., 2016; Komendantskaya and Li, 2018), which might argue against semi-decidability of coinductive HoCHC; loop detection in coinductive HoCHC may not be any simpler than loop detection directly on HoRS.

In CoLP, if there exists a circular unifier—and thus a loop—then an answer is found. CoLP is sound but incomplete. It terminates only if the term computable at infinity is a rational term (i.e. a term corresponding to a tree with a finite number of distinct subtrees). However, progress has been made in the proof search for (first-order)

conductive predicates by Komendantskaya and Johann (2015), as well as Basold et al. (2019), via a framework called *coinductive uniform proofs* (CUP).

Basold et al. (2019) identify fragments of coinductive clauses that "give a precise proof-theoretic characterisation to the different kinds of [first-order] coinduction described in the literature", of which coinductive higher-order hereditary Harrop clauses are the most subsuming. Coinductive uniform proofs are proved to be sound relative to an intuitionistic first-order logic. A notable result is the extension of proof search to some irregular proofs, which seem to cover encoded HoRS of order-1. CUP is incomplete, however, because it is intuitionistic and lacks an admissible cut rule.

Unfortunately, it turns out that cut cannot be eliminated from coinductive calculi (Komendantskaya et al., 2020). As a consequence, we cannot hope to prove all relevant theorems analytically, and the situation for semi-decidability of coinductive HoCHC over a (semi-)decidable background theory looks, perhaps, bleak.

HoRS equivalence problem. In case of semi-decidability of coinductive HoCHC, we would obtain a full decision procedure for the (open) HoRS equivalence problem (see e.g. Clairambault and Murawski, 2013; Ong, 2015) by dovetailing two coinductive HoCHC instances, as demonstrated in Chapter 4.6. A weaker result also suffice: (a) if coinductive HoCHC over the Maher theory of trees is semi-decidable, or (b) if the image of our HoRS-to-HoCHC encoding over the Maher theory (Chapter 4.3) lives in a semi-decidable fragment of coinductive HoCHC, as in Theorem 4.44.

$\lambda\mathbf{Y}$-calculus Böhm tree equivalence problem. The HoRS equivalence problem is recursively equivalent to the long-standing open $\lambda\mathbf{Y}$-calculus Böhm tree equivalence problem, which asks whether the Böhm trees of two given $\lambda\mathbf{Y}$-terms are equal (Clairambault and Murawski, 2013). Thus, semi-decidability of coinductive HoCHC would also allow us to decide this problem.

Note that the closely related $\lambda\mathbf{Y}$-calculus word problem is undecidable (Statman, 2004); it is generally undecidable whether two closed $\lambda\mathbf{Y}$-terms are $\beta\eta\mathbf{Y}$-equivalent, which is relevant because HoRS are programs of a simply sorted (typed) $\lambda\mathbf{Y}$-calculus constructed from a set of uninterpreted function symbols. This may indicate that the HoRS equivalence problem—and thus coinductive HoCHC over the Maher theory—is undecidable. However, HoRS define the same set of trees as ground-type $\lambda\mathbf{Y}$-terms with free variables (corresponding to terminal symbols) of order at most 1 (Salvati

and Walukiewicz, 2014) and are, thus, a strict subsystem of the $\lambda\mathbf{Y}$-calculus; the HoRS equivalence problem may still be decidable.

7.1.1.2 Solving HoMC problems

Recall that, given a higher-order recursion scheme $\mathcal{G}$ and a property φ expressed in some logic, the higher-order model checking problem seeks to answer the question whether the (potentially infinite) tree generated by $\mathcal{G}$ satisfies φ (Ong, 2006). The problem is decidable for MSO properties.

Safety. The arguably simplest version of this problem is the *safety problem*, where property φ is a safety property expressed by—say—a trivial automaton. Although we already know that the HoMC safety problem can be solved via a decidable higher-order Datalog fragment of HoCHC (Wagner, 2019, private communication), we can also reduce it to coinductive HoCHC in the following way.

Let $\mathcal{G} = \langle \mathcal{N}, \Sigma, \mathcal{R}, S \rangle$ be the HoRS. Tag the states of the automaton onto the relational variables R_F we generate from nonterminals $F \in \mathcal{N}$—e.g. R_F^q for state q—where we switch to a different state if $\$ \in \Sigma$ and stay in the same state otherwise (see Chapter 4.3). Each R_F^q then denotes a smaller relation than R_F. If $R_S^{q_0}$ is assigned a larger relation than $\lambda r.\, 0$ for initial state q_0 (in which case it is assigned a characteristic function of a single tree), then the trivial automaton runs on $[\![\mathcal{G}]\!]$ without crashing, and $[\![\mathcal{G}]\!]$ is accepted.

Liveness. Another class of HoMC problems is given by Büchi automata, which accept a tree $[\![\mathcal{G}]\!]$ if there exists a run of the automaton on $[\![\mathcal{G}]\!]$ in which all infinite paths in the run tree contain an infinitely occurring final state. Finally, for the full class of HoMC *liveness problems* (Ong, 2006) we would need to consider alternating parity tree automata or equivalent.

Our motivation for moving from inductive to coinductive HoCHC was the ability to check entire trees for the HoRS equivalence problem. This is ideal for liveness, for which checking a finite prefix of a tree is also inconclusive.

In addition to tagging states on to the relational variables of our encoded HoRS-to-HoCHC logic program, we would need to encode the acceptance condition, perhaps via an additional parameter—giving us $R_F^{q,b}$ for state q and boolean b—to denote whether a final state or even priority has been seen since the last unfolding. Alternatively,

it might be possible to tag modal μ-calculus formulas on to the relational variables instead of pairs q, b and change 'state' (i.e. formula) accordingly.

Thus, we anticipate that the HoMC liveness problem can be reduced to coinductive HoCHC over Maher's theory too, similar to how first-order liveness can be "elegantly" verified via (first-order) CoLP, according to Gupta et al. (2007).

7.1.1.3 Relation to HFL

Higher-order fixpoint logic (HFL) is a higher-order extension of the modal μ-calculus (Viswanathan and Viswanathan, 2004) with mutual translations between HoMC and HFL model checking (Kobayashi et al., 2017). Like HoCHC, HFL deals with functions in continuation-passing style and is semi-decidable.

HoCHC roughly corresponds to a fragment of (extended) HFL without modal operators and fixpoint alternations, where solvability of HoCHC is reduced to the validity of (extended) HFL formulas. HoCHC (unsolvability) captures the negation of Kobayashi et al. (2018)'s *may-reachability*, which is *non-reachability*: given a program P and event a, P never raises a. For coinductive HoCHC, it is solvability that is potentially semi-decidable. Thus, although we have not proved this formally, it seems that coinductive HoCHC corresponds to the negation of *must-reachability*: P does not always raise a. This relation remains to be clarified.

7.1.2 Intensional higher-order logic programming

An important characteristic of HoCHC is that it is *extensional*. That is, predicates are solely identified by the arguments for which they are true. This sets our work apart from e.g. λProlog (Miller and Nadathur, 1986) and HiLog (Chen et al., 1989), both of which are *intensional* higher-order languages.

Example 7.1 (Charalambidis et al., 2013)**.** Consider this λProlog program:

```
r p.
p X :- q X
q X :- p X
```

The goal `r q` is false for this program. Any extensional interpretation, however, would recognise that `p` and `q` both denote the same semantic element.

In HoCHC, `p` and `q` would be assigned the same element, so that $\mathcal{C}[\![\mathtt{p}]\!] = \mathcal{C}[\![\mathtt{q}]\!]$. The clause `r p` is interpreted as '`r` holds for whatever `p` is'. Therefore, `r p` would imply `r q`, and `r q` would hold.

Intensionality complicates the interpretation of uninstantiated relations. How do we interpret a goal `R john mary` where `R` is free? For this to be a meaningful goal, the programmer would need to specify the predicates `R` is allowed to range over (Nadathur and Miller, 1998, p. 50), because the semantic domain depends on the available terms rather than a domain-theoretic interpretation of sorts.

The syntax of λProlog is based on an intuitionistic theory of higher-order hereditary Harrop formulas, which generalise higher-order Horn clauses to allow nested implication. The result is a highly expressive language that can still facilitate extensionality. Unfortunately, leaving the realm of classical logic comes at the cost of extensionality of the system as a whole.

The semantics of functional programming languages tends to be extensional, which motivates our choice for—the extensional—HoCHC as a unifying framework for higher-order verification over e.g. the richer (perhaps too rich), intuitionistic higher-order hereditary Harrop clauses.

Despite this fundamental disparity between HoCHC and λProlog, our goal-oriented and syntax-driven resolution proof system may remind the reader of the *uniform proofs* underpinning λProlog (Miller et al., 1991). These uniform proofs are a class of cut-free sequent proofs that, too, are goal-oriented and syntax-driven: "the notion of a uniform proof reflects the search instructions associated with the logical connectives."

For higher-order Horn clauses (i.e. in classical logic), there exists a uniform proof for a goal if, and only if, this goal is valid. Thus, our resolution refutations correspond to sequent proofs. This correspondence between proofs and resolution dates back to Girard et al. (1989)—who suggested using the cut rule to model resolution for Horn formulas—and can be understood in the wider context of Curry-Howard. Both sequent proofs and SLD-refutations represent the operational semantics of a program. Note, however, that uniform proofs are much more powerful than resolution because they do not rely on classical logic.

The work in Chapter 5 is theoretical; we have shown there exists a sound and refutation-complete proof system for HoCHC, but have not ventured into implementation. The implementation from Charalambidis et al. (2013) for higher-order Horn

clauses was also largely a proof of concept, but progress has been made in efficient resolution proof systems in the context of (the intensional) λProlog. Work by Dunchev et al. (2015) is believed to have been the most efficient implementation of λProlog at the time of publication.

The Curry-Howard correspondence and λProlog may provide further insights into higher-order Horn clauses, as demonstrated by e.g. Fu and Komendantskaya (2017), who interpret Horn formulas as types to show completeness of SLD-resolution.

7.1.3 Higher-order Hoare logic

Our approach to Hoare logic for higher-order programs distinguishes itself from the literature through an aggregate of three components:

(i) our assertion logic is higher-order (cf. Honda et al., 2006, 2014),

(ii) our proof system is relatively complete (cf. Régis-Gianas and Pottier, 2008), and

(iii) we use a fully abstract model of $\mathrm{PCF}_{\mathrm{v}}$ (cf. Reus and Streicher, 2011).

The main contribution is a truly higher-order assertion logic for a functional programming language that does not preclude relative completeness.

Perhaps this result has come at the cost of the expressivity of our programming language, call-by-value PCF. Although we appreciate the 'pureness' of the higher-order assertion logic used in Chapter 6, we recognise that practical applications may require sacrificing some of this logical purity for expressivity.

An earlier strain of work on Hoare logic for higher-order programs was conducted in the 70s and 80s (Clarke, 1977; German et al., 1984; Damm and Josko, 1984; Goerdt, 1985). As one of the field's pioneers—Clarke (1977)—proved, there does not exist a sound and complete Hoare logic for Algol-like or Pascal-like programming languages, which are equipped with recursive, higher-order procedures, global variables, and static scoping. However, Clarke's proof does not contradict our result, because he assumes that the underlying assumption language is first-order. Damm and Josko (1984) first pointed out that Clarke's result does not necessarily hold for higher-order assertion languages.

Reus and Streicher (2005) studied first-order imperative programs with access to a higher-order store, which facilitates recursion through stores rather than fixpoints.

Their Hoare logic is sound; whether it is relatively complete is unknown, to the best of our knowledge. Reus and Streicher (2011) later deviated from Hoare logic for a sound and complete program logic for functional programs.

The programs studied by Honda et al. (2006, 2014) are essentially a higher-order counterpart of Reus and Streicher (2005)'s imperative programs. Their system is sound and relatively complete for an imperative version of $\mathrm{PCF}_{\mathrm{v}}$ with global references that can store higher-order procedures.

7.1.3.1 Extensions to other languages

Régis-Gianas and Pottier (2008) made the trade-off between completeness and expressivity in favour of a more expressive language that includes algebraic data types. Incorporating algebraic data types into $\mathrm{PCF}_{\mathrm{v}}$ and higher-order logic is possible as long as long as we can pattern-match on elements of these new types. We expect that such an incorporation would preserve soundness and relative completeness of our proof system.

Another natural line of work extends to call-by-name languages, starting with PCF proper. Régis-Gianas and Pottier (2008) explicitly state that their proof system is not sound for call-by-name. However, this is at least partly due to the definitions of their semantic domains, which do not contain divergent arguments. Beyond call-by-name PCF, one could consider Reynolds (1997)'s Idealized Algol. Its strict separation between expressions and commands leans itself well for program logics, like Reynolds's specification logic.

7.1.3.2 Higher-order relational Hoare logic

Some correctness properties from the literature cannot be captured by Hoare triples, particularly properties that relate pairs of executions, be it different programs or the same program with different inputs. Program equivalence is a clear example of such a property. To this end, relational Hoare logic was developed (Benton, 2004; Barthe et al., 2009; Barthe, 2020). It is a natural step to develop a relational version of our higher-order Hoare logic, in view of the relational form of ambient higher-order logic.

Bibliography

Davide Ancona. Regular corecursion in Prolog. *Computer Languages, Systems & Structures*, 39, 12 2013. doi: 10.1145/2245276.2232088. 4

Peter B. Andrews. Resolution in type theory. In *Automation of Reasoning*, pages 487–507. Springer, 1971. 118

Krzysztof R. Apt and Ernst-Rüdiger Olderog. Fifty years of Hoare's logic. *CoRR*, abs/1904.03917, 2019. URL http://arxiv.org/abs/1904.03917. 10

Krzysztof R. Apt, Frank S. de Boer, and Ernst-Rüdiger Olderog. *Verification of sequential and concurrent programs*. Springer Science & Business Media, 2010. 11

Ralph-Johan Back and Joakim Wright. *Refinement Calculus: A Systematic Introduction*. Texts in Computer Science. Springer New York, 1998. ISBN 9780387984179. URL https://books.google.co.uk/books?id=fRWQe23oB7kC. 181

Gilles Barthe. An introduction to relational program verification. working draft, 2020. URL https://software.imdea.org//~gbarthe/__introrelver.pdf. 192

Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. *SIGPLAN Not.*, 44(1):90–101, January 2009. ISSN 0362-1340. doi: 10.1145/1594834.1480894. URL https://doi.org/10.1145/1594834.1480894. 192

Henning Basold, Ekaterina Komendantskaya, and Yue Li. Coinduction in uniform: Foundations for corecursive proof search with Horn clauses. *Lecture Notes in Computer Science*, page 783–813, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-17184-1_28. 4, 187

Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, page 14–25, New York, NY,

USA, 2004. Association for Computing Machinery. ISBN 158113729X. doi: 10.1145/964001.964003. URL https://doi.org/10.1145/964001.964003. 12, 192

Christoph Benzmüller. Comparing approaches to resolution based higher-order theorem proving. *Synthese*, 133(1/2):203–235, 2002. ISSN 00397857, 15730964. URL http://www.jstor.org/stable/20117301. 118

Christoph Benzmüller and Michael Kohlhase. Extensional higher-order resolution. In Claude Kirchner and Hélène Kirchner, editors, *Automated Deduction — CADE-15*, pages 56–71, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-69110-5. 118

Martin Berger, Kohei Honda, and Nobuko Yoshida. A logical analysis of aliasing in imperative higher-order functions. *J. Funct. Program.*, 17(4-5):473–546, 2007. doi: 10.1017/S0956796807006417. URL https://doi.org/10.1017/S0956796807006417. 12

Tewodros A. Beyene, Corneliu Popeea, and Andrey Rybalchenko. Solving existentially quantified Horn clauses. In *Proceedings of the 25th International Conference on Computer Aided Verification - Volume 8044*, CAV 2013, page 869–882, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 9783642397981. 2

Nikolaj Bjørner, Kenneth McMillan, and Andrey Rybalchenko. Program verification as Satisfiability Modulo Theories. In Pascal Fontaine and Amit Goel, editors, *SMT 2012. 10th International Workshop on Satisfiability Modulo Theories*, volume 20 of *EPiC Series in Computing*, pages 3–11. EasyChair, 2013a. doi: 10.29007/1l7f. 1, 2, 7

Nikolaj Bjørner, Kenneth McMillan, and Andrey Rybalchenko. Higher-order program verification as Satisfiability Modulo Theories with algebraic data-types. *CoRR*, abs/1306.5264, 2013b. URL http://arxiv.org/abs/1306.5264. 2

Nikolaj Bjørner, Arie Gurfinkel, Kenneth McMillan, and Andrey Rybalchenko. Horn clause solvers for program verification. In *Fields of Logic and Computation II*, pages 24–51. Springer, 2015. doi: 10.1007/978-3-319-23534-9_2. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nbjorner-yurifest.pdf. 1, 2, 7

Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Recursion schemes and logical reflection. In *Proceedings of the 25th Annual IEEE*

Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom, pages 120–129. IEEE Computer Society, 2010. doi: 10.1109/LICS.2010.40. URL https://hal.archives-ouvertes.fr/hal-00479818. 113, 114, 115

Toby Cathcart Burn, C.-H. Luke Ong, and Steven J. Ramsay. Higher-order constrained Horn clauses for verification. *PACMPL*, 2(POPL):11:1–11:28, 2018. doi: 10.1145/3158099. a, 1, 8, 10, 25, 30, 33, 36, 38, 49, 67, 120

Angelos Charalambidis, Konstantinos Handjopoulos, Panagiotis Rondogiannis, and William W. Wadge. Extensional higher-order logic programming. *ACM Trans. Comput. Log.*, 14(3):21:1–21:40, 2013. doi: 10.1145/2499937.2499942. URL http://doi.acm.org/10.1145/2499937.2499942. 9, 13, 15, 22, 23, 34, 56, 59, 119, 120, 121, 122, 146, 147, 185, 189, 190

Weidong Chen, Michael Kifer, and David S. Warren. Hilog as a platform for database languages. In *Proceedings of the Second International Workshop on Database Programming Languages*, page 315–329, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1558600728. 189

Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940. ISSN 00224812. URL http://www.jstor.org/stable/2266170. 25

Pierre Clairambault and Andrzej S. Murawski. Böhm trees as higher-order recursive schemes. In Anil Seth and Nisheeth K. Vishnoi, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 91–102, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 978-3-939897-64-4. doi: 10.4230/LIPIcs.FSTTCS.2013.91. 7, 14, 110, 187

Edmund M. Clarke. Programming language constructs for which it is impossible to obtain good Hoare-like axiom systems. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, page 10–20, New York, NY, USA, 1977. Association for Computing Machinery. ISBN 9781450373500. doi: 10.1145/512950.512952. URL https://doi.org/10.1145/512950.512952. 191

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, September 2003. ISSN 0004-5411. doi: 10.1145/876638.876643. URL https://doi.org/10.1145/876638.876643. 7

Alain Colmerauer and Thi-Bich-Hanh Dao. Expressiveness of full first-order constraints in the algebra of finite or infinite trees. *Constraints*, 8(3):283–302, 2003. doi: 10.1023/A:1025675127871. URL https://hal.archives-ouvertes.fr/hal-00144924. 112

Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM Journal of Computing*, 1978. 11, 179

Bruno Courcelle. A representation of trees by languages I. *Theor. Comput. Sci.*, 6: 255–279, 1978. doi: 10.1016/0304-3975(78)90008-7. URL https://core.ac.uk/download/pdf/82601565.pdf. 6, 110

Werner Damm and Bernhard Josko. A sound and relatively complete axiomatization of Clarke's language L4. In Edmund M. Clarke and Dexter Kozen, editors, *Logics of Programs*, pages 161–175, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg. ISBN 978-3-540-38775-6. 191

Jaco de Bakker. Mathematical theory of program correctness. With the assistance of Arie de Bruin and Jeffery Zucker. Prentice-Hall International Series in Computer Science. Englewood Cliffs, New Jersey, etc.: Prentice-Hall International. XVII, 505 p. (1980)., 1980. 11

Frank S. de Boer and Cees Pierik. How to cook a complete Hoare logic for your pet OO language. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem-Paul de Roever, editors, *Formal Methods for Components and Objects*, pages 111–133, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30101-1. 11

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78800-3. 2

Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, August 1975. ISSN 0001-0782. doi: 10.1145/360933.360975. URL https://doi.org/10.1145/360933.360975. 11

Khalil Djelloul, Thi-Bich-Hanh Dao, and Thom Frühwirth. Theory of finite or infinite trees revisited. *Theory Pract. Log. Program.*, 8(04):431–489, Jul 2008. ISSN 1471-0684. doi: 10.1017/S1471068407003171. URL https://arxiv.org/abs/0706.4323. a, 112

Gilles Dowek. Higher-order unification and matching. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 1009–1062. Elsevier and MIT Press, 2001. doi: 10.1016/b978-044450813-3/50018-7. URL https://doi.org/10.1016/b978-044450813-3/50018-7. 123

Cvetan Dunchev, Ferruccio Guidi, Claudio Sacerdoti Coen, and Enrico Tassi. ELPI: Fast, embeddable, λProlog interpreter. In *Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning - Volume 9450*, LPAR-20 2015, page 460–468, Berlin, Heidelberg, 2015. Springer-Verlag. ISBN 9783662488980. doi: 10.1007/978-3-662-48899-7_32. URL https://doi.org/10.1007/978-3-662-48899-7_32. 191

Robert Bruce Findler and Matthias Felleisen. Contracts for higher-order functions. *SIGPLAN Not.*, 37(9):48–59, September 2002. ISSN 0362-1340. doi: 10.1145/583852.581484. URL https://doi.org/10.1145/583852.581484. 149

Robert W. Floyd. Assigning meanings to programs. *Mathematical aspects of computer science*, 19(19-32):1, 1967. URL http://www.cs.ucdavis.edu/~su/teaching/ecs240-s09/readings/FloydMeaning.pdf. 10, 15, 148

Tim Freeman and Frank Pfenning. Refinement types for ML. In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, PLDI '91, page 268–277, New York, NY, USA, 1991. Association for Computing Machinery. ISBN 0897914287. doi: 10.1145/113445.113468. URL https://doi.org/10.1145/113445.113468. 181

Peng Fu. Representing nonterminating rewriting with $\mathbf{F}_2^\mu$. *CoRR*, abs/1706.00746, 2017. URL http://arxiv.org/abs/1706.00746. 185

Peng Fu and Ekaterina Komendantskaya. Operational semantics of resolution and productivity in Horn clause logic. *Formal Aspects Comput.*, 29(3):453–474, 2017. doi: 10.1007/s00165-016-0403-1. URL https://doi.org/10.1007/s00165-016-0403-1. 191

Steven M. German, Edmund M. Clarke, and Joseph Y. Halpern. Reasoning about procedures as parameters. In Edmund Clarke and Dexter Kozen, editors, *Logics*

of Programs, pages 206–220, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg. ISBN 978-3-540-38775-6. 191

Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*, volume 7. Cambridge university press Cambridge, 1989. 190

Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931. 118

Andreas Goerdt. A Hoare calculus for functions defined by recursion on higher types. In Rohit Parikh, editor, *Logics of Programs*, pages 106–117, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg. ISBN 978-3-540-39527-0. 191

Gopal Gupta, Ajay Bansal, Richard Min, Luke Simon, and Ajay Mallya. Coinductive logic programming and its applications. In *Log. Program.*, pages 27–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-74610-2_4. URL https://personal.utdallas.edu/~gupta/iclp07paper.pdf. 3, 4, 69, 186, 189

Arie Gurfinkel, Temesghen Kahsai, Anvesh Komuravelli, and Jorge A. Navas. The SeaHorn verification framework. In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification*, pages 343–361, Cham, 2015. Springer International Publishing. ISBN 978-3-319-21690-4. 2

Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science (LICS 2008)*, pages 452–461. IEEE Computer Society Press, June 2008. 1, 6

Leon Henkin. Completeness in the theory of types. *J. Symbolic Logic*, 15(2):81–91, 06 1950. URL https://projecteuclid.org:443/euclid.jsl/1183730860. 28

C. A. R. Tony Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, October 1969. ISSN 0001-0782. doi: 10.1145/363235.363259. URL https://doi.org/10.1145/363235.363259. 10, 15, 148

C. A. R. Tony Hoare and Ronald H. Perrott. *Operating Systems Techniques*. Academic Press, 1972. ISBN 0123506506. 11

Kohei Honda, Martin Berger, and Nobuko Yoshida. Descriptive and relative completeness of logics for higher-order functions. In Michele Bugliesi, Bart Preneel,

Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 360–371, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1. 12, 16, 191, 192

Kohei Honda, Nobuko Yoshida, and Martin Berger. An observationally complete program logic for imperative higher-order functions. *Theor. Comput. Sci.*, 517: 75–101, January 2014. ISSN 0304-3975. doi: 10.1016/j.tcs.2013.11.003. URL http://dx.doi.org/10.1016/j.tcs.2013.11.003. 12, 191, 192

Gerard P. Huet. A mechanization of type theory. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, page 139–146, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc. 118

Martin Hyland and C.-H. Luke Ong. On full abstraction for PCF. *Inf. Comput.*, 163 (2):285–408, December 2000. ISSN 0890-5401. doi: 10.1006/inco.2000.2917. URL http://dx.doi.org/10.1006/inco.2000.2917. 159

Joxan Jaffar and Peter J. Stuckey. Semantics of infinite tree logic programming. *Theor. Comput. Sci.*, 46:141–158, Jan 1986. ISSN 0304-3975. doi: 10.1016/0304-3975(86)90027-7. 69

Petr Jancar. Decidability of DPDA language equivalence via first-order grammars. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 415–424, 2012. doi: 10.1109/LICS.2012.51. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1080.2859&rep=rep1&type=pdf. 110

D.C. Jensen and Tomasz Pietrzykowski. Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3(2):123 – 171, 1976. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(76)90021-9. URL http://www.sciencedirect.com/science/article/pii/0304397576900219. 118

Zhao Jianhua and Li Xuandong. Scope logic: An extension to Hoare logic for pointers and recursive data structures. In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *Theoretical Aspects of Computing – ICTAC 2013*, pages 409–426, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39718-9. 11

Roope Kaivola. On modal mu-calculus and Büchi tree automata. *Information Processing Letters*, 54(1):17–22, 1995. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(94)00227-P. URL https://www.sciencedirect.com/science/article/pii/002001909400227P. 114

Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43366-8. doi: 10.1007/3-540-45931-6_15. URL http://dx.doi.org/10.1007/3-540-45931-6_15. 6

Naoki Kobayashi. 10 years of the higher-order model checking project (extended abstract). In Ekaterina Komendantskaya, editor, *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 2:1–2:2. ACM, 2019. doi: 10.1145/3354166.3354167. URL https://doi.org/10.1145/3354166.3354167. 7

Naoki Kobayashi, Ryosuke Sato, and Hiroshi Unno. Predicate abstraction and CEGAR for higher-order model checking. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, page 222–233, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306638. doi: 10.1145/1993498.1993525. URL https://doi.org/10.1145/1993498.1993525. 7, 14

Naoki Kobayashi, Étienne Lozes, and Florian Bruse. On the relationship between higher-order recursion schemes and higher-order fixpoint logic. *SIGPLAN Not.*, 52(1):246–259, January 2017. ISSN 0362-1340. doi: 10.1145/3093333.3009854. URL http://www-kb.is.s.u-tokyo.ac.jp/~koba/papers/popl2017-long.pdf. 7, 14, 189

Naoki Kobayashi, Takeshi Tsukada, and Keiichi Watanabe. Higher-order program verification via HFL model checking. In Amal Ahmed, editor, *Programming Languages and Systems*, pages 711–738, Cham, 2018. Springer International Publishing. ISBN 978-3-319-89884-1. doi: 10.1007/978-3-319-89884-1_25. URL https://arxiv.org/abs/1710.08614. 7, 189

Ekaterina Komendantskaya and Patricia Johann. Structural resolution: a framework for coinductive proof search and proof construction in Horn clause logic. *CoRR*, abs/1511.07865, 2015. URL http://arxiv.org/abs/1511.07865. 4, 187

Ekaterina Komendantskaya and Yue Li. Productive corecursion in logic programming. *Theory and Practice of Logic Programming*, 17(5-6):906–923, 2017. doi: 10.1017/S147106841700028X. URL http://arxiv.org/abs/1707.01541. 69

Ekaterina Komendantskaya and Yue Li. Towards coinductive theory exploration in Horn clause logic: Position paper. In Temesghen Kahsai and German Vidal, editors, Proceedings 5th Workshop on *Horn Clauses for Verification and Synthesis,* Oxford, UK, 13th July 2018, volume 278 of *Electronic Proceedings in Theoretical Computer Science*, pages 27–33. Open Publishing Association, 2018. doi: 10.4204/EPTCS.278.5. 4, 186

Ekaterina Komendantskaya, John Power, and Martin Schmidt. Coalgebraic logic programming: from semantics to implementation. *Journal of Logic and Computation*, 26(2):745–783, 2016. doi: 10.1093/logcom/exu026. URL https://arxiv.org/abs/1312.6568. 4, 186

Ekaterina Komendantskaya, Dmitry Rozplokhas, and Henning Basold. The new normal: We cannot eliminate cuts in coinductive calculi, but we can explore them. *Theory and Practice of Logic Programming*, 20(6):990–1005, 2020. doi: 10.1017/S1471068420000423. 187

Robert Kowalski. Predicate logic as programming language. In *IFIP congress*, volume 74, pages 569–574, Jan 1974. 15, 118

Giorgio Levi, Catuscia Palamidessi, Pier Giorgio Bosco, Elio Giovannetti, and Corrado Moiso. A complete semantic characterization of K-leaf: A logic language with partial functions. In *Proceedings of the 1987 Symposium on Logic Programming, San Francisco, California, USA, August 31 - September 4, 1987*, pages 318–327. IEEE-CS, 1987. 71

John W. Lloyd. *Foundations of Logic Programming*. Artificial intelligence. Springer, 1987. ISBN 9783540181996. URL https://books.google.co.uk/books?id=rcI1nQEACAAJ. 40

Michael J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *LICS*, pages 348–357, 1988. ISBN 0818608536. doi: 10.1109/LICS.1988.5132. URL https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNyLiuB4/pdf. a, 14, 15, 16, 111, 112, 121, 143, 146, 184

Bertrand Meyer. Applying "design by contract". *Computer*, 25(10):40–51, October 1992. ISSN 0018-9162. doi: 10.1109/2.161279. URL https://doi.org/10.1109/2.161279. 149

Dale Miller and Gopalan Nadathur. Higher-order logic programming. In Ehud Shapiro, editor, *Third International Conference on Logic Programming*, pages 448–462, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg. 189

Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51(1): 125–157, 1991. ISSN 0168-0072. doi: 10.1016/0168-0072(91)90068-W. URL https://www.sciencedirect.com/science/article/pii/016800729190068W. 190

Paulo Moura. A portable and efficient implementation of coinductive logic programming. In Kostis Sagonas, editor, *Practical Aspects of Declarative Languages*, pages 77–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-45284-0. 4

Gopalan Nadathur and Dale Miller. Higher-order logic programming. *Handbook of logic in artificial intelligence and logic programming*, 5:499–590, 1998. 190

Aleksandar Nanevski, Greg Morrisett, and Lars Birkedal. Polymorphism and separation in Hoare type theory. In *Proceedings of the Eleventh ACM SIGPLAN International Conference on Functional Programming*, ICFP '06, page 62–73, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933093. doi: 10.1145/1159803.1159812. URL https://doi.org/10.1145/1159803.1159812. 12

Aleksandar Nanevski, Greg Morrisett, and Lars Birkedal. Hoare type theory, polymorphism and separation. *J. Funct. Program.*, 18:865–911, 09 2008. doi: 10.1017/S0956796808006953. 12

C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 81–90, 2006. doi: 10.1109/LICS.2006.38. URL https://www.cs.ox.ac.uk/people/luke.ong/personal/publications/lics06.pdf. 5, 6, 188

C.-H. Luke Ong. Higher-order model checking: An overview. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 1–15, 2015. doi: 10.1109/LICS.2015.9. URL http://www.cs.ox.ac.uk/people/luke.ong/personal/publications/LICS15.pdf. 110, 187

C.-H. Luke Ong and Dominik Wagner. Hochc: A refutationally complete and semantically invariant system of higher-order logic modulo theories. In *2019 34th Annual*

ACM/IEEE Symposium on Logic in Computer Science (LICS), pages 1–14, 2019. doi: 10.1109/LICS.2019.8785784. URL https://arxiv.org/abs/1902.10396. 15, 29, 36, 37, 117, 121

Susan S. Owicki. *Axiomatic Proof Techniques for Parallel Programs*. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1975. ISBN 0-8240-4413-4. 11

Christine Paulin-Mohring. Inductive definitions in the system Coq rules and properties. In Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications*, pages 328–345, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. ISBN 978-3-540-47586-6. 12

Long Pham, Steven J. Ramsay, and C.-H. Luke Ong. Defunctionalization of higher-order constrained Horn clauses. *CoRR*, abs/1810.03598, 2018. URL http://arxiv.org/abs/1810.03598. 10, 117, 119

Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(77)90044-5. URL http://www.sciencedirect.com/science/article/pii/0304397577900445. 149, 159

Gordon D. Plotkin. Domains (pisa notes), 1983. 17, 19, 22, 24

Steven J. Ramsay. *Intersection types and higer-order model checking*. PhD thesis, University of Oxford, UK, 2014. URL http://ora.ox.ac.uk/objects/uuid:46b7bc70-3dfe-476e-92e7-245b7629ae4e. 1

Yann Régis-Gianas and François Pottier. A Hoare logic for call-by-value functional programs. In Philippe Audebaud and Christine Paulin-Mohring, editors, *Mathematics of Program Construction, 9th International Conference, MPC 2008, Marseille, France, July 15-18, 2008. Proceedings*, volume 5133 of *Lecture Notes in Computer Science*, pages 305–335. Springer, 2008. doi: 10.1007/978-3-540-70594-9_17. URL https://doi.org/10.1007/978-3-540-70594-9_17. 11, 12, 16, 191, 192

Bernhard Reus and Thomas Streicher. About Hoare logics for higher-order store. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1337–1348. Springer, 2005. doi:

10.1007/11523468_108. URL https://doi.org/10.1007/11523468_108. 191, 192

Bernhard Reus and Thomas Streicher. Relative completeness for logics of functional programs. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 12, pages 470–480, 01 2011. doi: 10.4230/LIPIcs.CSL.2011.470. 191, 192

John C. Reynolds. Idealized algol and its specification logic. In Peter W. O'Hearn and Robert D. Tennent, editors, *Algol-like Languages*, pages 125–156. Birkhäuser Boston, Boston, MA, 1997. ISBN 978-1-4612-4118-8. doi: 10.1007/978-1-4612-4118-8_7. URL https://doi.org/10.1007/978-1-4612-4118-8_7. 192

Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. *Information and Computation*, 239:340 – 355, 2014. ISSN 0890-5401. doi: https://doi.org/10.1016/j.ic.2014.07.012. URL https://hal.inria.fr/inria-00589407/document. 187

Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, 1998. doi: 10.1017/S0960129598002527. 4

Ryosuke Sato, Naoki Iwayama, and Naoki Kobayashi. Combining higher-order model checking with refinement type inference. In *Proceedings of the 2019 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, PEPM 2019, page 47–53, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362269. doi: 10.1145/3294032.3294081. URL https://doi.org/10.1145/3294032.3294081. 7

Dana S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1):411 – 440, 1993. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(93)90095-B. URL http://www.sciencedirect.com/science/article/pii/030439759390095B. 149

Géraud Sénizergues. L(A)=L(B)? Decidability results from complete formal systems. *Theor. Comput. Sci.*, 251(1-2):1–166, 2001. doi: 10.1016/S0304-3975(00)00285-1. 110

Géraud Sénizergues. L(A)=L(B)? A simplified decidability proof. *Theor. Comput. Sci.*, 281(1-2):555–608, 2002. doi: 10.1016/S0304-3975(02)00027-0. 110

Kurt Sieber. Relating full abstraction results for different programming languages. In Kesav V. Nori and C. E. Veni Madhavan, editors, *Foundations of Software Technology and Theoretical Computer Science*, pages 373–387, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. 149, 153, 159

Luke Simon, Ajay Bansal, Ajay Mallya, and Gopal Gupta. Co-logic programming: Extending logic programming with coinduction. In *Autom. Lang. Program.*, pages 472–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-73420-8_42. 3, 4, 69, 186

Rick Statman. On the Lambda-Y calculus. *Annals of Pure and Applied Logic*, 130 (1-3 SPEC. ISS.):325–337, 2004. ISSN 01680072. doi: 10.1016/j.apal.2004.04.004. URL https://core.ac.uk/download/pdf/82358399.pdf. 110, 187

Colin Stirling. Decidability of DPDA equivalence. *Theor. Comput. Sci.*, 255(1-2): 1–31, 2001. doi: 10.1016/S0304-3975(00)00389-3. URL http://homepages.inf.ed.ac.uk/cps/dpda.pdf. 110

Colin Stirling. Decidability of higher-order matching. *Log. Methods Comput. Sci.*, 5 (3), 2009. URL http://arxiv.org/abs/0907.3804. 123

Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285 – 309, 1955. doi: pjm/1103044538. URL https://doi.org/. 19

Robert D. Tennent. *Semantics of programming languages*. Prentice Hall International Series in Computer Science. Prentice Hall, 1991. ISBN 978-0-13-805599-8. 21

Hiroshi Unno and Naoki Kobayashi. Dependent type inference with interpolants. In *Proceedings of the 11th ACM SIGPLAN conference on Principles and Practice of Declarative Programming*, pages 277–288, 2009. 183, 185

Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, and Simon L. Peyton Jones. Refinement types for Haskell. In Johan Jeuring and Manuel M. T. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, pages 269–282. ACM, 2014. doi: 10.1145/2628136.2628161. URL https://doi.org/10.1145/2628136.2628161. 5

Mahesh Viswanathan and Ramesh Viswanathan. A higher order modal fixed point logic. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR 2004 - Con-*

currency Theory, pages 512–528, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-28644-8. doi: 10.1007/978-3-540-28644-8_33. URL http://vmahesh.cs.illinois.edu/papers/concur04.pdf. 7, 189

William W. Wadge. Higher-order Horn logic programming. In *Logic Programming, Proceedings of the 1991 International Symposium, San Diego, California, USA, Oct. 28 - Nov 1, 1991*, pages 289–303, 1991. 34

Dominik Wagner. private communication, 2019. 14, 65, 188

Igor Walukiewicz. Automata theory and higher-order model-checking. *ACM SIGLOG News*, 3(4):13–31, December 2016. doi: 10.1145/3026744.3026745. URL https://www.labri.fr/perso/igw/Papers/igw-siglog16.pdf. 110

Keiichi Watanabe, Takeshi Tsukada, Hiroki Oshikawa, and Naoki Kobayashi. Reduction from branching-time property verification of higher-order programs to HFL validity checking. In *Proceedings of the 2019 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, PEPM 2019, page 22–34, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362269. doi: 10.1145/3294032.3294077. URL https://doi.org/10.1145/3294032.3294077. 7

Nobuko Yoshida, Kohei Honda, and Martin Berger. Logical reasoning for higher-order functions with local state. *CoRR*, abs/0806.2448, 2008. URL http://arxiv.org/abs/0806.2448. 12

Fabian Zaiser and C.-H. Luke Ong. The extended theory of trees and algebraic (co)datatypes. *Electronic Proceedings in Theoretical Computer Science*, 320: 167–196, Aug 2020. ISSN 2075-2180. doi: 10.4204/eptcs.320.14. 112

Index of definitions

Index of notations