Denotational World-indexed Logical Relations and Friends

PhD Dissertation, Jacob Thamsborg, IT University of Copenhagen

May 9, 2010

Abstract

As part of the long-standing drive for mathematical machinery to reason about computer programs, we use the technique of denotational logical relations to prove contextual equivalence of stateful programs. We propose the notion of approximate locations to solve the non-trivial problem of existence and solve the fundamental type-worlds circularity by metric-space theory. This approach scales to state-of-the-art step-indexed techniques and permits unrestricted relational reasoning by the use of so-called Bohr relations.

Along the way, we develop auxiliary theory; most notably a generalized version of a classical fixed-point theorem for functors on certain metric spaces by America and Rutten. Also we investigate the use of recursively defined metric worlds in an operational setting and arrive at constructions akin to step-indexed models.

On a different, though related, note, we explore a relational reading of separation logic with assertion variables. In particular, we give criteria for when standard, unary separation logic proofs lift to the binary setting. Phrased differently, given a module-dependent client and a standard separation logic proof of its correctness, we ponder the default question of representation-independence: is the client able to – or unable to – observe implementation specific details about its module.

Acknowledgements. I often, and only half-way joking, state that I chose my supervisor Lars Birkedal rather than my thesis topic. Looking back, I could have made no better choice.

Thanks goes also to Kristian Støvring and Hongseok Yang for collaboration and encouragement. To the PLS group at the IT University of Copenhagen for being a fun place to be. And to my wife for sharing the ups and downs.

Contents

Chapter 1

Introduction

1.1 Cannons and Contextual Equivalence

There was a time, when providing incentive meant more than taxing cigarettes. Cast-iron cannons, if cast badly, will burst into fragments when fired, much like a bomb; hence the makers of cannons were required to fire their products once while sitting on top of it. This tested the functionality and, in addition, encouraged constant care on part of the makers.

Quality assurance remains an issue, these days also in the development of computer programs. Malfunctioning software sends space-bound rockets falling from the skies, blows up gas pipelines and cause no small annoyance to those working close to deadlines. Such errors prevail, even in the face of extensive testing and despite significant (and mostly non-physical) incentive to get things right. In this dissertation we propose a further development of a formal method of proving computer programs indiscernible; it is a step in the classic quest of eliminating errors in computer programs by harnessing the power of mathematics

The *objects* of our study are computer programs. The long-term goal is to reason about real-life programs, so to speak, but we restrict attention to computer programs written in a somewhat academic programming language: *the polymorphic typed lambda calculus with recursive types and general references*. It is, despite the lengthy name, a standard language of theoretical computer science, well suited for formal reasoning. At the same time, it is no toy-language; it is easily as expressive as industry-standard programming languages, though it may fail to match their practical ease of use.

The *purpose* is (mainly) to give tools for proving computer programs indiscernible. The gold standard is *contextual equivalence*, a well-studied concept of theoretical computer science with a wide range of applications. Two computer programs are contextually equivalent if, roughly speaking, replacing one by the other in some larger program – the context – does not change the behavior of that larger program. If we break open the two programs and compare them, instruction by instruction, then they may be quite different, but there should be no way of telling that from the outside, i.e., from the observations that the context can make.

The *framework* is mathematics. We need to specify, in all details, the behavior of computer programs before we can hope to reason precisely about them. We do this by way of *denotational semantics*, a well-known method of giving meaning to computer programs as mathematical entities. It is pleasantly abstract, but is somewhat removed from the workings of real computers; hence we must take care not to loose touch with reality.

Our *method* of choice is *logical relations*. The programming language is typed,

meaning that it distinguishes between different types of data and their uses. We tend to eat oranges and throw frisbees instead of the other way round; in much the same way, a computer program may increase a number by one or spell-check a word, but typing rules will prevent it from doing the opposite. Hence our use of logical relations: they are a standard technique for reasoning about well-behaved, i.e., typed programs.

1.2 Reader's Guide

This dissertation is a collection of 6 papers / manuscripts and two minor notes. With the exception of the latter, each part comes with contextual information, i.e., motivation, textual explanation and discussion of related work, in addition to the technical results. This information will not be repeated in full here; we give only terse summaries and try to stress the interrelation between the parts. While all explanations are textual, the text assumes familiarity with semantic concepts such as logical relations, relational parametricity, etc. The aim is to provide a sort of (technical) reader's guide.

For each work we give the publication status, i.e., we denote whether it has been submitted, published, etc.

1.2.1 Relational Parametricity for References and Recursive Types [14]

We build a logical relation for a language with recursive types, universal types and general references. Existence of the logical relation is the major contribution; it is proved along the lines of Pitts [23]. As a means to that end, we devise the concept of *approximated locations*[1], a necessity for the proof to go through. A location, say l, is modelled, conceptually, not just as the integer l but rather as an infinite list $l, l, l, \ldots$; this crucially has approximations in the form of finite lists of l's.

The logical relation is indexed over worlds to keep track of types of values in the store. Worlds, in this setting, are syntactic, i.e., they map locations to syntactic types. The latter may contain free type variables and this entangles the choice of space of semantic types. Our answer, the somewhat awkwardly named *semantic closed types*, strikes a balance between pure syntax and pure semantics – more expressive than the former but without the definitional circularities of the latter. Their access to the world is indirect, but, in combination with a peculiar notion of future worlds, they are quite expressive.

Looking back, the notion of syntactic worlds is unsatisfactory to the semanticist as we get no proper understanding of types. Also, the semantic closed types seem unwieldy; the authors have not explored them beyond the scope of this paper. On the other hand, the paper did give the first denotational and relationally parametric logical relation for a language with the aforementioned types. And it provided ideas and intuition that pervade the following development, most notably the reuse of approximated locations.

This is joint work with Lars Birkedal and Kristian Støvring. The paper was presented in Savannah in 2009 at the TLDI workshop affiliated with the POPL conference.

[1] These are also, confusingly, referred to as *semantic locations* elsewhere.

1.2.2 Realizability Semantics of Parametric Polymorphism, General References, and Recursive Types [8]

Again we build a logical relation for a language with recursive and universal types and general references. But as opposed to the previous work, the worlds are now semantic, they associate semantic types with locations. This means, that we face the *type-world circularity* head on: the semantic definition of either presupposes the definition of the other. A proper solution to this is the main contribution: we phrase the circularity as a functor fixed-point equation on a certain category of metric spaces; the existence of a fixed point – our semantic types – is guaranteed by a classic result by America and Rutten [4].

The metrics pervade the development in general, semantic type are *non-expansive* maps from worlds to certain relations and the interpretation of recursive types is by Banach's fixed-point theorem. The approximated locations from the previous work are reemployed; they are necessary to prove non-expansiveness of the interpretation of reference types.

This is joint work with Lars Birkedal and Kristian Støvring. The paper has been submitted to Mathematical Structures in Computer Science; it has been reviewed to provisional acceptance, i.e., final acceptance is subject to certain revisions being carried out. These revisions are incorporated in the version included here. An extended abstract [13] of this paper was presented in York in 2009 at the FOSSACS conference.

1.2.3 The Category-Theoretic Solution of Recursive Metric-Space Equations [7]

We digress from the development of logical relations to give a generalization of the metric fixed-point theorem of America and Rutten [4] along the lines of work by Smyth and Plokin [26].

In solving the type-world circularity as described in the previous subsection, we observed that solving for types and then defining worlds afterwards worked out, but the other way round did not. Worlds come with an ordering corresponding to further allocation, and types need to be monotone with respect to that. But the theorem by America and Rutten supplies fixed-points only in a category of plain metric spaces and hence cannot be used to solve for worlds.

While merely a curiosity in the original construction, this is an issue proper in work to scale the model to the notion of worlds found in work by Ahmed, Dreyer and Rossberg [1]. In their work, which relies on step-indexing to break circularities as described in the next section, the type-world circularity is of a more complex nature; indeed, it is a world-world circularity rather. We were unable to solve it using the existing theorem by America and Rutten for the reason sketched above.

Such was, in summa, the motivation behind this work: to give a version of the theorem for a category of certain ordered metric spaces. In the end, we went further, as the existence result given in the paper suffices for, but is not limited to, that purpose; the notions of *M-categories* and *locally contractive functors* are quite general. Also the paper has related results on compact ultrametric spaces and the relation between O-categories and M-categories.

This is joint work with Lars Birkedal and Kristian Støvring. The paper has been accepted for publication in Theoretical Computer Science subject to the rephrasing of a minor comment. An extended abstract of this paper [9] was presented in Coimbra in 2009 at the FICS workshop.

1.2.4 A Relational Realizability Model for Higher-Order Stateful ADTs [11]

We demonstrate that the idea of worlds as the fixed-point of a functor on certain metric spaces scales to match the state-of-the-art notions of worlds found in work by Ahmed, Dreyer and Rossberg [1]. The generalized fixed point theorem proved by the authors [7] is applied and we need a to make various technical adjustments to the existing model [8]; the use of approximate locations is still crucial.

A novelty is the use of *Bohr relations*. In the previous models, we required *uniformity* of our relations; this implied that we could not, say, relate a pair of integers to their sum, roughly because they behave differently under certain projections. The insight here is, that the cause of this restriction is the pursuit of a logical relation that approximates contextual equivalence – and that aiming for contextual approximation instead lifts the restriction. The Bohr relations are the technical counterpart to this and we are free to make use of (the closure of) any relation when working in the model.

This is joint work with Lars Birkedal and Kristian Støvring. The paper has been submitted to the Nordic Workshop on Programming Theory 2009 special issue of Journal of Logic and Algebraic Programming.

1.2.5 Kripke Models over Recursively Defined Metric Worlds: Steps and Domains [10]

We re-apply the above recipe for constructing worlds, but in an operational rather than denotational setting. Concretely, we give a unary world-indexed syntactic interpretation of the types of polymorphic lambda calculus with general references. In the denotational setting, the metric on semantic values was derived from the structure supplied with solutions to recursive domain equations; this, obviously, is not possible here. Instead we decorate syntactic values with natural numbers that, inspired by step-indexed models, provide formal approximations to each element. We build metrics on top, and are, thus, able to phrase worlds as the fixed-point of a functor, very similar to our denotational setup [8]. The resulting model is related and akin to, e.g., the step-indexed model of Ahmed [3], but is, we argue, simpler because we have a proper notion of worlds instead of the stratified solution of the cited thesis.

Also we show that our approach specializes to recent work on *indirection theory* by Hobor, Dockins and Appel; indirection theory is, roughly, an abstraction and generalization of the stratified construction of worlds as proposed by Ahmed [3] and refined, e.g., by Ahmed, Dreyer and Rossberg [1]. Finally we give an operational analogue of an existing denotational model of separation logic for nested Hoare triples; for that application it is important to have a proper space of worlds. In conclusion, we provide evidence of the wide applicability of our metric techniques, also outside the realm of denotational semantics.

This is joint work with Lars Birkedal and Kristian Støvring. The manuscript is unpublished. The contents are, however, included in a paper submitted to the ICFP 2010 conference. The submission also applies the techniques to Chargueraud and Pottier's capability calculus [16] and has Reus, Schwinghammer and Yang as additional coauthors.

1.2.6 Two for the Price of One: Lifting Separation Logic Assertions [29]

The last work strays from the main development: We strive to give a relational reading of separation logic proofs involving assertion variables. The idea is to prove

a client program correct by means of standard separation logic and then give a relational reading of that proof to obtain – for free, so to speak – representation independence properties. It is a natural extension of work by Birkedal and Yang [15]; their assertions came without assertion variables.

The Achilless' heel is the use of the rule of consequence of separation logic. The hypotheses of this rule include implications between assertions; the main contribution of the manuscript are deliberations on whether, semantically speaking, validity in the unary reading of an implication between assertions *lifts* to validity in the binary reading. Concretely, we investigate assertions with assertion variables of a certain form and give, based on the layout of the assertion variables, a complete characterization of when semantic validity lifts from the unary to the binary reading.

This is joint work with Lars Birkedal and Hongseok Yang. This manuscript is, as of now, unpublished.

1.2.7 A Tale of Two Recursive Predicates (note) [31]

All of our logical relations rely on the use of approximate locations, in this note we seek to justify this. By contradiction, we show that certain world-indexed logical relations do not exist; these are cooked-down versions of our first logical relation [14] with the all-important distinction that locations are modelled as flat integers. This is not to say that approximated locations are the only solution to the problem, neither that they are necessarily the best, but it does demonstrate that there *is* a problem. In the spirit of approximated locations, this note also gives a possible work-around to an open problem posed by Benton, Kennedy, Beringer and Hofmann [6, Section 6.2].

This is a note, not a stand-alone work. It should be read in the context of our logical relation constructions, most notable the two first [14, 8], which have fairly direct notions of worlds.

1.2.8 From M-categories to O-categories (note) [30]

In our generalization [7] of the fixed-point theorem of America and Rutten [4] we gave a construction of M-categories from O-categories. It is such, that a locally continuous functor on the O-category gives rise to a locally contractive functor on the M-category; furthermore, we can move the unique fixed-point of the latter back across the divide to give a minimal invariant [23] of the former.

In this note, we go the other way. For starters, we construct complete partial orders from complete ultrametric spaces, and use this to prove Banach's fixed-point theorem on the former by Kleene's fixed-point theorem on the latter. We then scale to categories, proceeding exactly as sketched above, only going the other way.

This note is even less self-contained than the previous; it should be read only in connection with our work on the generalized fixed-point theorem [7].

1.3 Some Related Techniques

The bulk of this dissertation pursues contextual equivalence by means of (world-indexed) denotational logical relations. Other roads lead to Rome, in this section we sketch alternative techniques. This is no comprehensive survey but merely a pointer to a few crucial results closely related to the dissertation.

1.3.1 Syntactic Logical Relations

The origins of logical relations are denotational but they can also be built directly on the language terms / operational semantics. In an expository presentation [22,

Chapter 7] based on earlier work [25, 24], Pitts gives an operational logical relation for a functional language with universal types and term-level recursion. It is operational in the sense that the relation associated to a (closed) type really is a relation between terms of that type. To prove that relatedness is preserved under the typing rule rule for fixed-points, Pitts faces the standard recursive-term problem: the hypothesis only yields that finite unfoldings are related. To solve this, he effectively works with syntactic $\top\top$-closed relations, much in the same way we build relations on computations from relations on values.

An advantage of working in an operational setting is that Pitts obtains a logical relation that is not only sound but also complete with respect to contextual equivalence. Completeness is proved by squeezing contextual equivalence into ciu-equivalence; the latter is similar to contextual equivalence but limits the choice of contexts. Ciu-equivalence is, then again, squeezed into the logical relation. The proof relies on the observation that, for closed types, the logical relation respects ciu-equivalence; this is another benefit of working with $\top\top$-closed relations. While satisfying, practical uses of completeness are not immediate. One may 'import' into the logical relations contextual equivalences from elsewhere; apart from that, however, it does not appear readily applicable. Confer also the discussion in a recent submission by Dreyer, Neis and Birkedal [18, Section 8].

The programme to build syntactic logical relations by refitting denotational techniques was continued by Birkedal and Harper [12] and later Crary and Harper [17], the former include a single recursive type and the latter full recursive types and universal as well as existential quantification. As is the standard problem for logical relations over recursive types, definition by induction on types will not do, so one needs to argue the existence of the logical relation otherwise. The solution adopted in both papers is, stated tersely, to give a syntactic analogue of the minimal invariance property of solutions to domain equations [23].

The pioneering work of Pitts and Stark on world-indexed logical relations [21, Pages 227-274] also belong in this subsection; it is not unreasonable to say that main parts of this dissertation are (denotational) extensions of their work to richer languages. They build a syntactic logical relation for a language with term-level recursion and ground store, i.e., references to integers only; it is sound and complete with respect to contextual equivalence by arguments along the lines discussed above. More to the point, though, is that the logical relation is world-indexed, although this is not their choice of nomenclature. Worlds, as in this dissertation, are abstractions over stores; a pair of stores belong to a world if they meet the requirements of the world. In this early work, this is very concrete, worlds are plain relations on stores. Their use is twofold, again as in this dissertation: To interpret the reference type (and functions types) some (though not absolute) knowledge of the store is required; what are, e.g., the allocated locations and which locations store identical integers. This knowledge is obtained by parameterizing the logical relation over worlds. Secondly, worlds may capture reasoning that relies on local use of store, e.g., some relation between stored values that cannot be disrupted by the context because the locations are not exported. Pitts and Stark give a thorough presentation of the technical development and prove non-trivial equivalences such as contextual equivalence of certain functions and their memoised versions.

1.3.2 Step-indexed Models

Over the last decade, operational, *step-indexed* models have emerged. Coined by Appel and McAllester [5], the basic idea goes something like this. While it is nice to know that the value v has type τ, we sometimes can make do with less: it may suffice to know that v will behave as if it has type τ for some n execution steps. In other words, the assumption that v has type τ may be wrong, but it takes at least $n+1$

steps to discover that. We say that the v has type τ to approximation n; the pair $(\texttt{true}, (4, 3))$ of a boolean and a pair of integers, e.g., has type $\texttt{bool} \times (\texttt{bool} \times \texttt{bool})$ to approximation 2. This is because it will take two steps (two projections) to extract, e.g., 4 with the alleged type `bool` and then one more step to break down on account of this *faux pas*.

The reason for stratifying types is to break circularities. One such is the recursive-types circularities mentioned in the subsection above: We like to interpret types by induction on their structure, but in the presence of recursive types this breaks down: we would, e.g., say that $\texttt{fold}\, v$ has type $\mu\alpha.\,\tau$ if v has type $\tau[\mu\alpha.\,\tau/\alpha]$, but the latter type is no subtype of the former. If, however, we only require that $\texttt{fold}\, v$ has type $\mu\alpha.\,\tau$ to approximation $n+1$ then it will suffice to verify that v has type $\tau[\mu\alpha.\,\tau/\alpha]$ to approximation n because it takes one execution step to unfold the recursive value. Hence, if we do nested induction with the outer induction on the steps and the inner induction on the types, we break the circularity. Not only the recursive-types circularity but also the type-world circularity solved in this dissertation [8] by metric means can be broken. When interpreting a reference type, say $\texttt{ref}\,\tau$, one roughly wants to return the allocated locations that hold values of type τ; hence we index our interpretations by worlds, which, in this context, are lists of allocated locations with associated types. Unfortunately, this introduces a definitional circularity, if we store semantic types in our worlds: the space of semantic types depends on the space of worlds and vice versa. If, however, we only require that a location l has type $\texttt{ref}\,\tau$ to approximation $n+1$ then it suffices to verify that τ and the type of values stored at location l agree to approximation n because, e.g., a memory lookup takes one execution step. This again means, that to decide whether a value lies in some type in some world to approximation $n+1$, we need only inspect the types of the world to approximation n. Hence we may stratify our types / worlds and the circularity is broken.

In her thesis [3], Ahmed explores this idea. She considers an untyped polymorphic lambda calculus augmented with mutable references and existentials and gives a unary, step-indexed safe interpretation of types including recursive and impredicative quantified types as well as general references. It uses none of the aforementioned sophisticated operational techniques: syntactic minimal invariance is superfluous by the breaking of the recursive-types circularity and since, in essence, she considers only finite unfoldings of recursive terms there is no need for syntactic admissibility or $\top\top$-closure. Neither does she produce a proper solution to the type-world circularity as in the work in this dissertation, rather it is simply broken by the step-indexing as described above.

Ahmed proceeds [2] to build a step-indexed syntactic logical relation for a language with recursive and impredicative quantified types, but no state. The relation is proved sound and complete with respect to contextual equivalence, the latter by the 'standard' method of embedding ciu-approximation into the logical relation; this, again, require her to consider only ciu-approximation respecting relations when, e.g., arguing by relational parametricity.

In recent work, Ahmed, Dreyer and Rossberg [1] have extended Ahmed's thesis to give a logical relation for recursive types, general references and impredicative quantification. As compared to the thesis, the major novelty is the binary interpretation that is sound with respect to contextual equivalence and, more to the point, a quite expressive notion of worlds. Apart from keeping track of the types of allocated references, the worlds can capture complex and dynamic invariants of the heap that hold for reasons of local use of store. This idea was present already in the work by Pitts and Stark [21, Pages 227-274] as mentioned, but is extended considerably. In a recent submission by Dreyer, Neis and Birkedal [18], the dynamic behavior of such invariants is explored further, to the point where complex call-back patterns can be captured elegantly using a finite-state machine.

Much of this dissertation has to do with the type-world circularity in a denotational setting. Approximative locations are introduced [14] to permit a denotational logical relation indexed by syntactic worlds; they are reused [8] to give an actual solution to the type-world circularity by metric techniques, this grants a logical relation indexed over semantic worlds, i.e., worlds that associate locations with semantic, rather than syntactic, types. Finally we extend [11] the notion of worlds to meet the standards of the work by Ahmed, Dreyer and Rossberg [1], demonstrating that our techniques scale to state-of-the-art in terms of world-expressivity.

What, then, does our mathematical machinery buy us compared to the somewhat low-tech approach of step-indexing? There is room for discussion here. We know of no contextual equivalence that may be proved by our methods but not by step-indexing. On the other hand, our approach yields more abstract models. Indeed, it is not unreasonable to claim that direct proofs in our model are at a level of abstraction comparable to recent logics [19, 20] built on top of the step-indexed models; these feature modal logic to do away with the steps that otherwise clutter calculations in the model. Also we expect that our models will do better in terms of categorical / logical properties but this remains to be investigated further.

Bisimulation

In the search for contextual equivalence, the use of bisimulations is an alternative to logical relations. The two are dissimilar: Building a logical relation is a game, so to speak, of giving a binary interpretation of types that is rich enough to relate a great many things, whilst restrictive enough to be maintained across typed constructions of terms. This will, roughly speaking, imply contextual equivalence, since the context cannot 'escape' the relation – provided that the relation exists in the first place.

On the other hand, to prove a contextual equivalence by use of bisimulations, we try to outwit the context. We build a set, the bisimulation, of pairs of possibly different values of equal type; this set should be closed under any typed decomposition that a context might apply: if, say, it contains $((v_1, v_2), (w_1, w_2))$, both with type $\sigma \times \tau$, then it should contain (v_1, w_1) of type σ too. Intuitively, the set should contain all differences that the context may extract from the two programs under scrutiny. If no pairs in the set give rise to expressions that fail to co-terminate, then any two values that is in the set are contextually equivalent since there is no way for the context to produce uneven termination behavior.

Bisimulations are employed, e.g., in work by Sumii and Pierce [28], to give a complete proof method for a typed λ-calculus with full universal, existential and recursive types. This is extended to cover general references in recent work by Sumii [27]. To do so, one also needs to keep track of the states in which values are related to facilitate, e.g., lookup.

As compared to logical relations, bisimulations are closer to a direct proof of contextual equivalence. They employ only elementary math as opposed to denotational logical relations or even syntactic ones before the advent of step-indexing. Many concrete examples can be proved, some with nice and short proofs; programs that are alike in terms of syntax tend to do well. Sumii [27] gives a wider range of examples than Ahmed, Dreyer and Rossberg [1], confer also a recent submission by Dreyer, Neis and Birkedal [18]. On the other hand, bisimulations are less abstract, and do not provide an answer to questions such as, e.g., what all the contextually equivalent values at some type are. In other words, they do not give models of types.

Bibliography

[1] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. State-dependent representation independence. In Zhong Shao and Benjamin C. Pierce, editors, *POPL*, pages 340–353. ACM, 2009.

[2] Amal J. Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In Peter Sestoft, editor, *ESOP*, volume 3924 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2006.

[3] Amal Jamil Ahmed. *Semantics of Types for Mutable State*. PhD thesis, Princeton University, 2004.

[4] Pierre America and Jan J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.*, 39(3):343–375, 1989.

[5] Andrew W. Appel and David A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.

[6] Nick Benton, Andrew Kennedy, Lennart Beringer, and Martin Hofmann. Relational semantics for effect-based program transformations: higher-order store. In António Porto and Francisco Javier López-Fraguas, editors, *PPDP*, pages 301–312. ACM, 2009.

[7] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations. 2009.

[8] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. 2009.

[9] L. Birkedal, K. Støvring, and J. Thamsborg. Solutions of generalized recursive metric-space equations. In *Proceedings of FICS 2009*, September 2009.

[10] L. Birkedal, K. Støvring, and J. Thamsborg. Kripke models over recursively defined metric worlds: Steps and domains. 2010.

[11] L. Birkedal, K. Støvring, and J. Thamsborg. A relational realizability model for higher-order stateful ADTs. 2010.

[12] Lars Birkedal and Robert Harper. Relational interpretations of recursive types in an operational setting. *Inf. Comput.*, 155(1-2):3–63, 1999.

[13] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In Luca de Alfaro, editor, *FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 456–470. Springer, 2009.

[14] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. Relational parametricity for references and recursive types. In Andrew Kennedy and Amal Ahmed, editors, *TLDI*, pages 91–104. ACM, 2009.

[15] Lars Birkedal and Hongseok Yang. Relational parametricity and separation logic. In Helmut Seidl, editor, *FoSSaCS*, volume 4423 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2007.

[16] Arthur Charguéraud and François Pottier. Functional translation of a calculus of capabilities. In James Hook and Peter Thiemann, editors, *ICFP*, pages 213–224. ACM, 2008.

[17] Karl Crary and Robert Harper. Syntactic logical relations for polymorphic and recursive types. *Electr. Notes Theor. Comput. Sci.*, 172:259–299, 2007.

[18] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. 2010. Submitted for publication.

[19] Derek Dreyer, Amal Ahmed, and Lars Birkedal. Logical step-indexed logical relations. In *LICS*, pages 71–80. IEEE Computer Society, 2009.

[20] Derek Dreyer, Georg Neis, Andreas Rossberg, and Lars Birkedal. A relational modal logic for higher-order stateful adts. In Manuel V. Hermenegildo and Jens Palsberg, editors, *POPL*, pages 185–198. ACM, 2010.

[21] Andrew D. Gordon and Andrew M. Pitts, editors. *Higher order operational techniques in semantics*. Cambridge University Press, New York, NY, USA, 1998.

[22] Benjamin C. Pierce. *Advanced Topics in Types and Programming Languages*. MIT Press, Cambridge, MA, USA, 2002.

[23] Andrew M. Pitts. Relational properties of domains. *Inf. Comput.*, 127(2):66–90, 1996.

[24] Andrew M. Pitts. Existential types: Logical relations and operational equivalence. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 1998.

[25] Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in Computer Science*, 10(3):321–359, 2000.

[26] Michael B. Smyth and Gordon D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.*, 11(4):761–783, 1982.

[27] Eijiro Sumii. A complete characterization of observational equivalence in polymorphic *ambda*-calculus with general references. In Erich Grädel and Reinhard Kahle, editors, *CSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2009.

[28] Eijiro Sumii and Benjamin C. Pierce. A bisimulation for type abstraction and recursion. In Jens Palsberg and Martín Abadi, editors, *POPL*, pages 63–74. ACM, 2005.

[29] J. Thamsborg, L. Birkedal, and H. Yang. Two for the price of one: Lifting separation logic assertions. 2010.

[30] Jacob Thamsborg. From m-categories to o-categories. 2010.

[31] Jacob Thamsborg. A tale of two recursive predicates. 2010.

Chapter 2

Relational Parametricity for References and Recursive Types

Relational Parametricity for References and Recursive Types

Lars Birkedal Kristian Støvring Jacob Thamsborg
IT University of Copenhagen
{birkedal,kss,thamsborg}@itu.dk

Abstract

We present a possible world semantics for a call-by-value higher-order programming language with impredicative polymorphism, general references, and recursive types. The model is one of the first relationally parametric models of a programming language with all these features.

To model impredicative polymorphism we define the semantics of types via parameterized (world-indexed) logical relations over a universal domain. It is well-known that it is non-trivial to show the existence of logical relations in the presence of recursive types. Here the problems are exacerbated because of general references. We explain what the problems are and present our solution, which makes use of a novel approach to modeling references. We prove that the resulting semantics is adequate with respect to a standard operational semantics and include simple examples of reasoning about contextual equivalence via parametricity.

Categories and Subject Descriptors F.3.2 [*Logics and Meanings of Programs*]: Semantics of Programming Languages—Denotational semantics; F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs

General Terms Languages, Theory, Verification

Keywords Denotational Semantics, Possible World Semantics, Relational Parametricity, Impredicative Polymorphism, General References, Recursive Types

1. Introduction

Relational parametricity was proposed by Reynolds [34] to reason about polymorphic programs, in particular, to show equivalence of polymorphic programs and to show representation independence for abstract data types. In this paper we provide one of the first[1] relationally parametric models of a programming language with recursive types and general references. We prove that the resulting semantics is adequate with respect to a standard operational semantics, which means that we can use parametricity to show contextual equivalence of expressions in the language.

Our model is based on logical relations over an untyped model of the language. The logical relations are parameterized over possible worlds which are used to capture dynamic allocation of references, much as in [7, 11, 22, 31]. It is well-known that it is non-trivial to show the existence of logical relations in the presence of recursive types [28]. Here the problems are exacerbated because of general references. We explain the problems and present our solution, which makes use of a novel approach to modeling references.

In this paper we focus on the challenge of defining an adequate semantics, in particular on the challenge pertaining to the existence of the logical relations. The resulting model can be used to prove equivalence and parametricity results for programs using references in simple ways. In future work, we plan to extend the parameters of our logical relations to accommodate local relational reasoning about programs using local state. We plan to do this using the first author's earlier work on relational reasoning for languages with references and recursive types (but not polymorphism) [11].

1.1 Background

The theory of relational parametricity was originally proposed in the setting of the second-order lambda calculus. That setting is by now fairly well-understood, see, e.g., [9, 33]. But, of course, we would like to use relational parametricity for real programs with recursion and other effects. There has been a lot of research towards this goal — the efforts can be grouped roughly into two categories: *equational type theories with effects* and *programming languages with effects*.

Work in the former category was initiated by Plotkin [32], who suggested a second-order linear type theory with a polymorphic fixed-point combinator to combine polymorphism with recursion. That approach was further investigated in [10]. One of the remarkable features of this calculus is that it allows one to encode a wide range of data types, including recursive types, with the desired universal properties following from parametricity. Hasegawa studied the combination of polymorphism and another effect, namely control [15]. Recently, this line of work was extended by Møgelberg and Simpson [25], who proposed a general polymorphic type theory for effects, as captured by computational monads. The general framework has been specialized to control effects in [26].

Work in the latter category focuses on programming languages defined using an operational semantics, specifying evaluation order, etc., and was initiated by Wadler [37]. Relational parametricity is concerned with program equivalence which is here typically defined as *contextual equivalence*: two program expressions are equivalent if they have the same observable behaviour when placed in any program context C.

It is generally quite hard to show directly that two program expressions are contextually equivalent because of the universal quantification over all contexts. Thus there has been an extensive research effort to find reasoning methods that are easier to use for establishing contextual equivalence (see, e.g., [30] for a fairly recent overview), and the work on parametricity for programming languages with effects has been closely related to the research on reasoning methods for contextual equivalence. Relationally para-

[1] Independent work [3] by Ahmed, Dreyer and Rossberg came to our attention after writing this paper, cf. section 6; we know of no other models.

TLDI'09, January 24, 2009, Savannah, Georgia,USA.

metric models have been developed for languages with recursion and inductive / coinductive types, see, e.g., [8, 16, 17, 29] and, recently, also for languages with recursive types [2, 13, 23]. In addition, a number of bisimulation-based methods for proving contextual equivalence have recently been proposed; the methods most relevant for the work in this paper cover a pure language with recursive and existential types [36], untyped languages with general references and/or control operators [18, 19, 35], and a pure language with parametric polymorphism and recursive types [20, 21].

The two categories of work are, of course, related: the type theories serve as metalanguages and can be used to give semantics to programming languages. This has, e.g., been done by Møgelberg [24], who showed how to give a parametric model of the programming language FPC extended with polymorphism (i.e., a language with recursion, recursive types and polymorphism). Using a model of the type theory, adequacy wrt. the operational semantics of the programming language was proved, allowing Møgelberg to prove results about contextual equivalence using the reasoning principles of the type theory.

1.2 Overview of the technical development

In Section 2 we define the operational semantics of our programming language, which is a standard, direct-style, call-by-value higher-order language with impredicative polymorphism, recursive types, and references. The operational semantics is nondeterministic since dynamic allocation of references is modeled in the standard way via a nondeterministic choice of a new location.

In Section 3 we present an untyped denotational semantics of the language using a universal domain. In the denotational semantics we assume that the semantic set of locations is well-ordered (the set of locations is a copy of the natural numbers) and allocation is modeled by choosing the smallest free location. We use a novel form of semantic locations in the semantics; the motivation for these comes from the need to establish the existence of logical relations in the following section.

We prove that the denotational semantics is sound and adequate with respect to the operational semantics. This is done *almost* in the standard way by defining a logical formal approximation relation between the operational and denotational semantics. For the adequacy proof it suffices to give a logical relation for *closed* types and therefore, as we show, the existence of the logical relation can be proved using standard techniques [28]. The adverb 'almost' above refers to the following. For the existence proof one needs to show that the relations are suitably admissible and the standard proofs of that rely on determinacy of the operational semantics, see [28, Sec. 5, Page 81], but here we have a non-deterministic operational semantics. Intuitively, the denotational semantics should be adequate since the choice of new location should not matter for the final result of a program. In earlier domain-theoretic models of references for which adequacy have been proved [7, 11, 22], both language and denotational semantics have been defined in a monadic (continuation-passing) style; hence it was fairly easy to capture that the choice of location does not matter for the final result. Here we decide to stick to a direct-style language and operational semantics to make sure that our results do not depend on a monadic presentation and instead we define the logical relation in a continuation-passing style, which suffices for proving adequacy. In summary, the language is in direct style, but the proof of adequacy is in continuation-passing style.

The untyped semantics can only be used to establish simple forms of contextual equivalence. In Section 4 we therefore present a typed possible world semantics of the language by defining a family of parameterized logical relations over the universal domain for which we prove the fundamental theorem of logical relations. In combination with adequacy of the untyped semantics this proves adequacy for the typed semantics. To reason about parametricity we need to give a semantics not only of closed types but also of *open* types. This turns out to complicate the existence proof of the logical relation because, loosely speaking, we need to compare semantic types in the logical relation for reference types in order to check that the type for the location in the current world (a store typing) agrees with the type of the reference. We solve this problem by modeling references using a novel semantic notion of location which permits approximations to locations. The approximations are crucial for the existence proof of the logical relation. We explain what the problem is by highlighting what goes wrong if we omit such approximations.

In Section 5 we present a few examples of equivalences that can be proved using the resulting possible world semantics.

Finally, in Section 6 we conclude and briefly discuss directions for future work.

2. Types and Operational Semantics

Types, expressions and values are given in Figure 1. A *context of type variables* is a list of type variables with no repeats. For any such context Ξ and any type τ we write $\Xi \vdash \tau$ if the free type variables of τ are all in Ξ and we write $\mathbf{Type}_\Xi$ for all such types. A *world* is a partial map with finite domain from $\mathbb{N}$ to the set of types; we have a partial ordering on worlds defined by setting $\Delta \sqsubseteq \Delta'$ provided $\mathrm{dom}(\Delta) \subset \mathrm{dom}(\Delta')$ and $\Delta(l) = \Delta'(l)$ for all $l \in \mathrm{dom}(\Delta)$. A *context of term variables* is a partial map with finite domain from the set of term variables to the set of types. For any context of type variables Ξ and any world Δ we write $\Xi \vdash \Delta$ if $\Xi \vdash \Delta(l)$ for all $l \in \mathrm{dom}(\Delta)$ and we let $\mathbf{World}_\Xi$ be the set of worlds with this property. We define $\Xi \vdash \Gamma$ for a context of term variables Γ similarly.

We give selected typing rules in Figure 2, a complete presentation is found in Appendix A. The rules assign types to expressions under assumptions of contexts of type variables, worlds and contexts of term variables. It is not hard to see that the various side conditions ensure that $\Xi \mid \Delta \mid \Gamma \vdash e : \tau$ implies $\Xi \vdash \Delta$, $\Xi \vdash \Gamma$ as well as $\Xi \vdash \tau$. Also it is worth noticing that the language is explicitly typed to ensure type uniqueness: Given $\Xi \mid \Delta \mid \Gamma \vdash e : \tau_1$ and $\Xi \mid \Delta \mid \Gamma \vdash e : \tau_2$ we can conclude that $\tau_1 = \tau_2$ and that the derivations of the judgments coincide.

As usual we identify expressions up to α-equivalence. For convenience we write $\lambda^{\tau_0 \to \tau_1} x.\ e$ for $\mathtt{fix}^{\tau_0 \to \tau_1} f(x).e$ where f is some arbitrary variable not occurring free in e.

A *syntactic store* is a partial map with finite domain from $\mathbb{N}$ to the set of values. Using that definition, we define a standard big-step *operational semantics*; selected rules are given in Figure 3, see Appendix B for the unabridged story. It is a quaternary relation between syntactic stores and expressions on the one hand and syntactic stores and values on the other. Notice that the memory allocator is nondeterministic in the standard way: Any free location may be picked.

For a context Ξ, a world Δ, a context Γ and a syntactic store Π we write $\Xi \mid \Delta \mid \Gamma \vdash \Pi$ to denote that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Pi)$ and that for all $l \in \mathrm{dom}(\Delta)$ we have $\Xi \mid \Delta \mid \Gamma \vdash \Pi(l) : \Delta(l)$. We have the following standard proposition (see Chapter 13 of Pierce [27]):

Proposition 1 (Type Preservation)**.** *Assume $\Pi, e \Downarrow \Pi', v$. Suppose furthermore that we have $\emptyset \mid \Delta \mid \emptyset \vdash \Pi$ and $\emptyset \mid \Delta \mid \emptyset \vdash e : \tau$ for some world Δ and some type τ. Then there is $\Delta' \sqsupseteq \Delta$ such that $\emptyset \mid \Delta' \mid \emptyset \vdash \Pi'$ and $\emptyset \mid \Delta' \mid \emptyset \vdash v : \tau$.*

The proof is by induction on the structure of the derivation of the judgment. It relies on basic properties of the type system such as standard substitution lemmas for type and term variables as well as the fact that an expression of some type in one world has the same type in any larger world.

$$\tau ::= \alpha \mid \texttt{unit} \mid \texttt{int} \mid \tau\ \texttt{ref} \mid \tau \times \tau \mid \tau + \tau \mid \mu\alpha.\tau \mid \forall\alpha.\tau \mid \tau \to \tau$$

$$\begin{aligned} e ::= {} & x \mid () \mid n \mid l \mid \texttt{op}(e \pm e) \mid \texttt{ifzero}\ e\ \texttt{then}\ e\ \texttt{else}\ e \mid (e,e) \mid \texttt{fst}(e) \mid \texttt{snd}(e) \mid \texttt{inl}^{\tau_0+\tau_1}(e) \mid \\ & \texttt{inr}^{\tau_0+\tau_1}(e) \mid \texttt{case}\ e\ \texttt{of}\ \texttt{inl}(x).\, e\ \texttt{else}\ \texttt{inr}(x).\, e \mid \texttt{fold}^{\mu\alpha.\tau}(e) \mid \texttt{unfold}^{\mu\alpha.\tau}(e) \mid \\ & \Lambda\alpha.e \mid e[\tau] \mid \texttt{fix}^{\tau_0\to\tau_1} f(x).e \mid e(e) \mid \texttt{ref}(e) \mid !\, e \mid e := e \end{aligned}$$

$$v ::= () \mid n \mid l \mid (v,v) \mid \texttt{inl}^{\tau_0+\tau_1}(v) \mid \texttt{inr}^{\tau_0+\tau_1}(v) \mid \texttt{fold}^{\mu\alpha.\tau}(v) \mid \Lambda\alpha.e \mid \texttt{fix}^{\tau_0\to\tau_1} f(x).e$$

Figure 1. Types, expressions and values.

$$\frac{}{\Xi \mid \Delta \mid \Gamma \vdash l : \tau\ \texttt{ref}}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma, l \in \mathrm{dom}(\Delta), \Delta(l) = \tau)$$

$$\frac{\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \forall\alpha.\tau_0}{\Xi \mid \Delta \mid \Gamma \vdash e[\tau_1] : \tau_0[\tau_1/\alpha]}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Delta \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash e : \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash \texttt{fix}^{\tau_0\to\tau_1} f(x).e : \tau_0 \to \tau_1}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \texttt{ref}(e) : \tau\ \texttt{ref}} \qquad \frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \texttt{ref}}{\Xi \mid \Delta \mid \Gamma \vdash !\, e : \tau}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \texttt{ref} \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau}{\Xi \mid \Delta \mid \Gamma \vdash e_0 := e_1 : \texttt{unit}}$$

Figure 2. Select typing rules. The general form is $\Xi \mid \Delta \mid \Gamma \vdash e : \tau$ for a context of type variables Ξ, a world Δ, a context of term variables Γ, an expression e and a type τ.

$$\frac{}{\Pi, \Lambda\alpha.e \Downarrow \Pi, \Lambda\alpha.e}$$

$$\frac{}{\Pi, \texttt{fix}^{\tau_0\to\tau_1} f(x).e \Downarrow \Pi, \texttt{fix}^{\tau_0\to\tau_1} f(x).e}$$

$$\frac{\Pi, e \Downarrow \Pi', \Lambda\alpha.e' \qquad \Pi', e'[\tau/\alpha] \Downarrow \Pi'', v}{\Pi, e[\tau] \Downarrow \Pi'', v}$$

$$\frac{\Pi, e_0 \Downarrow \Pi', \texttt{fix}^{\tau_0\to\tau_1}\ f(x).e \quad \Pi', e_1 \Downarrow \Pi'', v \quad \Pi'', e[v/x, \texttt{fix}^{\tau_0\to\tau_1}\ f(x).e/f] \Downarrow \Pi''', v'}{\Pi, e_0(e_1) \Downarrow \Pi''', v'}$$

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \texttt{ref}(e) \Downarrow \Pi'[l \mapsto v], l}\ (l \notin \mathrm{dom}(\Pi'))$$

$$\frac{\Pi, e \Downarrow \Pi', l}{\Pi, !\, e \Downarrow \Pi', v}\ (l \in \mathrm{dom}(\Pi'), \Pi'(l) = v)$$

$$\frac{\Pi, e_0 \Downarrow \Pi', l \qquad \Pi', e_1 \Downarrow \Pi'', v}{\Pi, e_0 := e_1 \Downarrow \Pi''[l \mapsto v], ()}\ (l \in \mathrm{dom}(\Pi''))$$

Figure 3. Select rules of the big-step operational semantics. The general form is $\Pi, e \Downarrow \Pi', v$ where Π and Π' are syntactic stores, e is an expression and v a value.

Contextual equivalence of expressions (in empty worlds) is defined in the standard manner:

Definition 2. *If* $\Xi \mid \emptyset \mid \Gamma \vdash e_i : \tau$*, for* $i = 1, 2$*, then* e_1 *and* e_2 *are* contextually equivalent*, written*

$$\Xi \mid \emptyset \mid \Gamma \vdash e_1 =_{\mathrm{ctx}} e_2 : \tau\,,$$

if, for all closing contexts $C[.] : (\Xi \mid \emptyset \mid \Gamma \vdash \tau) \Rightarrow (\emptyset \mid \emptyset \mid \emptyset \vdash \texttt{int})$*, for all* n*,*

$$\exists \Pi_1.\, \emptyset, C[e_1] \Downarrow \Pi_1, n \Leftrightarrow \exists \Pi_2.\, \emptyset, C[e_2] \Downarrow \Pi_2, n$$

3. Untyped Denotational Semantics

We first present an 'untyped' denotational semantics of our language. By this we mean that all expressions are interpreted by means of a certain complete partial order (cpo) U, and that the interpretation essentially ignores all type information in the language. Since U must in effect allow us to model an untyped variant of our language, we have the familiar requirement for models of the untyped λ-calculus: U must contain a copy of a function space with U itself as the domain. Therefore we construct U by solving a recursive domain equation.

We work with a concrete, domain-theoretic setting: Let $\mathbf{Cppo}_\perp$ be the category of pointed ω-cpos (i.e., cpos containing a least element) and strict, continuous functions. The cpo U is constructed by solving a domain equation of the form

$$U \cong F(U, U)$$

where F is a mixed-variance functor on $\mathbf{Cppo}_\perp$ (see below).

It is not enough that U is any solution to the equation above: the standard methods for solving recursive domain equations give solutions that are so-called *minimal invariants* [28]. In our setting, minimal invariance of U means that there exist continuous functions $\pi_n : U \multimap U$ (one for each $n \in \mathbb{N}$) satisfying, among other properties, that for each $u \in U$,

$$\pi_0 u \sqsubseteq \pi_1 u \sqsubseteq \cdots \sqsubseteq \pi_n u \sqsubseteq \cdots \quad \text{and} \quad \bigsqcup_{n\in\mathbb{N}} \pi_n u = u\,.$$

We say that each element u of U is the limit of its projections $\pi_n u$. The 'projection' functions π_n therefore provide a handle for proving properties about U by induction on n. Moreover, unlike in any earlier work we are aware of, these functions are directly used in the definition of the (untyped) semantics; that will turn out to be essential when we construct our typed semantics in the next section.

We now turn to the formal development.

Definition 3. *Let* $i : F(U,U) \cong U$ *be a minimal invariant of the locally continuous functor* $F : \mathbf{Cppo}_\perp^{op} \times \mathbf{Cppo}_\perp \to \mathbf{Cppo}_\perp$

defined on objects by

$$F(D,E) = 1_\perp \oplus \mathbb{Z}_\perp \oplus (\mathbb{N}\times E)_\perp \oplus (E\otimes E)\oplus(E\oplus E)\oplus E\oplus\big[(\mathbb{N}\overset{fin}{\to} D_\downarrow)_\perp \multimap (\mathbb{N}\overset{fin}{\to} E_\downarrow)_\perp\otimes E\big]_\perp\oplus\big[(\mathbb{N}\overset{fin}{\to} D_\downarrow)_\perp\otimes D\multimap(\mathbb{N}\overset{fin}{\to} E_\downarrow)_\perp\otimes E\big]_\perp.$$

F is assembled from standard components [28] with one exception: For any pointed cpo D we define a new cpo $\mathbb{N}\overset{fin}{\to} D_\downarrow$ by having $s\sqsubseteq s'$ if $\mathrm{dom}(s)=\mathrm{dom}(s')$ and $s(l)\sqsubseteq s'(l)$ for all $l\in\mathrm{dom}(s)$. Lifting then yields the pointed cpo $(\mathbb{N}\overset{fin}{\to} D_\downarrow)_\perp$ and this endofunction on objects of $\mathbf{Cppo}_\perp$ is extended naturally – much as a smash product – to a locally continuous functor $\mathbf{Cppo}_\perp\to\mathbf{Cppo}_\perp$ which is used in the above definition. Here $A\overset{fin}{\to}B$ denotes partial maps with finite domain from a set A to a set B and $D_\downarrow$ is all but the least element of a pointed cpo D.

Notice that the minimal invariant U exists by virtue of Theorem 3.3 of [28]. In accordance with this source we define the continuous map $\delta:(U\multimap U)\to(U\multimap U)$ by $\delta(e)=i\circ F(e,e)\circ i^{-1}$ for any $e\in U\multimap U$. We then define π_n as $\delta^n(\perp)$ for any $n\in\mathbb{N}$; as discussed above, minimality of the invariant means that $\bigsqcup_{n\in\mathbb{N}}\pi_n = id_U$. Note $\pi_m\circ\pi_n=\pi_{m\wedge n}$ for any $m,n\in\mathbb{N}$.

The cpo U is our universal domain: one can intuitively think of U as the domain of all untyped semantic values, analogous to the untyped closed values of our syntactic language. We define $S=(\mathbb{N}\overset{fin}{\to}U_\downarrow)_\perp$ which intuitively is the collection of *states*. The cpo $S\multimap S\otimes U$ models *computations*, i.e., functions from an initial state either diverge or return a state and a semantic value.

From the isomorphism i and the definition of $F(U,U)$ we obtain functions for injecting integers, pairs, functions, etc. into the universal domain: $in_{\mathtt{unit}}:1\to U$, $in_{\mathtt{int}}:\mathbb{Z}\to U$, $in_{\mathtt{ref}}:(\mathbb{N}\times U)\to U$, $in_\times:U\otimes U\multimap U$, $in_+:U\oplus U\multimap U$, $in_\mu:U\multimap U$, $in_\forall:(S\multimap S\otimes U)\to U$, and $in_\to:(S\otimes U\multimap S\otimes U)\to U$. The injection $in_{\mathtt{ref}}$ is explained in more detail below. We use the cpo $S\multimap S\otimes U$ of 'computations' as the domain of $in_\forall$ because, in the untyped semantics, a syntactic value $\Lambda\alpha.e$ is treated simply as a suspension of the computation e: the type argument is ignored.

We now introduce the *semantic locations* as promised:

Definition 4. *For $l\in\mathbb{N}$ we define $\Lambda_l:U\to U$ continuous and order-monic by $\lambda u\in U.\ in_{\mathtt{ref}}(l,u)$. We define $\lambda_l^n=\Lambda_l^n(\perp)$ for any $l,n\in\mathbb{N}$ and finally choose*

$$\lambda_l=\bigsqcup_{n\in\mathbb{N}}\lambda_l^n.$$

Using the the observation that $\pi_{n+1}\circ\Lambda_l=\Lambda_l\circ\pi_n$ holds for any $l,n\in\mathbb{N}$ it is not hard to prove the following properties:

Lemma 5 (Location)**.** *For any $k,l,l',n\in\mathbb{N}$ and $u\in U$ we have:*

(i) $\pi_k(\lambda_l^n)=\lambda_l^{n\wedge k}$, $\pi_n(\lambda_l)=\lambda_l^n$.
(ii) $\lambda_l^{n+1}=\lambda_{l'}^{n+1}\Leftrightarrow l=l'$
(iii) $\pi_n(u)\sqsupseteq\lambda_l^n\Leftrightarrow\pi_n(u)=\lambda_l^n$, $u\sqsupseteq\lambda_l\Leftrightarrow u=\lambda_l$.
(iv) $u\sqsubseteq\lambda_l^n\Leftrightarrow\exists j\le n.\ u=\lambda_l^j$, $u\sqsubseteq\lambda_l\Leftrightarrow u=\lambda_l\vee\exists j.\ u=\lambda_l^j$.

Definition 6. *Any type judgment $\Xi\mid\Delta\mid\Gamma\vdash e:\tau$ is interpreted as $[\![\Xi\mid\Delta\mid\Gamma\vdash e:\tau]\!]\in S\otimes(\mathrm{dom}(\Gamma)\to U_\downarrow)_\perp\multimap S\otimes U$ by induction on the typing derivation, important cases are in Figure 4, see Appendix D for a complete presentation.*

Verification of continuity is tedious but standard with the one exception that we use the Location Lemma and the particular ordering on S in the cases involving references.

If we have $\emptyset\mid\Delta\mid\emptyset\vdash v:\tau$ for a value v then there naturally is a unique $u\in U_\downarrow$ such that $[\![\emptyset\mid\Delta\mid\emptyset\vdash v:\tau]\!]^s=\lfloor s,u\rfloor$ for any $s\in S_\downarrow$, we denote this u by $[\![\Delta\vdash v:\tau]\!]$. Similarly, if we have $\emptyset\mid\Delta\mid\emptyset\vdash\Pi$ we define $[\![\Delta\vdash\Pi]\!]=\lambda l\in\mathrm{dom}(\Delta).\ [\![\Delta\vdash\Pi(l):\Delta(l)]\!]\in S_\downarrow$. With this notation in place we are ready to prove adequacy and soundness of our untyped interpretation:

Theorem 7 (Adequacy)**.** *For $\emptyset\mid\Delta\mid\emptyset\vdash e:\mathtt{int}$ and $\emptyset\mid\Delta\mid\emptyset\vdash\Pi$ we get that*

$$[\![\emptyset\mid\Delta\mid\emptyset\vdash e:\mathtt{int}]\!]^{[\![\Delta\vdash\Pi]\!]}\neq\perp\implies\Pi,e\downarrow.$$

Here and below we use write $\Pi,e\downarrow$ to denote *termination*, i.e., the existence of a syntactic store Π' and a value v such that $\Pi,e\Downarrow\Pi',v$.

Proposition 8 (Soundness)**.** *For $\emptyset\mid\Delta\mid\emptyset\vdash e:\mathtt{int}$ and $\emptyset\mid\Delta\mid\emptyset\vdash\Pi$, any syntactic store Π' and any $n\in\mathbb{N}$ we get*

$$\Pi,e\Downarrow\Pi',n\implies\exists s\in S_\downarrow.\ [\![\emptyset\mid\Delta\mid\emptyset\vdash e:\mathtt{int}]\!]^{[\![\Delta\vdash\Pi]\!]}=\lfloor s,in_{\mathtt{int}}(n)\rfloor.$$

Proving soundness is slightly nontrivial due to the nondeterministic memory allocation of the operational semantics. On the other hand, the problem intuitively comes down to location renaming, i.e., we may perform substitutions of one location for another to make the operational semantics mimic the 'least free' memory allocation of the denotational semantics. Details are deferred to Appendix C .

To prove adequacy we proceed along the lines of the proof of Proposition 5.1 in [28]. But since our operational semantics is not deterministic due to the nondeterministic allocation, we resort to a continuation passing style proof to ensure admissibility of the 'formal approximation' relations. We introduce *continuations* for that purpose, these are just expressions with one free term variable: For a context of type variables Ξ, a world Δ, an expression K, a variable x and types τ_0 and τ_1 we write $\Xi\mid\Delta\vdash K:(x:\tau_0\to\tau_1)$ if we have $\Xi\mid\Delta\mid x:\tau_0\vdash K:\tau_1$ and we refer to K as a continuation. It is a simple yet important property that for any syntactic store Π and any expression e we have

$$\Pi,(\lambda^{\tau_0\to\tau_1}x.\ K)(e)\downarrow\ \Leftrightarrow\ \exists\Pi',v.\ \Pi,e\Downarrow\Pi',v\wedge\Pi',K[v/x]\downarrow$$

We fix some further sets of syntax: For a world Δ and a type τ with $\emptyset\vdash\Delta$ and $\emptyset\vdash\tau$ we let $\mathbf{Val}_\tau^\Delta$ and $\mathbf{Expr}_\tau^\Delta$ denote the set of values and expressions respectively that have type τ under the assumption of Δ and empty contexts. $\mathbf{SynSt}^\Delta$ is the set of syntactic stores Π with $\emptyset\mid\Delta\mid\emptyset\vdash\Pi$ and $\mathbf{Cont}^\Delta_{x:\tau_0\to\tau_1}$ is the set of continuations K with $\emptyset\mid\Delta\vdash K:(x:\tau_0\to\tau_1)$.

Proposition 9. *There is a family of 'formal approximation' relations $\lhd_\tau^\Delta\subset U_\downarrow\times\mathbf{Val}_\tau^\Delta$ with the properties of Figure 5 and with $\{u\in U_\downarrow\mid u\lhd_\tau^\Delta v\}$ chain complete and $\lhd_\tau^\Delta\subset\lhd_\tau^{\Delta'}$ for $\Delta\sqsubseteq\Delta'$.*

Proof. Denote by $\mathbf{UAdmSub}(U)$ all uniform and admissible subsets of U in the sense that they are closed under application of π_n for any $n\in\mathbb{N}$, contain $\perp$ and are chain complete. This constitutes a complete lattice with ordinary set inclusion as ordering since all properties are preserved by intersection, and hence the following is a complete lattice too with pointwise ordering:

$$\mathcal{K}=\Big\{f\in\prod_{(\Delta,\tau)\in\mathbf{World}_\emptyset\times\mathbf{Type}_\emptyset}\mathbf{Val}_\tau^\Delta\to\mathbf{UAdmSub}(U)\ \Big|\ \forall\Delta,\Delta'\in\mathbf{World}_\emptyset\forall\tau\in\mathbf{Type}_\emptyset\forall v\in\mathbf{Val}_\tau^\Delta.\ \Delta\sqsubseteq\Delta'\implies f(\Delta,\tau)(v)\subset f(\Delta',\tau)(v)\Big\}.$$

In Figure 6 we define a monotone map $\Phi:\mathcal{K}^{op}\times\mathcal{K}\to\mathcal{K}$ and mimicking the proof of Theorem 4.16 from [28] one can establish the existence of a fixed point, i.e., a $K\in\mathcal{K}$ with $\Phi(K,K)=K$. We write $u\lhd_\tau^\Delta v$ for $u\in K(\Delta,\tau)(v)\setminus\{\perp\}$ and are done. □

Note that in the proof above we make use of a complete lattice of *functions* from syntactic types (and worlds). This makes it

$$[\![\Xi \mid \Delta \mid \Gamma \vdash x : \tau]\!]^s_\rho = \lfloor s, \rho(x) \rfloor \qquad [\![\Xi \mid \Delta \mid \Gamma \vdash l : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s, \lambda_l \rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau]\!]^s_\rho = \big\lfloor s, in_\forall(\lambda s' \in S_\downarrow.[\![\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau]\!]^{s'}_\rho) \big\rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash e[\tau_1] : \tau_0[\tau_1/\alpha]]\!]^s_\rho = \begin{cases} \varphi(s') & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \forall\alpha.\tau_0]\!]^s_\rho = \lfloor s', in_\forall(\varphi) \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fold}^{\mu\alpha.\tau}(e) : \mu\alpha.\tau]\!]^s_\rho = \begin{cases} \lfloor s', in_\mu(u) \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau[\mu\alpha.\tau/\alpha]]\!]^s_\rho = \lfloor s', u \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fix}^{\tau_0\to\tau_1} f(x).e : \tau_0 \to \tau_1]\!]^s_\rho = \big\lfloor s, (in_\to \circ fix)\big(\lambda\varphi \in S \otimes U \multimap S \otimes U.$$
$$\lambda(s', u) \in S_\downarrow \times U_\downarrow.\ [\![\Xi \mid \Delta \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash e : \tau_1]\!]^{s'}_{\rho[f\mapsto in_\to(\varphi), x\mapsto u]}\big)\big\rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{ref}(e) : \tau\ \mathtt{ref}]\!]^s_\rho = \begin{cases} \lfloor s'[l \mapsto u], \lambda_l \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau]\!]^s_\rho = \lfloor s', u \rfloor, \\ l \notin \mathrm{dom}(s'), \forall l' < l.\ l' \in \mathrm{dom}(s') \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash\, !\, e : \tau]\!]^s_\rho = \begin{cases} \lfloor s', s'(l) \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', \lambda_l \rfloor,\ l \in \mathrm{dom}(s') \\ \lfloor s', \pi_n(s'(l)) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', u \rfloor,\ \pi_{n+1}(u) = \lambda^{n+1}_l, \\ \pi_{n+2}(u) \neq \lambda^{n+2}_l,\ l \in \mathrm{dom}(s'),\ \pi_n(s'(l)) \neq \bot \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash e_0 := e_1 : \mathtt{unit}]\!]^s_\rho = \begin{cases} \lfloor s''[l \mapsto u], in_{\mathtt{unit}}(*) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', \lambda_l \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s'}_\rho = \lfloor s'', u \rfloor,\ l \in \mathrm{dom}(s'') \end{array}\right. \\ \lfloor s''[l \mapsto \pi_n(u)], in_{\mathtt{unit}}(*) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', u' \rfloor \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s'}_\rho = \lfloor s'', u \rfloor,\ \pi_{n+1}(u') = \lambda^{n+1}_l, \\ \pi_{n+2}(u') \neq \lambda^{n+2}_l,\ l \in \mathrm{dom}(s''),\ \pi_n(u) \neq \bot \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

Figure 4. Untyped interpretation, select cases. The general form of the left hand side is $[\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau]\!]^s_\rho$ with $s \in S_\downarrow$ and $\rho \in \mathrm{dom}(\Gamma) \to U_\downarrow$. The right hand side is an element of $S \otimes U$, recall that $S = (Loc \xrightarrow{fin} U_\downarrow)_\bot$.

$$\begin{array}{rcl}
u \lhd^\Delta_{\mathtt{unit}} () & \iff & u = in_{\mathtt{unit}}(*) \\
u \lhd^\Delta_{\mathtt{int}} n & \iff & u = in_{\mathtt{int}}(n) \\
u \lhd^\Delta_{\tau\ \mathtt{ref}} l & \iff & u \sqsubseteq \lambda_l \\
u \lhd^\Delta_{\tau_0\times\tau_1} (v_0, v_1) & \iff & \exists u_0, u_1 \in U_\downarrow.\ u = in_\times(\lfloor u_0, u_1 \rfloor) \wedge u_0 \lhd^\Delta_{\tau_0} v_0 \wedge u_1 \lhd^\Delta_{\tau_1} v_1 \\
u \lhd^\Delta_{\tau_0+\tau_1} \mathtt{inl}^{\tau_0+\tau_1}(v) & \iff & \exists u' \in U_\downarrow.\ u = (in_+ \circ inl)(u') \wedge u' \lhd^\Delta_{\tau_0} v \\
u \lhd^\Delta_{\tau_0+\tau_1} \mathtt{inr}^{\tau_0+\tau_1}(v) & \iff & \exists u' \in U_\downarrow.\ u = (in_+ \circ inr)(u') \wedge u' \lhd^\Delta_{\tau_1} v \\
u \lhd^\Delta_{\mu\alpha.\tau} \mathtt{fold}^{\mu\alpha.\tau}(v) & \iff & \exists u' \in U_\downarrow.\ u = in_\mu(u') \wedge u' \lhd^\Delta_{\tau[\mu\alpha.\tau/\alpha]} v \\
u \lhd^\Delta_{\forall\alpha.\tau} \Lambda\alpha.e & \iff & \exists\varphi \in S \multimap S \otimes U.\ u = in_\forall(\varphi) \wedge \forall\Delta' \sqsupseteq \Delta.\ \forall\tau' \in \mathbf{Type}_\emptyset.\ \varphi \lhd^{\Delta'}_{T\tau[\tau'/\alpha]} e[\tau'/\alpha] \\
u \lhd^\Delta_{\tau_0\to\tau_1} \mathtt{fix}^{\tau_0\to\tau_1} f(x).e & \iff & \exists\varphi \in S \otimes U \multimap S \otimes U.\ u = in_\to(\varphi) \wedge \forall\Delta' \sqsupseteq \Delta.\ \forall u', v.\ u' \lhd^{\Delta'}_{\tau_0} v \Rightarrow \\
& & \qquad \lambda s \in S_\downarrow.\ \varphi(\lfloor s, u' \rfloor) \lhd^{\Delta'}_{T\tau_1} e[\mathtt{fix}^{\tau_0\to\tau_1} f(x).e/f, v/x] \\
s \lhd^\Delta \Pi & \iff & \mathrm{dom}(s) = \mathrm{dom}(\Pi) \wedge \forall l \in \mathrm{dom}(s).\ s(l) \lhd^\Delta_{\Delta(l)} \Pi(l) \\
k \lhd^\Delta_{K\tau} K & \iff & \forall\Delta' \sqsupseteq \Delta.\ \forall u, v, s, \Pi.\ u \lhd^{\Delta'}_\tau v \wedge s \lhd^{\Delta'} \Pi \Rightarrow \big[k(\lfloor s, u \rfloor) \neq \bot \Rightarrow \Pi, (\lambda^{\tau\to\mathtt{int}} x.\ K)(v) \downarrow\big] \\
\varphi \lhd^\Delta_{T\tau} e & \iff & \forall s, \Pi, k, K.\ s \lhd^\Delta \Pi \wedge k \lhd^\Delta_{K\tau} K \Rightarrow \big[k(\varphi(s)) \neq \bot \Rightarrow \Pi, (\lambda^{\tau\to\mathtt{int}} x.\ K)(e) \downarrow\big]
\end{array}$$

Figure 5. Desired properties of an indexed family of 'formal approximation' relations $\lhd^\Delta_\tau \subset U_\downarrow \times \mathbf{Val}^\Delta_\tau$. Also we define three auxiliary families of relations, $\lhd^\Delta \subset S_\downarrow \times \mathbf{SynSt}^\Delta$, $\lhd^\Delta_{K\tau} \subset (S \otimes U \multimap S \otimes U) \times \mathbf{Cont}^\Delta_{x:\tau\to\mathtt{int}}$ and $\lhd^\Delta_{T\tau} \subset (S \multimap S \otimes U) \times \mathbf{Expr}^\Delta_\tau$.

particularly easy to define the interpretation of (closed) recursive and polymorphic types, cf., the definition of Φ in Figure 6, and means that we find the interpretation of all types by taking one fixed point of Φ, rather than via a nested sequence of fixed points as in, e.g., [13, 14]. This idea of using a function-space lattice was also used in the first author's earlier work [11], albeit more implicitly.

Proposition 10. *Given $\Xi \mid \Delta \mid \Gamma \vdash e : \tau$ with $\Xi = \alpha_1, \ldots, \alpha_m$ and $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$. Pick $\sigma_1, \ldots, \sigma_m \in \mathbf{Type}_\emptyset$ and denote application of the substitution $[\sigma_1/\alpha_1, \ldots, \sigma_m/\alpha_m]$ by overlining. For any $\Delta_0 \in \mathbf{World}_\emptyset$ with $\Delta_0 \sqsupseteq \overline{\Delta}$ and any $u_1, \ldots, u_n \in U_\downarrow$ and any values $v_1, \ldots, v_n$ with $u_i \lhd^{\Delta_0}_{\overline{\tau_i}} v_i$ for all $1 \leq i \leq n$, we have*

$$\lambda s \in S_\downarrow.[\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau]\!]^s_\rho \lhd^{\Delta_0}_{T\overline{\tau}} \overline{e}[v_1/x_1, \ldots, v_n/x_n]$$

with $\rho = [x_1 \mapsto u_1, \ldots, x_n \mapsto u_n] \in \mathrm{dom}(\Gamma) \to U_\downarrow$.

Loosely, this proposition says that any expression is related to itself. Applying the identity continuation it is not hard to see that it has adequacy as a corollary as we have $[\![\Delta \vdash v : \tau]\!] \lhd^\Delta_\tau v$ for any value v with $\emptyset \mid \Delta \mid \emptyset \vdash v : \tau$.

Proof. We prove the proposition by induction on the typing derivation, details follow for a few cases. For the case of memory alloca-

$$
\begin{aligned}
\Phi(\mathcal{R},\mathcal{S})(\Delta,\mathtt{unit})(()) &= \{\bot\}\cup\{in_{\mathtt{unit}}(*)\}\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\mathtt{int})(n) &= \{\bot\}\cup\{in_{\mathtt{int}}(n)\}\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\tau\ \mathtt{ref})(l) &= \{u\in U\mid u\sqsubseteq\lambda_l\}\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\tau_0\times\tau_1)((v_0,v_1)) &= \{\bot\}\cup(in_\times\circ\lfloor-\rfloor)(\mathcal{S}(\Delta,\tau_0)(v_0)\setminus\{\bot\}\times\mathcal{S}(\Delta,\tau_1)(v_1)\setminus\{\bot\})\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\tau_0+\tau_1)(\mathtt{inl}(v)) &= \{\bot\}\cup(in_+\circ inl)(\mathcal{S}(\Delta,\tau_0)(v)\setminus\{\bot\})\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\tau_0+\tau_1)(\mathtt{inr}(v)) &= \{\bot\}\cup(in_+\circ inr)(\mathcal{S}(\Delta,\tau_1)(v)\setminus\{\bot\})\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\mu\alpha.\tau)(\mathtt{fold}(v)) &= \{\bot\}\cup in_\mu(\mathcal{S}(\Delta,\tau[\mu\alpha.\tau/\alpha])(v)\setminus\{\bot\})\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\forall\alpha.\tau)(\Lambda\alpha.e) &= \{\bot\}\cup\big\{in_\forall(\varphi)\mid\varphi\in S\multimap S\otimes U\wedge\\
&\qquad\forall\Delta'\sqsupseteq\Delta.\forall\tau'\in\mathbf{Type}_\emptyset.\varphi\in\Phi^T(\mathcal{R},\mathcal{S})(\Delta',\tau[\tau'/\alpha])(e[\tau'/\alpha])\big\}\\
\Phi(\mathcal{R},\mathcal{S})(\Delta,\tau_0\to\tau_1)(\mathtt{fix}^{\tau_0\to\tau_1}f(x).e) &= \{\bot\}\cup\big\{in_\to(\varphi)\mid\varphi\in S\otimes U\multimap S\otimes U\wedge\\
&\qquad\forall\Delta'\sqsupseteq\Delta.\forall v\,\forall u\in\mathcal{R}(\Delta',\tau_0)(v)\setminus\{\bot\}.\\
&\qquad\lambda s\in S_\downarrow.\,\varphi(\lfloor s,u\rfloor)\in\Phi^T(\mathcal{R},\mathcal{S})(\Delta',\tau_1)\\
&\qquad(e[\mathtt{fix}^{\tau_0\to\tau_1}f(x).e/f,v/x])\big\}\\
\Phi^S(\mathcal{S})(\Delta)(\Pi) &= \{\bot\}\cup\big\{s\in S_\downarrow\mid\mathrm{dom}(s)=\mathrm{dom}(\Pi)\wedge\forall l\in\mathrm{dom}(s).\,s(l)\in\mathcal{S}(\Delta,\Delta(l))(\Pi(l))\setminus\{\bot\}\big\}\\
\Phi^K(\mathcal{R},\mathcal{S})(\Delta,\tau)(K) &= \big\{k\in S\otimes U\multimap S\otimes U\mid\forall\Delta'\sqsupseteq\Delta.\forall v,\Pi.\\
&\qquad\forall u\in\mathcal{R}(\Delta',\tau)(v)\setminus\{\bot\}.\forall s\in\Phi^S(\mathcal{R})(\Delta')(\Pi)\setminus\{\bot\}.\\
&\qquad k(\lfloor s,u\rfloor)\neq\bot\Rightarrow\Pi,(\lambda^{\tau\to\mathtt{int}}x.\,K)(v)\downarrow\big\}\\
\Phi^T(\mathcal{R},\mathcal{S})(\Delta,\tau)(e) &= \big\{\varphi\in S\multimap S\otimes U\mid\forall\Pi,K.\\
&\qquad\forall s\in\Phi^S(\mathcal{R})(\Delta)(\Pi)\setminus\{\bot\}.\forall k\in\Phi^K(\mathcal{S},\mathcal{R})(\Delta,\tau)(K).\\
&\qquad k(\varphi(s))\neq\bot\Rightarrow\Pi,(\lambda^{\tau\to\mathtt{int}}x.\,K)(e)\downarrow\big\}
\end{aligned}
$$

Figure 6. Definition of $\Phi:\mathcal{K}^{op}\times\mathcal{K}\to\mathcal{K}$ using three auxiliary maps $\Phi^S:\mathcal{K}\to\prod_{\Delta\in\mathbf{World}_\emptyset}\mathbf{SynSt}^\Delta\to\mathcal{P}(S)$, $\Phi^K:\mathcal{K}^{op}\times\mathcal{K}\to\prod_{(\Delta,\tau)\in\mathbf{World}_\emptyset\times\mathbf{Type}_\emptyset}\mathbf{Cont}^\Delta_{x:\tau\to\mathtt{int}}\to\mathcal{P}(S\otimes U\multimap S\otimes U)$ and $\Phi^T:\mathcal{K}^{op}\times\mathcal{K}\to\prod_{(\Delta,\tau)\in\mathbf{World}_\emptyset\times\mathbf{Type}_\emptyset}\mathbf{Expr}^\Delta_\tau\to\mathcal{P}(S\multimap S\otimes U)$.

tion, consider

$$\frac{\Xi\mid\Delta\mid\Gamma\vdash e:\tau}{\Xi\mid\Delta\mid\Gamma\vdash\mathtt{ref}(e):\tau\ \mathtt{ref}}$$

and assume that the proposition holds for the premise. Choose types $\sigma_1,\ldots,\sigma_m$, $\Delta_0\sqsupseteq\overline{\Delta}$ and $u_1,\ldots,u_n$ elements of $U_\downarrow$, values $v_1,\ldots,v_n$ and $\rho\in\mathrm{dom}(\Gamma)\to U_\downarrow$ as stated. Furthermore pick s,Π,k and K with $s\lhd^{\Delta_0}\Pi$ and $k\lhd^{\Delta_0}_{K\overline{\tau}\ \mathtt{ref}}K$. We now assume that

$$k([\![\emptyset\mid\overline{\Delta}\mid\overline{\Gamma}\vdash\mathtt{ref}(\overline{e}):\overline{\tau}\ \mathtt{ref}]\!]^s_\rho)\neq\bot$$

and are to prove that

$$\Pi,(\lambda^{\overline{\tau}\ \mathtt{ref}\to\mathtt{int}}x.\,K)(\mathtt{ref}(\overline{e}[v_1/x_1,\ldots,v_n/x_n]))\downarrow.$$

The map $\lambda(s,u)\in S_\downarrow\times U_\downarrow.\ k(\lfloor s[l\mapsto u],\lambda_l\rfloor)$ where we choose $l\in\mathbb{N}$ minimal such that $l\notin\mathrm{dom}(s)$ defines a map $k'\in S\otimes U\multimap S\otimes U$. Also we define a continuation $K'\in\mathbf{Cont}^{\Delta_0}_{x:\overline{\tau}\to\mathtt{int}}$ by $K'=(\lambda^{\overline{\tau}\ \mathtt{ref}\to\mathtt{int}}x.\,K)(\mathtt{ref}(x))$ and by the induction hypothesis it suffices to show that $k'\lhd^{\Delta_0}_{K\overline{\tau}}K'$. According to definition we pick $\Delta'\sqsupseteq\Delta_0$ and u',v',s',Π' with $u'\lhd^{\Delta'}_{\overline{\tau}}v'$ and $s'\lhd^{\Delta'}\Pi'$, we assume that $k'(\lfloor s',u'\rfloor)\neq\bot$ and aim to prove that $\Pi',(\lambda^{\overline{\tau}\to\mathtt{int}}x.\,K')(v')\downarrow$. We remark that

$$\bot\neq k'(\lfloor s',u'\rfloor)=k(\lfloor s'[l'\mapsto u'],\lambda_{l'}\rfloor)$$

for $l'\in\mathbb{N}$ with $l'\notin\mathrm{dom}(s')$. By $s'\lhd^{\Delta'}\Pi'$ we have $l'\notin\mathrm{dom}(s')=\mathrm{dom}(\Pi')$ and hence $\Pi',\mathtt{ref}(v')\Downarrow\Pi'[l'\mapsto v'],l'$. We obviously have $\Delta'[l'\mapsto\overline{\tau}]\sqsupseteq\Delta'$ and $\lambda_{l'}\lhd^{\Delta'[l'\mapsto\overline{\tau}]}_{\overline{\tau}\ \mathtt{ref}}l'$. Also for any $l\in\mathrm{dom}(s')\cup\{l'\}$ we have $s'[l'\mapsto u'](l)\lhd^{\Delta'}_{\Delta'[l'\mapsto\overline{\tau}](l)}\Pi'[l'\mapsto v'](l)$ and hence $s'[l'\mapsto u'](l)\lhd^{\Delta'[l'\mapsto\overline{\tau}]}_{\Delta'[l'\mapsto\overline{\tau}](l)}\Pi'[l'\mapsto v'](l)$ too which means that $s'[l'\mapsto u']\lhd^{\Delta'[l'\mapsto\overline{\tau}]}\Pi'[l'\mapsto v']$ and we are done as we initially assumed that $k\lhd^{\Delta_0}_{K\overline{\tau}\ \mathtt{ref}}K$.

This case warrants some comments: It is here that we need the continuations to 'work' in all future worlds, in the other cases this property is just pushed through the proof. Also this is where we rely on the property that the formal approximations grow with larger worlds. Finally note that the operational semantics may allocate any free location, in particular we can pick the least free to match the behavior of the denotational semantics.

Consider now the case of lookup, i.e., consider

$$\frac{\Xi\mid\Delta\mid\Gamma\vdash e:\tau\ \mathtt{ref}}{\Xi\mid\Delta\mid\Gamma\vdash!\,e:\tau}$$

and assume that the proposition holds for the premise. Choose types $\sigma_1,\ldots,\sigma_m$, a world $\Delta_0\sqsupseteq\Delta$ and $u_1,\ldots,u_n$ elements of $U_\downarrow$, values $v_1,\ldots,v_n$ and $\rho\in\mathrm{dom}(\Gamma)\to U_\downarrow$ as stated. Furthermore pick s,Π,k and K with $s\lhd^{\Delta_0}\Pi$ and $k\lhd^{\Delta_0}_{K\overline{\tau}}K$. We now assume that

$$k([\![\emptyset\mid\overline{\Delta}\mid\overline{\Gamma}\vdash!\,\overline{e}:\overline{\tau}]\!]^s_\rho)\neq\bot$$

and are to prove that

$$\Pi,(\lambda^{\overline{\tau}\to\mathtt{int}}x.\,K)(!\,\overline{e}[v_1/x_1,\ldots,v_n/x_n])\downarrow.$$

We define $k'\in S\otimes U\multimap S\otimes U$ by mapping any $(s,u)\in S_\downarrow\times U_\downarrow$ to $S\otimes U$ by copying the interpretation of lookup and applying k:

$$\begin{cases}k(\lfloor s,s(l)\rfloor) & u=\lambda_l,l\in\mathrm{dom}(s)\\ k(\lfloor s,\pi_n(s(l))\rfloor) & \pi_{n+1}(u)=\lambda_l^{n+1},\pi_n(s(l))\neq\bot,\\ & \pi_{n+2}(u)\neq\lambda_l^{n+2},l\in\mathrm{dom}(s)\\ \bot & \text{otherwise}\end{cases}$$

Similarly we define $K'\in\mathbf{Cont}^{\overline{\Delta}}_{x:\overline{\tau}\ \mathtt{ref}\to\mathtt{int}}$ by

$$K'=(\lambda^{\overline{\tau}\to\mathtt{int}}x.\,K)(!\,x)$$

and by induction it suffices to show that $k'\lhd^{\Delta_0}_{K\overline{\tau}\ \mathtt{ref}}K'$.

For that purpose we pick $\Delta'\sqsupseteq\Delta_0$, u',v',s' and Π' with $u'\lhd^{\Delta'}_{\overline{\tau}\ \mathtt{ref}}v'$ and $s'\lhd^{\Delta'}\Pi'$, we assume that $k'(\lfloor s',u'\rfloor)\neq\bot$ and have to prove that $\Pi',(\lambda^{\overline{\tau}\ \mathtt{ref}\to\mathtt{int}}x.\,K')(v')\downarrow$. From $u'\lhd^{\Delta'}_{\overline{\tau}\ \mathtt{ref}}v'$ we deduce that there is $l'\in\mathrm{dom}(\Delta')$ with $v'=l',\Delta'(l')=\overline{\tau}$ and $u'\sqsubseteq\lambda_{l'}$. Also $s'\lhd^{\Delta'}\Pi'$ yields that $l'\in\mathrm{dom}(\Delta')=\mathrm{dom}(s')=$

$\mathrm{dom}(\Pi')$ and this gives $\Pi', !\ l' \Downarrow \Pi', \Pi'(l')$ and we also have $s'(l') \lhd^{\Delta'}_{\overline{\tau}} \Pi'(l')$.

Assume now that $u' = \lambda_{l'}$. From the definition of k' we get that $\bot \neq k(\lfloor s', s'(l') \rfloor)$ and we may use the original assumption $k \lhd^{\Delta_0}_{K\overline{\tau}} K$ to prove the required. Suppose now that $u' \neq \lambda_{l'}$, i.e., that $u' = \lambda^{n'+1}_{l'}$ for some $n' \in \mathbb{N}$. We get $\bot \neq k(\lfloor s', \pi_{n'}(s'(l')) \rfloor)$ which yields $\bot \neq k(\lfloor s', s'(l') \rfloor)$ by monotonicity and we are back on the above track. □

4. Typed Denotational Semantics

In this section we present the typed possible world semantics. As mentioned in the Introduction, to reason about parametricity we need to give a semantics not only of closed types (as sufficed for proving adequacy in the previous section) but also of *open* types. This has two consequences for the technical development which we explain before proceeding with the technical development proper.

Recall first that the overall idea is to define the semantics of types by means of world-indexed binary relations over the universal domain U. These relations will be both *uniform* and *admissible*: such relations are completely determined by their elements of the form $(\pi_n u,\ \pi_n u')$. One explanation of the formal construction below is therefore the following. To define the various relations that together constitute the semantics of types, it suffices to determine for each $n \in \mathbb{N}$ whether the pairs of the form $(\pi_n u,\ \pi_n u')$ belong to the various relations; this can be done by induction on n. For all types except reference types, this approach works well due to properties of the π_n. For example, $\pi_{n+1}(in_+(inl\, u)) = in_+(inl\, (\pi_n u))$ and $\pi_{n+1}(in_\times(u_1, u_2)) = in_\times(\pi_n u_1, \pi_n u_2)$.

For the case of reference types, the idea is roughly that, for a type $\Xi \vdash \tau$, for a world $\Delta \in \mathbf{World}_\Xi$, and for semantics types $\overline{\nu}$ corresponding to the type environment Ξ,

$$(u, u') \in [\![\tau\ \mathtt{ref}]\!]_\Xi(\overline{\nu})(\Delta)$$

if and only if,

$$\exists l \in \mathrm{dom}(\Delta).\, u = u' = l \wedge [\![\tau]\!]_\Xi(\overline{\nu})(\Delta) = [\![\Delta(l)]\!]_\Xi(\overline{\nu})(\Delta).$$

That is, u and u' should be the same location l and, moreover, the interpretation of the type τ should be the same as the interpretation of the type $\Delta(l)$ found in the store type Δ. The latter is, of course, to ensure sound modelling of lookup and assignment.

The problem with the above definition is that it is not inductive: to determine whether the pair $(\pi_{n+1}\, u, \pi_{n+1}\, u')$ belongs to $[\![\tau\ \mathtt{ref}]\!]_\Xi(\overline{\nu})(\Delta)$, we need to know the entire relations $[\![\tau]\!]_\Xi(\overline{\nu})(\Delta)$ and $[\![\Delta(l)]\!]_\Xi(\overline{\nu})(\Delta)$, not just their elements of the form $(\pi_n u_0, \pi_n u_0')$. That is the reason for introducing semantic locations λ_l and λ^n_l. By means of these we can refine the above idea to what you see in Figure 8. The idea is that approximative locations λ^{n+1}_l are related in case the interpretations of types agree up to level n and that ideal locations λ_l are related in case the interpretations really are equal. (As shown, the real definition also includes a quantification over future worlds, but that is not related to what we are discussing here.)

More formally, the problem with the definition of $[\![\tau\ \mathtt{ref}]\!]$ above is that it prevents one from proving the existence of the family of logical relations constituting the semantics of types. The usual proof by fixed-point induction and minimal invariance [28] does not go through: indeed, for an earlier variant of the setup presented here, we could actually give a formal proof showing that relations satisfying such conditions do not exist.

Clearly, there are some relations between our semantic locations and step-indexed approaches to recursive types [2, 5, 6]; see Subsection 4.1 for comments on how one can attempt to make the connection formal.

The second consequence of interpreting open types is also related to the use of world-indexed relations. It has to do with how we should interpret quantified types $\forall\alpha.\,\tau$. When one is only interested in a semantics of closed types, one can define the semantics of $\forall\alpha.\,\tau$ simply as the intersection over all *syntactic types* $\bigcap_{\sigma\in\mathbf{Type}}[\![\tau[\sigma/\alpha]]\!]$, as we essentially did in the adequacy proof earlier. For a semantics of open types, one typically defines the semantics of $\forall\alpha.\,\tau$ by a big intersection over some universe ST of *semantic types* (think of ST as the set of all admissible relations) $\bigcap_{\nu\in\mathrm{ST}}[\![\tau]\!](\nu)$. However, in our case the meaning of a type *depends on* the current world. One attempt to accommodate this dependency would be to interpret a closed universal type $\forall\alpha.\tau$ essentially as an intersection over semantic types indexed by closed worlds: $\bigcap_{\nu\in\mathbf{World}_\emptyset\to\mathrm{ST}}[\![\tau]\!](\nu)$. The problem with this attempt is that in the natural Kripke-style definition of $[\![\tau]\!](\nu)$ one needs to apply ν not only to closed worlds, but to worlds containing free occurrences of α. Worse, if τ contains nested universal types, one needs to apply ν to worlds containing additional new type variables. For example, if α occurs in τ below a universal quantifier $\forall\beta$, then one needs to be able to apply ν to an arbitrary world $\Delta \in \mathbf{World}_{\alpha,\beta}$.

Informally, one attempt to interpret such an occurrence of α would be

$$[\![\alpha]\!]_{\alpha,\beta}(\nu,\nu')(\Delta) = \nu\,(\nu,\nu')\,(\Delta)\,,$$

i.e., to interpret α in a world $\Delta \in \mathbf{World}_{\alpha,\beta}$, one applies ν not only to Δ, but also to the ν and ν' that interpret α and β, respectively. But this attempt introduces a circularity: it is not clear what the formal definition of ν should be, since ν must now be applicable to itself as well as an arbitrary other ν'. To break the circularity in the above interpretation of α, we instead apply ν to *the interpretation function itself*, partially applied to (ν,ν') and Δ:

$$[\![\alpha]\!]_{\alpha,\beta}(\nu,\nu')(\Delta) = \nu\,[\lambda\tau_0 \in \mathbf{Type}_\emptyset.[\![\tau_0]\!]_{\alpha,\beta}(\nu,\nu')(\Delta)]\,.$$

In this way (ν,ν') and Δ are indirectly passed to ν. (Notice that the τ_0 on the right hand side contains fewer free type variables than α.)

In summary, we use a novel interpretation of types, where $\forall\alpha.\,\tau$ is interpreted essentially by a big intersection

$$\bigcap_{\nu\in(\mathbf{Type}\to\mathrm{ST})\to\mathrm{ST}} [\![\tau]\!](\nu)$$

over semantic types *indexed over* a function that can interpret closed types (i.e., types with one fewer type variable than τ). This essentially allows us to *delay* the choice of semantic type until we know how the world should be interpreted.

We now continue with the formal development after which we discuss an alternative approach to dealing with the second issue mentioned above and then present some examples. In the formal development we make use of admissible relations that satisfy a couple of additional conditions, uniformity and strictness. Uniformity is typical for interpretations of polymorphism and recursive types [4]; strictness is used to capture contextual equivalence (also used in [11]).

Definition 11. *Let* $\mathbf{UARel}(U)$ *be the set of binary relations on* U *that relate* $\bot$ *to* $\bot$ *and to nothing else, are closed under* π_n *for any* $n \in \mathbb{N}$ *and are closed under taking least upper bounds of chains.*

We speak of *uniform* and *admissible* relations over U. It is not hard to see that $\mathbf{UARel}(U)$ with ordinary set inclusion constitutes a complete lattice as all properties are preserved by intersection. We can now define the *semantic closed types*:

Definition 12. *For any context of type variables* Ξ *we let* $\mathbf{SCT}_\Xi$ *be the monotone maps* ν *of*

$$\left[\mathbf{Type}_\Xi \to \mathbf{UARel}(U)\right] \overset{mon}{\to} \mathbf{UARel}(U)$$

for which it holds that for any two arguments $\varphi, \varphi' \in \mathbf{Type}_\Xi \to \mathbf{UARel}(U)$ *and any* $n \in \mathbb{N}$ *we have that*

$$\big[\forall \tau \in \mathbf{Type}_\Xi.\ \varphi(\tau) \overset{n}{=} \varphi'(\tau)\big] \implies \nu(\varphi) \overset{n}{=} \nu(\varphi').$$

Above and below we use $R \overset{n}{=} S$ for $n \in \mathbb{N}$ and $R, S \in \mathbf{UARel}(U)$ to mean that $\pi_n(R) \subset S$ and $\pi_n(S) \subset R$ hold; we shall also use $R \overset{n}{\subset} S$ to denote just the first of these properties. Intuitively, the demand that 'n-equality' be preserved by semantic closed types allows us to work with approximations of types – a property we need to prove the existence of the desired interpretation of types. For any context of type variables Ξ and any syntactic type $\sigma \in \mathbf{Type}_\Xi$ we furthermore define

$$\nu_\Xi(\sigma) = \lambda\varphi \in \mathbf{Type}_\Xi \to \mathbf{UARel}(U).\ \varphi(\sigma)$$

and it is easily verified that we indeed have $\nu_\Xi(\sigma) \in \mathbf{SCT}_\Xi$.

We shall need a few minor definitions: For every type in context $\alpha_1, \ldots, \alpha_m \vdash \tau$ we define the following measure

$$\#(\alpha_1, \ldots, \alpha_m \vdash \tau) = \min\{0 \le n \le m \mid \alpha_1, \ldots, \alpha_n \vdash \tau\}\,;$$

recall here that type contexts are ordered lists. For a context of type variables $\Xi = \alpha_1, \alpha_2, \ldots, \alpha_m$ we write $\mathbf{SCT}^\Xi$ as shorthand for

$$\mathbf{SCT}_\emptyset \times \mathbf{SCT}_{\alpha_1} \times \mathbf{SCT}_{\alpha_1,\alpha_2} \times \cdots \times \mathbf{SCT}_{\alpha_1,\alpha_2,\ldots,\alpha_{m-1}}.$$

Finally, assume that we have type contexts Ξ, Ξ', $\overline{\nu} \in \mathbf{SCT}^\Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta \in \mathbf{World}_\Xi$ and $\Delta' \in \mathbf{World}_{\Xi'}$. We now define $[\Xi|\overline{\nu}|\Delta] \sqsubseteq [\Xi'|\overline{\nu}'|\Delta']$ to denote the existence of type variables $\beta_1, \beta_2, \ldots, \beta_m$ and semantic closed types $\nu_1 \in \mathbf{SCT}_\Xi$, $\nu_2 \in \mathbf{SCT}_{\Xi,\beta_1}$ up to $\nu_m \in \mathbf{SCT}_{\Xi,\beta_1,\ldots,\beta_{m-1}}$ such that $\Xi, \beta_1, \beta_2, \ldots, \beta_m = \Xi'$ and $(\overline{\nu}, \nu_1, \nu_2, \ldots, \nu_m) = \overline{\nu}'$ and $\Delta \sqsubseteq \Delta'$. This definition captures our typed notion of 'future' worlds: Not only does the world itself grow, we also allow extension of the context of type variables with corresponding semantic closed types. We are now ready to define the lattice $\mathcal{L}$ of type interpretations:

Definition 13. *We define a complete lattice by pointwise ordering*

$$\mathcal{L} = \Big\{ f \in \prod_\Xi \mathbf{SCT}^\Xi \to \mathbf{Type}_\Xi \to \mathbf{World}_\Xi \to \mathbf{UARel}(U) \;\Big|\; [\Xi|\overline{\nu}|\Delta] \sqsubseteq [\Xi'|\overline{\nu}'|\Delta'] \wedge \tau \in \mathbf{Type}_\Xi \implies f(\Xi)(\overline{\nu})(\tau)(\Delta) \subset f(\Xi')(\overline{\nu}')(\tau)(\Delta') \Big\}.$$

We also define $\Psi : \mathcal{L}^{op} \times \mathcal{L} \to \mathcal{L}$ *monotone in Figure 7: The definition of* $\Psi(\mathcal{R}, \mathcal{S})(\Xi)(\overline{\nu})(\tau)(\Delta)$ *is by induction on* $\#(\Xi \vdash \tau)$.

Members of $\mathcal{L}$ interpret types, and as we deal with open types we parameterize over semantic closed types to 'plug in' for the free variables as well as over worlds. The intuition behind defining Ψ by induction on $\#(\Xi \vdash \tau)$ is that $\Psi(\mathcal{R}, \mathcal{S})(\Xi)(\overline{\nu})(\tau)(\Delta)$ is obviously well defined if τ is not a type variable, and from that point on we can interpret type variables in the order they occur in Ξ. It is worth noticing that the definition of Ψ in the cases of references, polymorphic types and functions has been carefully tailored to comply with monotonicity property in the definition of $\mathcal{L}$: the quantification over 'future' worlds has been baked in.

To obtain the desired interpretation of types we could just appeal to the approach of the proof of Theorem 4.16 of [28] as done above in the proof of Theorem 7. Instead we construct the sequence of approximations and the fixed point by hand – we proceed in the style of Kleene's fixed point theorem rather than by appeal to the Knaster-Tarski fixed point theorem. The difference is merely one of presentation: The present approach is less general but arguably has a more constructive feel to it that goes well with the intuition of the semantic locations.

We define $\mathcal{R}_0 \in \mathcal{L}$ as constant $\{(\bot, \bot)\}$, $\mathcal{S}_0 \in \mathcal{L}$ as constant $\{(u, u') \in U \times U \mid \forall n \in \mathbb{N}.\ \pi_n(u) = \bot \Leftrightarrow \pi_n(u') = \bot\}$ and inductively $\mathcal{R}_{n+1} = \Psi(\mathcal{S}_n, \mathcal{R}_n) \in \mathcal{L} \ni \Psi(\mathcal{R}_n, \mathcal{S}_n) = \mathcal{S}_{n+1}$ for all $n \in \mathbb{N}$. By induction we get the crucial fact that $\mathcal{R}_n \overset{n}{=} \mathcal{S}_n$, for all $n \in \mathbb{N}$, and choosing $\nabla = \cap_{n\in\mathbb{N}} S_n$ yields a fixed point of Ψ, i.e., $\Psi(\nabla, \nabla) = \nabla$. We are now able to interpret types: We shall denote $\nabla(\Xi)(\tau)(\overline{\nu})(\Delta) \setminus \{(\bot, \bot)\}$ by $[\![\tau]\!]_\Xi(\overline{\nu})(\Delta)$ and it is immediate that this interpretation has the properties listed in Figure 8. Also the following is a consequence of the construction:

Lemma 14 (Monotonicity)**.** *For* $[\Xi|\overline{\nu}|\Delta] \sqsubseteq [\Xi'|\overline{\nu}'|\Delta']$ *and* $\tau \in \mathbf{Type}_\Xi$ *we have* $[\![\tau]\!]_\Xi(\overline{\nu})(\Delta) \subset [\![\tau]\!]_{\Xi'}(\overline{\nu}')(\Delta')$.

We remark that (1) as for the adequacy proof, we again make use of a complete lattice of *functions*, cf., Definition 13; and (2) we would need to prove the existence of the logical relations using a proof as the one above, even if we had left out recursive types from the language. In that case, we could define the relation by induction on the type, for all other type constructors but `ref` – for `ref` it would not be possible, since the definition in the case for τ `ref` involves the meaning of arbitrary types in future worlds. This is typical for models of higher-order store in which one can have recursion through the store, even without recursive types.

Lemma 15 (Degenerate Substitution)**.** *With natural ranges of variables, in particular* $\tau \in \mathbf{Type}_{\Xi,\alpha,\Xi'}$, $\Delta \in \mathbf{World}_{\Xi,\alpha,\Xi'}$ *and* $\sigma \in \mathbf{Type}_\Xi$*, we have that*

$$[\![\tau]\!]_{\Xi,\alpha,\Xi'}(\overline{\nu}, \nu_\Xi(\sigma), \overline{\nu}')(\Delta) = [\![\tau[\sigma/\alpha]]\!]_{\Xi,\alpha,\Xi'}(\overline{\nu}, \nu_\Xi(\sigma), \overline{\nu}')(\Delta).$$

It is easily proved by induction that the property holds for $\mathcal{S}_n$ for all $n \in \mathbb{N}$ and the above lemma follows – the validity of this lemma is partial justification for the definition of interpretation of type variables. We refer to the lemma as *degenerate* because we perform no substitution in the world Δ and do not remove α and $\nu_\Xi(\sigma)$ on the right hand side. It is possible to state and prove a more standard substitution lemma, but we shall not need that.

The main result of this section is a 'fundamental theorem of logical relations,' intuitively stating that every well-typed term is related to itself. First some notation: For any two contexts of type variables Ξ and Ξ' we write $\Xi \subset \Xi'$ if all variables of Ξ occur in Ξ', i.e., if the inclusion holds when interpreting the contexts as sets.

Definition 16. *Two expressions in context* $\Xi \mid \Delta \mid \Gamma \vdash e_i : \tau$ *($i = 1, 2$) are* semantically related*, written*

$$\Xi \mid \Delta \mid \Gamma \vdash e_1 \sim e_2 : \tau\,,$$

if for all $\Xi' \supset \Xi$*, all* $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$*, all* $\Delta' \in \mathbf{World}_{\Xi'}$ *with* $\Delta' \sqsupseteq \Delta$ *and all* $\rho, \rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ *such that* $(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi'}(\overline{\nu}')(\Delta')$ *for each* $x \in \mathrm{dom}(\Gamma)$*, we have that the pair*

$$\big(\lambda s \in S_\downarrow.[\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^s_\rho, \lambda s \in S_\downarrow.[\![\Xi \mid \Delta \mid \Gamma \vdash e_2 : \tau]\!]^s_{\rho'}\big)$$

belongs to $[\![\tau]\!]^T_{\Xi'}(\overline{\nu}')(\Delta')$.

Theorem 17. $\Xi \mid \Delta \mid \Gamma \vdash e : \tau$ *implies* $\Xi \mid \Delta \mid \Gamma \vdash e \sim e : \tau$.

Proof. The proof is by induction on the typing derivation, we shall present three decisive cases. Consider the lookup case, i.e., consider

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \mathtt{ref}}{\Xi \mid \Delta \mid \Gamma \vdash !\,e : \tau}$$

and assume that the proposition holds for the premise. We pick arbitrary $\Xi' \supset \Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta' \in \mathbf{World}_{\Xi'}$ with $\Delta' \sqsupseteq \Delta$, and $\rho, \rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ as specified in the definition of semantic relatedness. Also we take arbitrary $(s, s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k, k') \in [\![\tau]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$ and we have to prove that either

$$k\big([\![\Xi \mid \Delta \mid \Gamma \vdash !\,e : \tau]\!]^s_\rho\big) = \bot = k'\big([\![\Xi \mid \Delta \mid \Gamma \vdash !\,e : \tau]\!]^{s'}_{\rho'}\big)$$

or that the left hand side and the right hand side both terminate and moreover both yield the value $in_{\mathtt{int}}(n)$ for some $n \in \mathbb{N}$,

$$
\begin{array}{rcl}
\Psi(\mathcal{R},\mathcal{S})(\alpha_1,\ldots,\alpha_m)(\nu_1,\ldots,\nu_m)(\alpha_n)(\Delta) & = & \nu_n\big[\lambda\tau\in\mathbf{Type}_{\alpha_1,\ldots,\alpha_{n-1}}.\,\Psi(\mathcal{R},\mathcal{S})(\alpha_1,\ldots,\alpha_m)(\nu_1,\ldots,\nu_m)(\tau)(\Delta)\big]\\
\Psi(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\mathtt{1})(\Delta) & = & \{(\bot,\bot)\}\cup\{(in_{\mathtt{unit}}(*),in_{\mathtt{unit}}(*))\}\\
\Psi(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\mathtt{int})(\Delta) & = & \{(\bot,\bot)\}\cup\{(in_{\mathtt{int}}(n),in_{\mathtt{int}}(n))\mid n\in\mathbb{N}\}\\
\Psi(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\tau\ \mathtt{ref})(\Delta) & = & \{(\bot,\bot)\}\cup\\
 & & \big\{(\lambda_l^{n+1},\lambda_l^{n+1})\mid l\in\mathrm{dom}(\Delta)\wedge n\in\mathbb{N}\wedge\\
 & & \qquad\forall[\Xi'|\overline{\nu}'|\Delta']\sqsupseteq[\Xi|\overline{\nu}|\Delta].\,\mathcal{R}(\Xi')(\overline{\nu}')(\tau)(\Delta')\overset{n}{\subset}\mathcal{S}(\Xi')(\overline{\nu}')(\Delta'(l))(\Delta')\wedge\\
 & & \qquad\mathcal{R}(\Xi')(\overline{\nu}')(\Delta'(l))(\Delta')\overset{n}{\subset}\mathcal{S}(\Xi')(\overline{\nu}')(\tau)(\Delta')\big\}\cup\\
 & & \big\{(\lambda_l,\lambda_l)\mid l\in\mathrm{dom}(\Delta)\wedge\\
 & & \qquad\forall[\Xi'|\overline{\nu}'|\Delta']\sqsupseteq[\Xi|\overline{\nu}|\Delta].\,\mathcal{R}(\Xi')(\overline{\nu}')(\tau)(\Delta')\subset\mathcal{S}(\Xi')(\overline{\nu}')(\Delta'(l))(\Delta')\wedge\\
 & & \qquad\mathcal{R}(\Xi')(\overline{\nu}')(\Delta'(l))(\Delta')\subset\mathcal{S}(\Xi')(\overline{\nu}')(\tau)(\Delta')\big\}\\
\Psi(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\tau_0\times\tau_1)(\Delta) & = & \{(\bot,\bot)\}\cup\big\{(in_\times(\lfloor u_0,u_1\rfloor),in_\times(\lfloor u_0',u_1'\rfloor))\mid(u_0,u_0')\in\mathcal{S}(\Xi)(\overline{\nu})(\tau_0)(\Delta)\setminus\{(\bot,\bot)\}\wedge\\
 & & \qquad(u_1,u_1')\in\mathcal{S}(\Xi)(\overline{\nu})(\tau_1)(\Delta)\setminus\{(\bot,\bot)\}\big\}\\
\Psi(\mathcal{R},\mathcal{S})(\overline{\nu})(\Xi)(\tau_0+\tau_1)(\Delta) & = & \{(\bot,\bot)\}\cup\big\{((in_+\circ inl)(u),(in_+\circ inl)(u'))\mid(u,u')\in\mathcal{S}(\Xi)(\overline{\nu})(\tau_0)(\Delta)\setminus\{(\bot,\bot)\}\big\}\\
 & & \cup\big\{((in_+\circ inr)(u),(in_+\circ inr)(u'))\mid(u,u')\in\mathcal{S}(\Xi)(\overline{\nu})(\tau_1)(\Delta)\setminus\{(\bot,\bot)\}\big\}\\
\Psi(\mathcal{R},\mathcal{S})(\overline{\nu})(\Xi)(\mu\alpha.\tau)(\Delta) & = & \{(\bot,\bot)\}\cup\big\{(in_\mu(u),in_\mu(u'))\mid(u,u')\in\mathcal{S}(\Xi)(\overline{\nu})(\tau[\mu\alpha.\tau/\alpha])(\Delta)\setminus\{(\bot,\bot)\}\big\}\\
\Psi(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\forall\alpha.\tau)(\Delta) & = & \{(\bot,\bot)\}\cup\big\{(in_\forall(\varphi),in_\forall(\varphi'))\mid\varphi,\varphi'\in S\multimap S\otimes U\wedge\\
 & & \qquad\forall[\Xi'|\overline{\nu}'|\Delta']\sqsupseteq[\Xi|\overline{\nu}|\Delta]\forall\nu\in\mathbf{SCT}_{\Xi'}.\\
 & & \qquad(\varphi,\varphi')\in\Psi^T(\mathcal{R},\mathcal{S})(\Xi',\alpha)(\overline{\nu}',\nu)(\tau)(\Delta')\big\}\\
\Psi(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\tau_0\to\tau_1)(\Delta) & = & \{(\bot,\bot)\}\cup\\
 & & \big\{(in_\to(\varphi),in_\to(\varphi'))\mid\varphi,\varphi'\in S\otimes U\multimap S\otimes U\wedge\\
 & & \qquad\forall[\Xi'|\overline{\nu}'|\Delta']\sqsupseteq[\Xi|\overline{\nu}|\Delta]\forall(u,u')\in\mathcal{R}(\Xi')(\overline{\nu}')(\tau_0)(\Delta')\setminus\{(\bot,\bot)\}.\\
 & & \qquad[\lambda s\in S_\downarrow.\varphi(\lfloor s,u\rfloor),\lambda s'\in S_\downarrow.\varphi'(\lfloor s',u'\rfloor)]\in\Psi^T(\mathcal{R},\mathcal{S})(\Xi')(\overline{\nu}')(\tau_1)(\Delta')\big\}\\
\Psi^S(\mathcal{S})(\Xi)(\overline{\nu})(\Delta) & = & \{(\bot,\bot)\}\cup\big\{(s,s')\in(S_\downarrow)^2\mid\mathrm{dom}(\Delta)=\mathrm{dom}(s)=\mathrm{dom}(s')\wedge\\
 & & \qquad\forall l\in\mathrm{dom}(\Delta).\,(s(l),s'(l))\in\mathcal{S}(\Xi)(\overline{\nu})(\Delta(l))(\Delta)\setminus\{(\bot,\bot)\}\big\}\\
\Psi^K(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\tau)(\Delta) & = & \big\{(k,k')\in(S\otimes U\multimap S\otimes U)^2\mid\forall[\Xi'|\overline{\nu}'|\Delta']\sqsupseteq[\Xi|\overline{\nu}|\Delta].\\
 & & \qquad\forall(s,s')\in\Psi^S(\mathcal{R})(\Xi')(\overline{\nu}')(\Delta')\setminus\{(\bot,\bot)\}.\\
 & & \qquad\forall(u,u')\in\mathcal{R}(\Xi')(\overline{\nu}')(\tau)(\Delta')\setminus\{(\bot,\bot)\}.\\
 & & \qquad\quad[k(\lfloor s,u\rfloor)=\bot=k'(\lfloor s',u'\rfloor)]\vee\\
 & & \qquad\quad[\exists t,t'\in S_\downarrow\exists n\in\mathbb{Z}.\,k(\lfloor s,u\rfloor)=\lfloor t,in_{\mathtt{int}}(n)\rfloor\wedge\\
 & & \qquad\qquad k'(\lfloor s',u'\rfloor)=\lfloor t',in_{\mathtt{int}}(n)\rfloor\big\}\\
\Psi^T(\mathcal{R},\mathcal{S})(\Xi)(\overline{\nu})(\tau)(\Delta) & = & \big\{(\varphi,\varphi')\in(S\multimap S\otimes U)^2\mid\forall(s,s')\in\Psi^S(\mathcal{R})(\Xi)(\overline{\nu})(\Delta)\setminus\{(\bot,\bot)\}.\\
 & & \qquad\forall(k,k')\in\Psi^K(\mathcal{S},\mathcal{R})(\Xi)(\overline{\nu})(\tau)(\Delta).\\
 & & \qquad[k(\varphi(s))=\bot=k'(\varphi'(s'))]\vee\\
 & & \qquad[\exists t,t'\in S_\downarrow\exists n\in\mathbb{Z}.\,k(\varphi(s))=\lfloor t,in_{\mathtt{int}}(n)\rfloor\wedge\\
 & & \qquad\qquad k'(\varphi'(s'))=\lfloor t',in_{\mathtt{int}}(n)\rfloor\big\}
\end{array}
$$

Figure 7. Definition of $\Psi:\mathcal{L}^{op}\times\mathcal{L}\to\mathcal{L}$ using maps $\Psi^S:\mathcal{L}\to\prod_\Xi\mathbf{SCT}^\Xi\to\mathbf{World}_\Xi\to\mathcal{P}(S^2)$, $\Psi^K:\mathcal{L}^{op}\times\mathcal{L}\to\prod_\Xi\mathbf{SCT}^\Xi\to\mathbf{Type}_\Xi\to\mathbf{World}_\Xi\to\mathcal{P}((S\otimes U\multimap S\otimes U)^2)$ and $\Psi^T:\mathcal{L}^{op}\times\mathcal{L}\to\prod_\Xi\mathbf{SCT}^\Xi\to\mathbf{Type}_\Xi\to\mathbf{World}_\Xi\to\mathcal{P}((S\multimap S\otimes U)^2)$.

confer the definition of $[\![\tau]\!]^T_{\Xi'}(\overline{\nu}')(\Delta')$. Consider now the maps $k_0,k_0':S\otimes U\multimap S\otimes U$ built by copying the interpretation of lookup and applying k respectively k', i.e., k_0 is obtained by mapping any $(s_0,u_0)\in S_\downarrow\times U_\downarrow$ to $S\otimes U$ as follows

$$
\begin{cases}
k(\lfloor s_0,s_0(l)\rfloor) & u_0=\lambda_l, l\in\mathrm{dom}(s_0)\\
k(\lfloor s_0,\pi_n(s_0(l))\rfloor) & \pi_{n+1}(u_0)=\lambda_l^{n+1},\pi_n(s_0(l))\neq\bot,\\
 & \pi_{n+2}(u_0)\neq\lambda_l^{n+2}, l\in\mathrm{dom}(s_0)\\
\bot & \text{otherwise}
\end{cases}
$$

and k_0' is identical, with k' exchanged for k. By the induction hypothesis it suffices to prove that $(k_0,k_0')\in[\![\tau\ \mathtt{ref}]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. For that purpose we pick $[\Xi_0|\overline{\nu}_0|\Delta_0]\sqsupseteq[\Xi'|\overline{\nu}'|\Delta']$ and we pick $(s_0,s_0')\in[\![\Delta_0]\!]^S_{\Xi_0}(\overline{\nu}_0)$ and $(u_0,u_0')\in[\![\tau\ \mathtt{ref}]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. The latter yields one of two: Either we have $u_0=u_0'=\lambda_l^{n+1}$ for some $l\in\mathrm{dom}(\Delta_0)$ and an $n\in\mathbb{N}$ with $[\![\Delta_0(l)]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)\overset{n}{\subset}[\![\tau]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)\cup\{(\bot,\bot)\}$ or we have $u_0=u_0'=\lambda_l$ for some $l\in\mathrm{dom}(\Delta_0)$ with $[\![\Delta_0(l)]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)=[\![\tau]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. And in both cases the desired follows from the definitions of k_0 and k_0' and from $(s_0,s_0')\in[\![\Delta_0]\!]^S_{\Xi_0}(\overline{\nu}_0)$ and $(k,k')\in[\![\tau]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$.

Let us now look at the case of memory allocation, i.e., consider

$$\frac{\Xi\mid\Delta\mid\Gamma\vdash e:\tau}{\Xi\mid\Delta\mid\Gamma\vdash\mathtt{ref}(e):\tau\ \mathtt{ref}}$$

and assume that the proposition holds for the premise. We proceed as above, i.e., we pick arbitrary $\Xi'\supset\Xi$, $\overline{\nu}'\in\mathbf{SCT}^{\Xi'}$, $\Delta'\in\mathbf{World}_{\Xi'}$ with $\Delta'\sqsupseteq\Delta$, and $\rho,\rho':\mathrm{dom}(\Gamma)\to U_\downarrow$ as specified in the definition of semantic relatedness. Also we take arbitrary $(s,s')\in[\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k,k')\in[\![\tau\ \mathtt{ref}]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$ and we construct $k_0,k_0':S\otimes U\multimap S\otimes U$ by copying the interpretation of allocation and applying k respectively k', i.e., k_0 is built from the map

$$\lambda(s_0,u_0)\in S_\downarrow\times U_\downarrow.\,k(\lfloor s_0[l\mapsto u_0],\lambda_l\rfloor)$$

$$(u,u') \in [\![\alpha_{\mathtt{n}}]\!]_{\alpha_1,\ldots,\alpha_m}(\nu_1,\ldots,\nu_m)(\Delta) \Leftrightarrow (u,u') \in \nu_n\big[\lambda\tau \in \mathbf{Type}_{\alpha_1,\ldots,\alpha_{n-1}}.[\![\tau]\!]_{\alpha_1,\ldots,\alpha_m}(\nu_1,\ldots,\nu_m)(\Delta) \cup \{(\bot,\bot)\}\big] \setminus \{(\bot,\bot)\}$$

$$\begin{array}{rcl}
(u,u') \in [\![\mathtt{1}]\!]_\Xi(\overline{\nu})(\Delta) & \iff & u = u' = in_{\mathtt{unit}}(*) \\
(u,u') \in [\![\mathtt{int}]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \exists n \in \mathbb{Z}.\ u = u' = in_{\mathtt{int}}(n) \\
(u,u') \in [\![\tau\ \mathtt{ref}]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \big[\exists l \in \mathrm{dom}(\Delta) \exists n \in \mathbb{N}.\ u = u' = \lambda_l^{n+1} \wedge \\
 & & \quad \forall[\Xi'|\overline{\nu}'|\Delta'] \sqsupseteq [\Xi|\overline{\nu}|\Delta].\ [\![\tau]\!]_{\Xi'}(\overline{\nu}')(\Delta') \cup \{(\bot,\bot)\} \stackrel{n}{=} [\![\Delta'(l)]\!]_{\Xi'}(\overline{\nu}')(\Delta') \cup \{(\bot,\bot)\}\big] \vee \\
 & & \big[\exists l \in \mathrm{dom}(\Delta).\ u = u' = \lambda_l \wedge \\
 & & \quad \forall[\Xi'|\overline{\nu}'|\Delta'] \sqsupseteq [\Xi|\overline{\nu}|\Delta].\ [\![\tau]\!]_{\Xi'}(\overline{\nu}')(\Delta') = [\![\Delta'(l)]\!]_{\Xi'}(\overline{\nu}')(\Delta')\big] \\
(u,u') \in [\![\tau_0 \times \tau_1]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \exists (u_0,u_0') \in [\![\tau_0]\!]_\Xi(\overline{\nu})(\Delta) \exists (u_1,u_1') \in [\![\tau_1]\!]_\Xi(\overline{\nu})(\Delta).\ u = in_\times(\lfloor u_0,u_1\rfloor) \wedge u' = in_\times(\lfloor u_0',u_1'\rfloor) \\
(u,u') \in [\![\tau_0 + \tau_1]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \big[\exists (u_0,u_0') \in [\![\tau_0]\!]_\Xi(\overline{\nu})(\Delta).\ u = (in_+ \circ inl)(u_0) \wedge u' = (in_+ \circ inl)(u_0')\big] \vee \\
 & & \big[\exists (u_1,u_1') \in [\![\tau_1]\!]_\Xi(\overline{\nu})(\Delta).\ u = (in_+ \circ inr)(u_1) \wedge u' = (in_+ \circ inr)(u_1')\big] \\
(u,u') \in [\![\mu\alpha.\tau]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \exists (u_0,u_0') \in [\![\tau[\mu\alpha.\tau/\alpha]]\!]_\Xi(\overline{\nu})(\Delta).\ u = in_\mu(u_0) \wedge u' = in_\mu(u_0') \\
(u,u') \in [\![\forall\alpha.\tau]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \exists \varphi,\varphi' \in S \multimap S \otimes U.\ u = in_\forall(\varphi) \wedge u' = in_\forall(\varphi') \wedge \\
 & & \quad \forall[\Xi'|\overline{\nu}'|\Delta'] \sqsupseteq [\Xi|\overline{\nu}|\Delta] \forall \nu \in \mathbf{SCT}_{\Xi'}.\ (\varphi,\varphi') \in [\![\tau]\!]^T_{\Xi',\alpha}(\overline{\nu}',\nu)(\Delta') \\
(u,u') \in [\![\tau_0 \to \tau_1]\!]_\Xi(\overline{\nu})(\Delta) & \iff & \exists \varphi,\varphi' \in S \otimes U \multimap S \otimes U.\ u = in_\to(\varphi) \wedge u' = in_\to(\varphi') \wedge \\
 & & \quad \forall[\Xi'|\overline{\nu}'|\Delta'] \sqsupseteq [\Xi|\overline{\nu}|\Delta] \forall (u_0,u_0') \in [\![\tau_0]\!]_{\Xi'}(\overline{\nu}')(\Delta').\ \\
 & & \quad\quad [\lambda s \in S_\downarrow.\varphi(\lfloor s,u_0\rfloor), \lambda s' \in S_\downarrow.\varphi'(\lfloor s',u_0'\rfloor)] \in [\![\tau_1]\!]^T_{\Xi'}(\overline{\nu}')(\Delta') \\
 & & \\
(s,s') \in [\![\Delta]\!]^S_\Xi(\overline{\nu}) & \iff & \mathrm{dom}(\Delta) = \mathrm{dom}(s) = \mathrm{dom}(s') \wedge \forall l \in \mathrm{dom}(\Delta).\ (s(l),s'(l)) \in [\![\Delta(l)]\!]_\Xi(\overline{\nu})(\Delta) \\
(k,k') \in [\![\tau]\!]^K_\Xi(\overline{\nu})(\Delta) & \iff & \forall[\Xi'|\overline{\nu}'|\Delta'] \sqsupseteq [\Xi|\overline{\nu}|\Delta] \forall (s,s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}') \forall (u,u') \in [\![\tau]\!]_{\Xi'}(\overline{\nu}')(\Delta').\ \\
 & & \quad \big[k(\lfloor s,u\rfloor) = \bot = k'(\lfloor s',u'\rfloor)\big] \vee \\
 & & \quad \big[\exists t,t' \in S_\downarrow \exists n \in \mathbb{Z}.\ k(\lfloor s,u\rfloor) = \lfloor t, in_{\mathtt{int}}(n)\rfloor \wedge k'(\lfloor s',u'\rfloor) = \lfloor t', in_{\mathtt{int}}(n)\rfloor\big] \\
(\varphi,\varphi') \in [\![\tau]\!]^T_\Xi(\overline{\nu})(\Delta) & \iff & \forall (s,s') \in [\![\Delta]\!]^S_\Xi(\overline{\nu})\, \forall (k,k') \in [\![\tau]\!]^K_\Xi(\overline{\nu})(\Delta).\ \\
 & & \quad \big[k(\varphi(s)) = \bot = k'(\varphi'(s'))\big] \vee \\
 & & \quad \big[\exists t,t' \in S_\downarrow \exists n \in \mathbb{Z}.\ k(\varphi(s)) = \lfloor t, in_{\mathtt{int}}(n)\rfloor \wedge k'(\varphi'(s')) = \lfloor t', in_{\mathtt{int}}(n)\rfloor\big]
\end{array}$$

Figure 8. Desired properties of interpretation of types. For $u,u' \in U_\downarrow$, a context Ξ, $\tau \in \mathbf{Type}_\Xi$, $\overline{\nu} \in \mathbf{SCT}^\Xi$ and $\Delta \in \mathbf{World}_\Xi$ we specify when $(u,u') \in [\![\tau]\!]_\Xi(\overline{\nu})(\Delta)$. Also we define $[\![\Delta]\!]^S_\Xi(\overline{\nu}) \subset (S_\downarrow)^2$, $[\![\tau]\!]^K_\Xi(\overline{\nu})(\Delta) \subset (S \otimes U \multimap S \otimes U)^2$, and $[\![\tau]\!]^T_\Xi(\overline{\nu})(\Delta) \subset (S \multimap S \otimes U)^2$.

where we choose $l \in \mathbb{N}$ with $l \notin \mathrm{dom}(s_0)$ and $\forall l' < l.\ l' \in \mathrm{dom}(s_0)$ and k_0' is identical, with k' exchanged for k. It now remains to prove $(k_0,k_0') \in [\![\tau]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. For that purpose we pick $[\Xi_0|\overline{\nu}_0|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$ and we pick $(s_0,s_0') \in [\![\Delta_0]\!]^S_{\Xi_0}(\overline{\nu}_0)$ and $(u_0,u_0') \in [\![\tau]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. From the former of these we get $k_0(\lfloor s_0,u_0\rfloor) = k(\lfloor s_0[l \mapsto u_0], \lambda_l\rfloor)$ and $k_0'(\lfloor s_0',u_0'\rfloor) = k'(\lfloor s_0'[l \mapsto u_0'], \lambda_l\rfloor)$ with l the least such that $l \notin \mathrm{dom}(\Delta_0)$. It is immediate that $(\lambda_l,\lambda_l) \in [\![\tau\ \mathtt{ref}]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0[l \mapsto \tau])$ and for any $l' \in \mathrm{dom}(\Delta_0[l \mapsto \tau])$ we have $(s_0[l \mapsto u_0](l'), s_0'[l \mapsto u_0'](l')) \in [\![\Delta_0[l \mapsto \tau](l')]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$ and hence $(s_0[l \mapsto u_0], s_0'[l \mapsto u_0']) \in [\![\Delta_0[l' \mapsto \tau]]\!]_{\Xi_0}(\overline{\nu}_0)$ by the Monotonicity Lemma. And applying the original assumption $(k,k') \in [\![\tau\ \mathtt{ref}]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$ we are done.

Finally we arrive at the the case of type application, this is where we require the Degenerate Substitution Lemma. We consider

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \forall\alpha.\tau_0}{\Xi \mid \Delta \mid \Gamma \vdash e[\tau_1] : \tau_0[\tau_1/\alpha]}\,(\Xi \vdash \tau_1)$$

and assume that the proposition holds for the premise. We proceed as usual, pick arbitrary $\Xi' \supset \Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta' \in \mathbf{World}_{\Xi'}$ with $\Delta' \sqsupseteq \Delta$ and $\rho,\rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ as specified in the definition of semantic relatedness. Also we take arbitrary $(s,s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k,k') \in [\![\tau_0[\tau_1/\alpha]]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$ and we construct $k_0,k_0' : S \otimes U \multimap S \otimes U$ by copying the interpretation of type application and applying k respectively k', i.e., k_0 is built from the map

$$\lambda(s,u) \in S_\downarrow \times U_\downarrow.\ \begin{cases} k(\varphi(s)) & u = in_\forall(\varphi) \\ \bot & \text{otherwise} \end{cases}$$

and k_0' is identical, with k' exchanged for k. It now remains to prove $(k_0,k_0') \in [\![\forall\alpha.\tau_0]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. For that purpose we pick $[\Xi_0|\overline{\nu}_0|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$ and we pick $(s_0,s_0') \in [\![\Delta_0]\!]^S_{\Xi_0}(\overline{\nu}_0)$ and $(u_0,u_0') \in [\![\forall\alpha.\tau_0]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. From the latter we get $\varphi_0,\varphi_0' \in S \multimap S \otimes U$ such that $u_0 = in_\forall(\varphi_0)$, $u_0' = in_\forall(\varphi_0')$ and $(\varphi,\varphi') \in [\![\tau_0]\!]^T_{\Xi_0,\alpha}(\overline{\nu}_0, \nu_{\Xi_0}(\tau_1))(\Delta_0)$, now it remains to show that we have

$$(s_0,s_0') \in [\![\Delta_0]\!]^S_{\Xi_0,\alpha}(\overline{\nu}_0, \nu_{\Xi_0}(\tau_1))$$

and that we have

$$(k,k') \in [\![\tau_0]\!]^K_{\Xi_0,\alpha}(\overline{\nu}_0, \nu_{\Xi_0}(\tau_1))(\Delta_0).$$

The first is an easy consequence of the Monotonicity lemma, for the latter we use the Degenerate Substitution Lemma to conclude

$$[\![\tau_0]\!]^K_{\Xi_0,\alpha}(\overline{\nu}_0, \nu_{\Xi_0}(\tau_1))(\Delta_0) = [\![\tau_0[\tau_1/\alpha]]\!]^K_{\Xi_0,\alpha}(\overline{\nu}_0, \nu_{\Xi_0}(\tau_1))(\Delta_0),$$

this suffices as $[\Xi_0,\alpha|(\overline{\nu}_0,\nu_{\Xi_0}(\sigma))|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$. □

Corollary 18. *Semantically related expressions in context are contextually equivalent: if* $\Xi \mid \emptyset \mid \Gamma \vdash e_1 \sim e_2 : \tau$ *then* $\Xi \mid \emptyset \mid \Gamma \vdash e_1 =_{\mathrm{ctx}} e_2 : \tau$.

Proof. This follows in the standard manner from the proof of the fundamental theorem (Theorem 17) above together with the adequacy and soundness results from the previous section. □

The theorem above, and its corollary, forms the basis for simple reasoning about parametricity using our model. A few examples are shown in Section 5.

4.1 Alternative Approach

In this subsection we briefly discuss and sketch an alternative approach to the second issue: the interpretation of open types depending on worlds as mentioned in the introduction of Section 4.

Here, we interpret quantified types as intersections over semantic closed types, the latter are members of $\mathbf{UARel}(U)$ parameterized over interpretations of types with fewer type variables. This somewhat syntactic choice goes nicely with syntactic worlds containing free type variables. The alternative approach is to have semantic worlds, mapping locations to semantic types and letting semantic types be world-indexed members of $\mathbf{UARel}(U)$. This introduces a mutual dependency between worlds and semantic types; in effect, we ask for solutions to the mutually recursive equations (recall that locations are natural numbers):

$$\begin{array}{rcl} \mathrm{ST} & = & W \to \mathbf{UARel}(U) \\ W & = & \mathbb{N} \stackrel{fin}{\to} \mathrm{ST} \end{array}$$

It turns out that one can solve equations similar to the above in suitable categories of complete ultra-metric spaces. Our solution relies on well-known metrics associated with partial equivalence relations [1, 4]. The alternative approach gives a more semantic understanding of open types, in particular one can interpret quantified types $\forall\alpha.\tau$ by the more standard $\bigcap_{\nu\in\mathrm{ST}}[\![\tau]\!](\nu)$. But it does come at the price of using (yet) more mathematical machinery. The present approach is a fairly (if not entirely) simple alternative. And while the two approaches yield different models, it is not immediate that either is superior in terms of proving more equivalences.

For lack of space we cannot present the details of the alternative approach here; that will be done in a forthcoming paper. With this approach we will also be able to make a more detailed comparison to the step-indexed approach to recursive types and references [2, 6]; indeed, approximations to equations similar to those shown above play a key role in recent step-indexed models [6].

5. Examples of Parametricity Reasoning

As explained in the Introduction, this paper focuses on the key technical challenges involved in defining an adequate, parametric model for a language with recursive types and general references. The main contributions of the paper are our solutions to these challenges, including the concepts of *semantic locations* and *semantic closed types*; extending the current setup to allow for more advanced applications involving local state [11] is deferred to future work (see Section 6).

As illustrated by the first example below, one can use the parametricity results in this paper to prove equivalences between different functional implementations of abstract data types in an imperative language. The proof essentially proceeds in the standard manner — but the point now is that the clients of such abstract data types may be implemented using *all* the features of the language, including general references, recursive types, etc. The remaining three examples below illustrate that one can prove simple equivalences involving imperative abstract data types and local state.

In the examples we use the standard encoding of n-ary products by means of binary products. And we refer to the type `unit` by 1.

Example 19. In the first example, we show that a client of a module that implements a counter cannot distinguish between two different, but related implementations of the module. The two implementations are very simple functional implementations, but we emphasize that the reasoning works for *any* client of the right type; the client may be implemented using all the features of the language.

The type of counter-module clients is

$$\tau_{\mathrm{cl}} = \forall\alpha.((1 \to \alpha) \times (\alpha \to \alpha) \times (\alpha \to \mathtt{int}) \to \mathtt{int})\,.$$

Intuitively, a client c of a counter module takes an unknown type α (the concrete type used internally by the module to represent counters) and three functions (the first for creating a new counter, the second for incrementing a counter, and the third for getting the value of a counter) and returns a result of type `int`.

Let the two counter implementations be given by I_1 and I_2:

$$\begin{array}{rcl} I_1 & = & (\lambda x:1.\,0,\ \lambda x:\mathtt{int}.\,x+1,\ \lambda x:\mathtt{int}.\,x) \\ I_2 & = & (\lambda x:1.\,0,\ \lambda x:\mathtt{int}.\,x-1,\ \lambda x:\mathtt{int}.\,-x). \end{array}$$

We can now use Corollary 18 to prove that

$$\emptyset \mid \emptyset \mid c : \tau_{\mathrm{cl}} \vdash c[\mathtt{int}]I_1 =_{\mathrm{ctx}} c[\mathtt{int}]I_2 : \mathtt{int}\,.$$

The proof of relatedness of $c[\mathtt{int}]I_1$ and $c[\mathtt{int}]I_2$ proceeds as expected, except that it is in continuation-passing style, and, of course, involves the definition of a relation relating each integer n to $-n$. Formally, one uses the semantic closed type $\nu_0 \in \mathbf{SCT}_\Xi$ defined by

$$\nu_0(\varphi) = \{(\bot,\bot)\} \cup \{(in_{\mathtt{int}}(n), in_{\mathtt{int}}(-n)) \mid n \in \mathbb{N}\}\,.$$

Example 20. Consider now the following type of clients of an *imperative* counter module:

$$\tau'_{\mathrm{cl}} = \forall\alpha.((1 \to \alpha) \times (\alpha \to 1) \times (\alpha \to \mathtt{int}) \to \mathtt{int})\,.$$

As in the previous example, the intuition is that a client takes an unknown type α and three functions implementing operations on counters. The difference from the previous example is that the second of the three functions has the type $\alpha \to 1$, reflecting that the 'increment' operation modifies its input and does not need to return a result.

Let the two imperative implementations be given by I'_1 and I'_2:

$$\begin{array}{rcl} I'_1 & = & (\lambda x:1.\,\mathtt{ref}(0), \\ & & \ \lambda x:\mathtt{int\ ref}.\,x \ :=\ !x+1, \\ & & \ \lambda x:\mathtt{int\ ref}.\,!x) \\ I'_2 & = & (\lambda x:1.\,\mathtt{ref}(0), \\ & & \ \lambda x:\mathtt{int\ ref}.\,x \ :=\ !x-1, \\ & & \ \lambda x:\mathtt{int\ ref}.\,-(!x)) \end{array}$$

We can now use Corollary 18 to prove that

$$\emptyset \mid \emptyset \mid c : \tau'_{\mathrm{cl}} \vdash c[\mathtt{int\ ref}]I'_1 =_{\mathrm{ctx}} c[\mathtt{int\ ref}]I'_2 : \mathtt{int}\,.$$

To show semantic relatedness, we let $\Delta \in \mathbf{World}_\Xi$ and $\overline{\nu} \in \mathbf{SCT}^\Xi$ and $(c_1,c_2) \in [\![\tau'_{\mathrm{cl}}]\!]_\Xi(\overline{\nu})(\Delta)$ for some arbitrary Ξ. We now exploit the fact that 'future worlds' may contain arbitrary new type variables. Pick $\alpha_0 \notin \Xi$; it suffices to show that

$$\begin{array}{l} ([\![I'_1]\!],[\![I'_2]\!]) \in \\ \quad [\![(1 \to \alpha) \times (\alpha \to 1) \times (\alpha \to \mathtt{int})]\!]_{\Xi,\alpha_0,\alpha}(\overline{\nu},\nu_0,\nu)(\emptyset)\,, \end{array}$$

where ν_0 is defined as in the previous example, and where $\nu = \nu_{(\Xi,\alpha_0)}(\alpha_0\,\mathtt{ref})$ is the semantic closed type corresponding to the syntactic type $\alpha_0\,\mathtt{ref}$.

From here, the most interesting part of the proof is the relatedness of the two implementations of the operation for creating a new counter. The core of the proof obligation is the following: given $[\Xi'|\overline{\nu}'|\Delta'] \sqsupseteq [(\Xi,\alpha_0,\alpha)|(\overline{\nu},\nu_0,\nu)|\emptyset]$, states $(s,s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$, and continuations $(k,k') \in [\![\alpha]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$, we must show that $k\,[\![\mathtt{ref}\ 0]\!]^s_\emptyset$ and $k'\,[\![\mathtt{ref}\ 0]\!]^{s'}_\emptyset$ are both $\bot$ or contain the same integer component. But the characterization of $[\![\alpha]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$ in Figure 8 involves a quantification over all $\Delta'' \sqsupseteq \Delta'$: we can exploit that quantification by choosing $\Delta'' = \Delta'[l \mapsto \alpha_0]$ where l is the smallest number not in the domain of Δ'. The result easily follows.

Example 21. As in Crary and Harper [13], we can introduce the usual encoding of existential types by means of universal types:

$$\exists\alpha.\tau = \forall\beta.(\forall\alpha.\,\tau \to \beta) \to \beta\,.$$

We then revisit the previous example: the type

$$\tau_{\mathrm{m}} = \exists\alpha.\,(1 \to \alpha) \times (\alpha \to 1) \times (\alpha \to \mathtt{int})$$

can be used to model imperative counter modules.

Consider the following two module implementations, i.e., closed terms of type τ_{m}:

$$J_1 = \Lambda\beta.\lambda c.\, c[\mathtt{int\,ref}]I_1' \quad \text{and} \quad J_2 = \Lambda\beta.\lambda c.\, c[\mathtt{int\,ref}]I_2'$$

(where I_1' and I_2' are defined in the previous example). We can use Corollary 18 to prove that J_1 and J_2 are contextually equivalent. The reasoning is essentially as in the previous example, except that the 'answer type' is now a universally quantified type variable β instead of the fixed type `int`.

Example 22. One can alternatively implement an imperative counter module by means of a local reference and two closures. Consider the type $\tau_{\mathrm{lr}} = 1 \to ((1 \to 1) \times (1 \to \mathtt{int}))$ and the two counter implementations

$$\begin{aligned} J &= \lambda x : 1.\,\mathtt{let}\, r \;=\; \mathtt{ref}\, 0\, \mathtt{in}\; (\lambda y : 1.\, r \;:=\; !r + 1,\;\; \lambda y : 1.\, !r) \\ J' &= \lambda x : 1.\,\mathtt{let}\, r \;=\; \mathtt{ref}\, 0 \\ &\qquad\quad \mathtt{in}\; (\lambda y : 1.\, r \;:=\; !r - 1,\;\; \lambda y : 1.\; -\,(!r)) \end{aligned}$$

where the `let ... in` construct is syntactic sugar for a β-redex in the usual way. Both J and J' are closed terms of type τ_{lr}, and we can use Corollary 18 to show that the two terms are contextually equivalent. As in Example 20, the proof involves introducing a new type variable α_0, interpreted by ν_0.

6. Conclusion and Future Work

We have given a first relationally parametric possible world semantics for a call-by-value higher-order language with impredicative polymorphism, general references, and recursive types. In particular, we have discovered a technical challenge in establishing the existence of the requisite relational interpretations of types and solved the problem of existence by a novel model of references using a semantic notion of location that permits a useful approximation relation. We are convinced that the technical challenge is a real one and think that the reason it has not been observed before when modelling references with domains is that it only shows up when one insists on modeling *open* types (as needed for parametricity).

As already mentioned, the logical relations suffice for proving parametricity results for a language with recursive types and general references. They are, however, not tailored for maximal 'proof strength', rather the focus is on the underlying semantic challenges. In particular, reasoning about local state is not in general possible, we may, e.g., not prove 'garbage collection' of unused references. We plan to extend and combine the present work with earlier work on reasoning about local state [11] — this allows for formal proofs that two implementations of an abstract type using local state in different ways are related. Indeed in [12], the first author and Nina Bohr extended the techniques in [11] to a language with impredicative polymorphism and references to *closed* types (closed to avoid the technical challenges addressed in this paper), and were, e.g., able to prove two implementations of an abstract stack type related, one implementation using an ML-style list and the other using a linked list implementation for the stack [12, Sec. 5].

Finally, recent work [3] by Ahmed, Dreyer and Rossberg came to our attention after writing this paper. They too provide a relationally parametric possible world semantics of a similar language, but using a step-indexed approach rather than a domain theoretic. Also their worlds are more flexible and hence applicable to more examples. Indeed, their work extend ideas from the aforementioned work [11] but does so in a step-indexed fashion.

References

[1] M. Abadi and G. Plotkin. A per model of polymorphism and recursive types. In *Proceedings of LICS*, pages 355–365, 1990.

[2] A. Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In *Proc. of ESOP*, pages 69–83, 2006.

[3] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. To appear at POPL 2009.

[4] R. M. Amadio. Recursion over realizability structures. *Information and Computation*, 91(1):55–85, 1991.

[5] A. W. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *TOPLAS*, 23(5):657–683, 2001.

[6] A. W. Appel, P.-A. Melliès, C. D. Richards, and J. Vouillon. A very modal model of a modern, major, general type system. In *Proc. of POPL*, pages 109–122, 2007.

[7] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *Proc. of TLCA*, volume 3461 of *LNCS*, 2005.

[8] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In *Proc. of HOOTS*, volume 41 of *ENTCS*, 2000.

[9] L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*, 15(4):709–772, 2005.

[10] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Linear Abadi & Plotkin logic. *Logical Methods in Computer Science*, 2(5:1):1–33, 2006.

[11] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Proc. of APLAS*, pages 79–96, 2006.

[12] N. Bohr and L. Birkedal. Relational parametricity for recursive types and references of closed types. Technical report, IT University of Copenhagen, 2007. A Chapter in Nina Bohr's Ph.D. dissertation 2007.

[13] K. Crary and R. Harper. Syntactic logical relations for polymorphic and recursive types. *ENTCS*, 172:259–299, 2007.

[14] A. Filinski. On the relations between monadic semantics. *Theoretical Computer Science*, 375(1–3):41–75, 2007.

[15] M. Hasegawa. Relational parametricity and control. *Logical Methods in Computer Science*, 2(3):1–22, 2006.

[16] P. Johann. On proving the correctness of program transformations based on free theorems for higher-order polymorphic calculi. *Mathematical Structures in Computer Science*, 10(2):201–229, 2005.

[17] P. Johann and J. Voigtlaender. The impact of seq on free theorems-based program transformations. *Fundamenta Informaticae*, 69(1–2):63–102, 2006.

[18] V. Koutavas and M. Wand. Bisimulations for untyped imperative objects. In *Proc. of ESOP*, pages 146–161, 2006.

[19] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *Proc. of POPL*, pages 141–152, 2006.

[20] S. B. Lassen and P. B. Levy. Normal form bisimulation for parametric polymorphism. To appear at LICS 2008.

[21] S. B. Lassen and P. B. Levy. Typed normal form bisimulation. In *Proc. of CSL*, volume 4646 of *LNCS*, pages 283–297, 2007.

[22] P. Levy. Possible world semantics for general storage in call-by-value. In *Proc. of CSL*, volume 2471 of *LNCS*, pages 232–246, 2002.

[23] P.-A. Melliès and J. Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *Proc. of LICS*, pages

82–91, 2005.

[24] R. Møgelberg. Interpreting polymorphic FPC into domain theoretic models of parametric polymorphism. In *Proc. of ICALP*, pages 372–383, 2006.

[25] R. Møgelberg and A. Simpson. Relational parametricity for computational effects. In *Proc. of LICS*, pages 346–355, 2007.

[26] R. Møgelberg and A. Simpson. Relational parametricity for control considered as a computational effect. In *Proc. of MFPS*, ENTCS, pages 295–312, 2007.

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

[28] A. M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.

[29] A. M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in computer Science*, 10:321–359, 2000.

[30] A. M. Pitts. *Advanced Topics in Types and Programming Languages*, chapter Typed Operational Reasoning. The MIT Press, 2005.

[31] A. M. Pitts and I. Stark. Observable properties of higher order functions that dynamically create local names, or: What's new? In *Proceedings of MFPS*, volume 711 of *LNCS*, pages 122–141, 1993.

[32] G. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, Feb. 1993.

[33] G. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Proc. of TLCA*, volume 664 of *LNCS*, pages 361–375, 1993.

[34] J. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983.

[35] K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In *Proc. of POPL*, pages 161–172, 2007.

[36] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. In *Proc. of POPL*, pages 63–74, 2005.

[37] P. Wadler. Theorems for free! In *4th Symposium on Functional Programming Languages and Computer Architecture*, pages 347–359, 1989.

A. Typing Rules

$$\frac{}{\Xi \mid \Delta \mid \Gamma \vdash x : \tau}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma, x \in \mathrm{dom}(\Gamma), \Gamma(x) = \tau)$$

$$\frac{}{\Xi \mid \Delta \mid \Gamma \vdash () : \mathtt{unit}}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma)$$

$$\frac{}{\Xi \mid \Delta \mid \Gamma \vdash n : \mathtt{int}}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma)$$

$$\frac{}{\Xi \mid \Delta \mid \Gamma \vdash l : \tau\ \mathtt{ref}}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma, l \in \mathrm{dom}(\Delta), \Delta(l) = \tau)$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \mathtt{int} \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \mathtt{int}}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{op}(e_0 \pm e_1) : \mathtt{int}}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \mathtt{int} \qquad \Xi \mid \Delta \mid \Gamma \vdash e_i : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{ifzero}\ e\ \mathtt{then}\ e_0\ \mathtt{else}\ e_1 : \tau}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau_0 \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash (e_0, e_1) : \tau_0 \times \tau_1}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 \times \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fst}(e) : \tau_0} \qquad \frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 \times \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{snd}(e) : \tau_1}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{inl}^{\tau_0+\tau_1}(e) : \tau_0 + \tau_1}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{inr}^{\tau_0+\tau_1}(e) : \tau_0 + \tau_1}\ (\Xi \vdash \tau_0)$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 + \tau_1 \qquad \Xi \mid \Delta \mid \Gamma, x : \tau_i \vdash e_i : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(x).\, e_0\ \mathtt{else}\ \mathtt{inr}(x).\, e_1 : \tau}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau[\mu\alpha.\tau/\alpha]}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fold}^{\mu\alpha.\tau}(e) : \mu\alpha.\tau}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \mu\alpha.\tau}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{unfold}^{\mu\alpha.\tau}(e) : \tau[\mu\alpha.\tau/\alpha]}$$

$$\frac{\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau}\ (\Xi \vdash \Delta, \Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \forall\alpha.\tau_0}{\Xi \mid \Delta \mid \Gamma \vdash e[\tau_1] : \tau_0[\tau_1/\alpha]}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Delta \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash e : \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fix}^{\tau_0 \to \tau_1} f(x).e : \tau_0 \to \tau_1}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau_0 \to \tau_1 \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau_0}{\Xi \mid \Delta \mid \Gamma \vdash e_0(e_1) : \tau_1}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{ref}(e) : \tau\ \mathtt{ref}} \qquad \frac{\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \mathtt{ref}}{\Xi \mid \Delta \mid \Gamma \vdash !e : \tau}$$

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \mathtt{ref} \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau}{\Xi \mid \Delta \mid \Gamma \vdash e_0 := e_1 : \mathtt{unit}}$$

Figure 9. Typing rules. The general form is $\Xi \mid \Delta \mid \Gamma \vdash e : \tau$ for a context of type variables Ξ, a world Δ, a context of term variables Γ, an expression e and a type τ.

B. Big-Step Operational Semantics

$$\frac{}{\Pi, () \Downarrow \Pi, ()} \qquad \frac{}{\Pi, n \Downarrow \Pi, n} \qquad \frac{}{\Pi, l \Downarrow \Pi, l}$$

$$\frac{}{\Pi, \Lambda\alpha.e \Downarrow \Pi, \Lambda\alpha.e}$$

$$\frac{}{\Pi, \mathtt{fix}^{\tau_0 \to \tau_1} f(x).e \Downarrow \Pi, \mathtt{fix}^{\tau_0 \to \tau_1} f(x).e}$$

$$\frac{\Pi, e_0 \Downarrow \Pi', n_0 \qquad \Pi', e_1 \Downarrow \Pi'', n_1}{\Pi, \mathtt{op}(e_0 \pm e_1) \Downarrow \Pi'', n_0 \pm n_1}$$

$$\frac{\Pi, e \Downarrow \Pi', 0 \qquad \Pi', e_0 \Downarrow \Pi'', v}{\Pi, \mathtt{ifzero}\ e\ \mathtt{then}\ e_0\ \mathtt{else}\ e_1\ \Downarrow \Pi'', v'}$$

$$\frac{\Pi, e \Downarrow \Pi', n \qquad \Pi', e_1 \Downarrow \Pi'', v}{\Pi, \mathtt{ifzero}\ e\ \mathtt{then}\ e_0\ \mathtt{else}\ e_1\ \Downarrow \Pi'', v'}\ (n \neq 0)$$

$$\frac{\Pi, e_0 \Downarrow \Pi', v_0 \qquad \Pi', e_1 \Downarrow \Pi'', v_1}{\Pi, (e_0, e_1) \Downarrow \Pi'', (v_0, v_1)}$$

$$\frac{\Pi, e \Downarrow \Pi', (v_0, v_1)}{\Pi, \mathtt{fst}(e) \Downarrow \Pi', v_0} \qquad \frac{\Pi, e \Downarrow \Pi', (v_0, v_1)}{\Pi, \mathtt{snd}(e) \Downarrow \Pi', v_1}$$

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \mathtt{inl}^{\tau_0+\tau_1}(e) \Downarrow \Pi', \mathtt{inl}^{\tau_0+\tau_1}(v)}$$

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \mathtt{inr}^{\tau_0+\tau_1}(e) \Downarrow \Pi', \mathtt{inr}^{\tau_0+\tau_1}(v)}$$

$$\frac{\Pi, e \Downarrow \Pi', \mathtt{inl}(v) \qquad \Pi', e_0[v/x] \Downarrow \Pi'', v'}{\Pi, \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(x).\ e_0\ \mathtt{else}\ \mathtt{inr}(x).\ e_1\ \Downarrow \Pi'', v'}$$

$$\frac{\Pi, e \Downarrow \Pi', \mathtt{inr}(v) \qquad \Pi', e_1[v/x] \Downarrow \Pi'', v'}{\Pi, \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(x).\ e_0\ \mathtt{else}\ \mathtt{inr}(x).\ e_1\ \Downarrow \Pi'', v'}$$

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \mathtt{fold}^{\mu\alpha.\tau}(e) \Downarrow \Pi', \mathtt{fold}^{\mu\alpha.\tau}(v)}$$

$$\frac{\Pi, e \Downarrow \Pi', \mathtt{fold}^{\mu\alpha.\tau}(v)}{\Pi, \mathtt{unfold}^{\mu\alpha.\tau}(e) \Downarrow \Pi', v}$$

$$\frac{\Pi, e \Downarrow \Pi', \Lambda\alpha.e' \qquad \Pi', e'[\tau/\alpha] \Downarrow \Pi'', v}{\Pi, e[\tau] \Downarrow \Pi'', v}$$

$$\frac{\begin{array}{c}\Pi, e_0 \Downarrow \Pi', \mathtt{fix}^{\tau_0 \to \tau_1}\ f(x).e \quad \Pi', e_1 \Downarrow \Pi'', v \\ \Pi'', e[v/x, \mathtt{fix}^{\tau_0 \to \tau_1}\ f(x).e/f] \Downarrow \Pi''', v'\end{array}}{\Pi, e_0(e_1) \Downarrow \Pi''', v'}$$

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \mathtt{ref}(e) \Downarrow \Pi'[l \mapsto v], l}\ (l \notin \mathrm{dom}(\Pi'))$$

$$\frac{\Pi, e \Downarrow \Pi', l}{\Pi, !\,e \Downarrow \Pi', v}\ (l \in \mathrm{dom}(\Pi'), \Pi'(l) = v)$$

$$\frac{\Pi, e_0 \Downarrow \Pi', l \qquad \Pi', e_1 \Downarrow \Pi'', v}{\Pi, e_0 := e_1 \Downarrow \Pi''[l \mapsto v], ()}\ (l \in \mathrm{dom}(\Pi''))$$

Figure 10. Big-step operational semantics. The general form is $\Pi, e \Downarrow \Pi', v$ where Π and Π' are syntactic stores, e is an expression and v a value.

C. Proof of Soundness

The following definition captures equality up to location renaming:

Definition 23. *For syntactic stores* Π, Π', *expressions* e, e' *and a bijection* $\mu : \mathrm{dom}(\Pi) \to \mathrm{dom}(\Pi')$ *we write* $\Pi, e \overset{\mu}{\sim} \Pi', e'$ *when*

(i) $\mu(e) = e'$, $\forall l \in \mathrm{dom}(\Pi).\ \mu(\Pi(l)) = \Pi'(\mu(l))$.
(ii) $\mathrm{FL}(e) \subset \mathrm{dom}(\Pi)$, $\forall l \in \mathrm{dom}(\Pi).\ \mathrm{FL}(\Pi(l)) \subset \mathrm{dom}(\Pi)$.

Here application of μ to an expression denotes the obvious substitution and FL is the set of locations occurring in an expression. The second clause prevents stray locations, it is necessary because of the untyped approach we take in this appendix. Notice how the definition also ensures that we have $\mathrm{FL}(e') \subset \mathrm{dom}(\Pi')$ and that for all $l' \in \mathrm{dom}(\Pi')$ have $\mathrm{FL}(\Pi'(l')) \subset \mathrm{dom}(\Pi')$. We are able to prove a simple coherence result:

Lemma 24. *If we have* $\Pi, e \overset{\mu}{\sim} \Pi', e'$, $\Pi, e \Downarrow \Pi_f, v$ *and* $\Pi', e' \Downarrow \Pi'_f, v'$ *then there is a a bijection* $\mu_f : \mathrm{dom}(\Pi_f) \to \mathrm{dom}(\Pi'_f)$ *with* $\Pi_f, v \overset{\mu_f}{\sim} \Pi'_f, v'$ *and with* $\mu_f(l) = \mu(l)$ *for all* $l \in \mathrm{dom}(\Pi)$.

Notice that we implicitly rely on the fact that the domains of syntactic stores cannot shrink during evaluation.

Proof. The lemma is stronger than we need to prove soundness but the formulation is necessary for a proof by induction on the structure of the first evaluation derivation to go through. We give details of the case of memory allocation, i.e., we look at the rule

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \mathtt{ref}(e) \Downarrow \Pi'[l \mapsto v], l}\ (l \notin \mathrm{dom}(\Pi'))$$

and assume that the lemma holds for the premise. Pick $\Pi^\dagger$, $e^\dagger$, $\Pi^\dagger_f$, $v^\dagger$ and any bijection $\mu : \mathrm{dom}(\Pi) \to \mathrm{dom}(\Pi^\dagger)$ and assume that $\Pi, \mathtt{ref}(e) \overset{\mu}{\sim} \Pi^\dagger, e^\dagger$ and that $\Pi^\dagger, e^\dagger \Downarrow \Pi^\dagger_f, v^\dagger$. As we have $e^\dagger = \mu(\mathtt{ref}(e)) = \mathtt{ref}(\mu(e))$ we must have that the derivation of $\Pi^\dagger, e^\dagger \Downarrow \Pi^\dagger_f, v^\dagger$ ends in

$$\frac{\Pi^\dagger, \mu(e) \Downarrow \Pi^\ddagger, v^\ddagger}{\Pi^\dagger, \mathtt{ref}(\mu(e)) \Downarrow \Pi^\ddagger[l^\dagger \mapsto v^\ddagger], l^\dagger}$$

for some $\Pi^\ddagger$, $v^\ddagger$ and $l^\dagger \notin \mathrm{dom}(\Pi^\ddagger)$ with $\Pi^\dagger_f = \Pi^\ddagger[l^\dagger \mapsto v^\ddagger]$ and $v^\dagger = l^\dagger$. By the induction hypothesis and the fact that $\Pi, e \overset{\mu}{\sim} \Pi^\dagger, \mu(e)$ we get a bijection $\mu^\dagger : \mathrm{dom}(\Pi') \to \mathrm{dom}(\Pi^\ddagger)$ extending μ with $\Pi', v \overset{\mu^\dagger}{\sim} \Pi^\ddagger, v^\ddagger$. And as we know $l \notin \mathrm{dom}(\Pi')$ and $l^\dagger \notin \mathrm{dom}(\Pi^\ddagger)$ we may extend $\mu^\dagger$ to a bijection $\mu^\dagger[l \mapsto l^\dagger] : \mathrm{dom}(\Pi'[l \mapsto v]) \to \mathrm{dom}(\Pi^\ddagger[l^\dagger \mapsto v^\ddagger])$. It is immediate that $\mu^\dagger[l \mapsto l^\dagger](l) = l^\dagger$. For $l' \in \mathrm{dom}(\Pi'[l \mapsto v])$ we have either $l' \in \mathrm{dom}(\Pi')$ and hence that

$$\begin{aligned}\mu^\dagger[l \mapsto l^\dagger](\Pi'[l \mapsto v](l')) &= \mu^\dagger(\Pi'(l')) \\ &= \Pi^\ddagger(\mu^\dagger(l')) \\ &= \Pi^\ddagger[l^\dagger \mapsto v^\ddagger](\mu^\dagger[l \mapsto l^\dagger](l'))\end{aligned}$$

or we have $l' = l$ in which case we get $\mu^\dagger[l \mapsto l^\dagger](\Pi'[l \mapsto v](l')) = v^\ddagger = \Pi^\ddagger[l^\dagger \mapsto v^\ddagger](\mu^\dagger[l \mapsto l^\dagger](l'))$. We now have

$$\Pi'[l \mapsto v], l \overset{\mu^\dagger[l \mapsto l^\dagger]}{\sim} \Pi^\ddagger[l^\dagger \mapsto v^\ddagger], l^\dagger$$

as the second clause of Definition 23 easily is verified. □

We also need a deterministic version of the operational semantics denoted $\Downarrow^*$, it is identical to the previously defined except for a new rule for memory allocation:

$$\frac{\Pi, e \Downarrow \Pi', v}{\Pi, \mathtt{ref}(e) \Downarrow \Pi'[l \mapsto v], l}\ \begin{array}{l}(l \notin \mathrm{dom}(\Pi'),\\ \forall l' < l.\ l' \in \mathrm{dom}(\Pi'))\end{array}$$

Switching from nondeterminism to the determinism does not affect termination, but with appropriate substitution lemmas we may prove soundness with respect to the deterministic version in the ordinary way. This is made precise in the following two results:

Lemma 25. $\Pi, e \overset{\mu}{\sim} \Pi^*, e^*$ *and* $\Pi, e \downarrow$ *implies that* $\Pi^*, e^* \downarrow^*$.

Here the conclusion denotes existence of a syntactic store Π' and a value v such that $\Pi^*, e^* \Downarrow^* \Pi', v$.

Proof. The proof is by induction on derivation of the termination assumption – again we prove a little more than we need so as to make the induction go through. We give details in two cases:

Consider the first case of branching on a sum, i.e., the rule

$$\frac{\Pi, e \Downarrow \Pi', \mathtt{inl}(v) \qquad \Pi', e_0[v/x] \Downarrow \Pi'', v'}{\Pi, \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(x).\, e_0\ \mathtt{else}\ \mathtt{inr}(x).\, e_1 \Downarrow \Pi'', v'}$$

and assume that the lemma holds for the premises. Take any syntactic store Π^*, any expression e^* and any bijection $\mu : \mathrm{dom}(\Pi) \to \mathrm{dom}(\Pi^*)$ and assume that $\Pi, \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(x).\, e_0\ \mathtt{else}\ \mathtt{inr}(x).\, e_1 \overset{\mu}{\sim} \Pi^*, e^*$ holds. By applying the bijection we get that $e^* = \mathtt{case}\ \mu(e)\ \mathtt{of}\ \mathtt{inl}(x).\, \mu(e_0)\ \mathtt{else}\ \mathtt{inr}(x).\, \mu(e_1)$, also we have $\Pi, e \overset{\mu}{\sim} \Pi^*, \mu(e)$ which by the induction hypothesis yields $\Pi^\dagger$ and v_0^* with $\Pi^*, \mu(e) \Downarrow^* \Pi^\dagger, v_0^*$. Lemma 24 provides us with a bijection $\mu_f : \mathrm{dom}(\Pi') \to \mathrm{dom}(\Pi^\dagger)$ extending μ such that $\Pi', \mathtt{inl}(v) \overset{\mu_f}{\sim} \Pi^\dagger, v_0^*$, in particular we must have $v_0^* = \mu_f(\mathtt{inl}(v)) = \mathtt{inl}(\mu_f(v))$. As $\mathrm{FL}(e_0) \subset \mathrm{dom}(\Pi)$ we get that $\mu_f(e_0[v/x]) = \mu_f(e_0)[\mu_f(v)/x] = \mu(e_0)[\mu_f(v)/x]$ which implies that $\Pi', e_0[v/x] \overset{\mu_f}{\sim} \Pi^\dagger, \mu(e_0)[\mu_f(v)/x]$ and the induction hypothesis takes us home.

Consider now assignment, i.e., the rule

$$\frac{\Pi, e_0 \Downarrow \Pi', l \qquad \Pi', e_1 \Downarrow \Pi'', v}{\Pi, e_0 := e_1 \Downarrow \Pi''[l \mapsto v], ()}\ (l \in \mathrm{dom}(\Pi''))$$

and assume that the lemma holds for the premises. Take any syntactic store Π^*, any expression e^* and any bijection $\mu : \mathrm{dom}(\Pi) \to \mathrm{dom}(\Pi^*)$ and assume that $\Pi, e_0 := e_1 \overset{\mu}{\sim} \Pi^*, e^*$ holds. We get that $e^* = \mu(e_0 := e_1) = \mu(e_0) := \mu(e_1)$ and that $\Pi, e_0 \overset{\mu}{\sim} \Pi^*, \mu(e_0)$ which by the induction hypothesis yields $\Pi^\dagger$ and v_0^* with $\Pi^*, \mu(e_0) \Downarrow^* \Pi^\dagger, v_0^*$. Lemma 24 provides us with a bijection $\mu_f : \mathrm{dom}(\Pi') \to \mathrm{dom}(\Pi^\dagger)$ extending μ such that $\Pi', l \overset{\mu_f}{\sim} \Pi^\dagger, v_0^*$. As $\mathrm{FL}(e_1) \subset \mathrm{dom}(\Pi)$ we get that $\mu_f(e_1) = \mu(e_1)$ which implies that $\Pi', e_1 \overset{\mu_f}{\sim} \Pi^\dagger, \mu(e_1)$ and the induction hypothesis gives us $\Pi^\ddagger$ and v_1^* with $\Pi^\dagger, \mu(e_1) \Downarrow^* \Pi^\ddagger, v_1^*$. Finally we just remark that $v_0^* = \mu_f(l) \in \mathrm{dom}(\Pi^\dagger) \subset \mathrm{dom}(\Pi^\ddagger)$ and are done. □

Proposition 26. *For* $\emptyset \mid \Delta \mid \emptyset \vdash e : \tau$ *and* $\emptyset \mid \Delta \mid \emptyset \vdash \Pi$, *any syntactic store* Π' *and any value* v *we get*

$$\begin{aligned}&\Pi, e \Downarrow^* \Pi', v \implies \\ &\quad \exists \Delta' \sqsupseteq \Delta.\ \emptyset \mid \Delta' \mid \emptyset \vdash v : \tau \wedge \emptyset \mid \Delta' \mid \emptyset \vdash \Pi' \wedge \\ &\quad [\![\emptyset \mid \Delta \mid \emptyset \vdash e : \tau]\!]^{[\![\Delta \vdash \Pi]\!]} = \lfloor [\![\Delta' \vdash \Pi']\!], [\![\Delta' \vdash v]\!] \rfloor.\end{aligned}$$

Proof of Proposition 8. Assume $\emptyset \mid \Delta \mid \emptyset \vdash e : \mathtt{int}$ and $\emptyset \mid \Delta \mid \emptyset \vdash \Pi$. Suppose that for some syntactic store Π' and some $n \in \mathbb{N}$ we have $\Pi, e \Downarrow \Pi', n$. As a consequence of our assumptions we have $\Pi, e \overset{\mu}{\sim} \Pi, e$ with $\mu = id_{\mathrm{dom}(\Pi)}$. By Lemma 25 there is Π^* and v with $\Pi, e \Downarrow^* \Pi^*, v$ and Lemma 24 ensures that $v = n$ as numbers are invariant under any renaming of locations. All that remains now is to apply the above proposition. □

D. Untyped Interpretation

$$[\![\Xi \mid \Delta \mid \Gamma \vdash x : \tau]\!]^s_\rho = \lfloor s, \rho(x) \rfloor \quad [\![\Xi \mid \Delta \mid \Gamma \vdash () : \mathtt{unit}]\!]^s_\rho = \lfloor s, in_{\mathtt{unit}}(*) \rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash n : \mathtt{int}]\!]^s_\rho = \lfloor s, in_{\mathtt{int}}(n) \rfloor \quad [\![\Xi \mid \Delta \mid \Gamma \vdash l : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s, \lambda_l \rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{op}(e_0 \pm e_1) : \mathtt{int}]\!]^s_\rho = \begin{cases} \lfloor s'', in_{\mathtt{int}}(n_0 \pm n_1) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \mathtt{int}]\!]^s_\rho = \lfloor s', in_{\mathtt{int}}(n_0) \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \mathtt{int}]\!]^{s'}_\rho = \lfloor s'', in_{\mathtt{int}}(n_1) \rfloor \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{ifzero}\ e\ \mathtt{then}\ e_0\ \mathtt{else}\ e_1 : \tau]\!]^s_\rho = \begin{cases} \lfloor s'', u \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \mathtt{int}]\!]^s_\rho = \lfloor s', in_{\mathtt{int}}(0) \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau]\!]^{s'}_\rho = \lfloor s'', u \rfloor \end{array}\right. \\ \lfloor s'', u \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \mathtt{int}]\!]^s_\rho = \lfloor s', in_{\mathtt{int}}(n) \rfloor, n \neq 0, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s'}_\rho = \lfloor s'', u \rfloor \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash (e_0, e_1) : \tau_0 \times \tau_1]\!]^s_\rho = \begin{cases} \lfloor s'', in_\times(\lfloor u_0, u_1 \rfloor) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau_0]\!]^s_\rho = \lfloor s', u_0 \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau_1]\!]^{s'}_\rho = \lfloor s'', u_1 \rfloor \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fst}(e) : \tau_0]\!]^s_\rho = \begin{cases} \lfloor s', u_0 \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 \times \tau_1]\!]^s_\rho = \lfloor s', in_\times(\lfloor u_0, u_1 \rfloor) \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{snd}(e) : \tau_1]\!]^s_\rho = \begin{cases} \lfloor s', u_1 \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 \times \tau_1]\!]^s_\rho = \lfloor s', in_\times(\lfloor u_0, u_1 \rfloor) \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{inl}^{\tau_0+\tau_1}(e) : \tau_0 + \tau_1]\!]^s_\rho = \begin{cases} \lfloor s', (in_+ \circ inl)(u) \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0]\!]^s_\rho = \lfloor s', u \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{inr}^{\tau_0+\tau_1}(e) : \tau_0 + \tau_1]\!]^s_\rho = \begin{cases} \lfloor s', (in_+ \circ inr)(u) \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau_1]\!]^s_\rho = \lfloor s', u \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(x).\, e_0\ \mathtt{else}\ \mathtt{inr}(x).\, e_1 : \tau]\!]^s_\rho =$$

$$\begin{cases} \lfloor s'', u' \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 + \tau_1]\!]^s_\rho = \lfloor s', (in_+ \circ inl)(u) \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma, x : \tau_0 \vdash e_0 : \tau]\!]^{s'}_{\rho[x \mapsto u]} = \lfloor s'', u' \rfloor \end{array}\right. \\ \lfloor s'', u' \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau_0 + \tau_1]\!]^s_\rho = \lfloor s', (in_+ \circ inr)(u) \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma, x : \tau_1 \vdash e_1 : \tau]\!]^{s'}_{\rho[x \mapsto u]} = \lfloor s'', u' \rfloor \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fold}^{\mu\alpha.\tau}(e) : \mu\alpha.\tau]\!]^s_\rho = \begin{cases} \lfloor s', in_\mu(u) \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau[\mu\alpha.\tau/\alpha]]\!]^s_\rho = \lfloor s', u \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{unfold}^{\mu\alpha.\tau}(e) : \tau[\mu\alpha.\tau/\alpha]]\!]^s_\rho = \begin{cases} \lfloor s', u \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \mu\alpha.\tau]\!]^s_\rho = \lfloor s', in_\mu(u) \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau]\!]^s_\rho = \lfloor s, in_\forall(\lambda s' \in S_\downarrow . [\![\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau]\!]^{s'}_\rho) \rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash e[\tau_1] : \tau_0[\tau_1/\alpha]]\!]^s_\rho = \begin{cases} \varphi(s') & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \forall\alpha.\tau_0]\!]^s_\rho = \lfloor s', in_\forall(\varphi) \rfloor \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fix}^{\tau_0 \to \tau_1} f(x).e : \tau_0 \to \tau_1]\!]^s_\rho = \lfloor s, (in_\to \circ fix)\big(\lambda\varphi \in S \otimes U \multimap S \otimes U.$$
$$\lambda(s', u) \in S_\downarrow \times U_\downarrow . [\![\Xi \mid \Delta \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash e : \tau_1]\!]^{s'}_{\rho[f \mapsto in_\to(\varphi), x \mapsto u]}\big) \rfloor$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash e_0(e_1) : \tau_1]\!]^s_\rho = \begin{cases} \varphi(\lfloor s'', u \rfloor) & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau_0 \to \tau_1]\!]^s_\rho = \lfloor s', in_\to(\varphi) \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau_0]\!]^{s'}_\rho = \lfloor s'', u \rfloor \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash \mathtt{ref}(e) : \tau\ \mathtt{ref}]\!]^s_\rho = \begin{cases} \lfloor s'[l \mapsto u], \lambda_l \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau]\!]^s_\rho = \lfloor s', u \rfloor, \\ l \notin \mathrm{dom}(s'), \forall l' < l.\, l' \in \mathrm{dom}(s') \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash\, !\, e : \tau]\!]^s_\rho = \begin{cases} \lfloor s', s'(l) \rfloor & [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', \lambda_l \rfloor, l \in \mathrm{dom}(s') \\ \lfloor s', \pi_n(s'(l)) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', u \rfloor, \pi_{n+1}(u) = \lambda_l^{n+1}, \\ \pi_{n+2}(u) \neq \lambda_l^{n+2}, l \in \mathrm{dom}(s'), \pi_n(s'(l)) \neq \bot \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\Xi \mid \Delta \mid \Gamma \vdash e_0 := e_1 : \mathtt{unit}]\!]^s_\rho = \begin{cases} \lfloor s''[l \mapsto u], in_{\mathtt{unit}}(*) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', \lambda_l \rfloor, \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s'}_\rho = \lfloor s'', u \rfloor, l \in \mathrm{dom}(s'') \end{array}\right. \\ \lfloor s''[l \mapsto \pi_n(u)], in_{\mathtt{unit}}(*) \rfloor & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \mathtt{ref}]\!]^s_\rho = \lfloor s', u' \rfloor \\ [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s'}_\rho = \lfloor s'', u \rfloor, \pi_{n+1}(u') = \lambda_l^{n+1}, \\ \pi_{n+2}(u') \neq \lambda_l^{n+2}, l \in \mathrm{dom}(s''), \pi_n(u) \neq \bot \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

Figure 11. Untyped interpretation. The general form of the left hand side is $[\![\Xi \mid \Delta \mid \Gamma \vdash e : \tau]\!]^s_\rho$ with $s \in S_\downarrow$ and $\rho \in ((\mathrm{dom}(\Gamma) \to U_\downarrow)_\perp)_\downarrow$. The right hand side is an element of $S \otimes U$, remember that $S = (Loc \xrightarrow{fin} U_\downarrow)_\perp$.

E. Elaboration of Proof of Theorem 17

Proof of Theorem 17, continued. Consider the assignment case

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau\ \mathtt{ref} \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau}{\Xi \mid \Delta \mid \Gamma \vdash e_0 := e_1 : \mathtt{unit}}$$

and assume that the proposition holds for the premises. We pick arbitrary $\Xi' \supset \Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta' \in \mathbf{World}_{\Xi'}$ with $\Delta' \sqsupseteq \Delta$ and $\rho, \rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ as specified in the definition of semantic relatedness. Also we take arbitrary $(s, s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k, k') \in [\![\mathtt{unit}]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. We build $k_0 : S \otimes U \to S \otimes U$ from the map that takes $(s_0, u_0) \in S_\downarrow \times U_\downarrow$ to

$$k(\lfloor s_1[l \mapsto u_1], in_{\mathtt{unit}}(*)\rfloor) \quad \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s_0}_\rho = \\ \lfloor s_1, u_1\rfloor, \\ u_0 = \lambda_l, \\ l \in \mathrm{dom}(s_1) \end{array}\right.$$

$$k(\lfloor s_1[l \mapsto \pi_n(u_1)], in_{\mathtt{unit}}(*)\rfloor) \quad \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau]\!]^{s_0}_\rho = \\ \lfloor s_1, u_1\rfloor, \\ \pi_{n+1}(u_0) = \lambda_l^{n+1}, \\ \pi_{n+2}(u_0) \neq \lambda_l^{n+2}, \\ l \in \mathrm{dom}(s_1), \\ \pi_n(u_1) \neq \bot \end{array}\right.$$

$$\bot \quad \text{otherwise}$$

and naturally build $k_0' : S \otimes U \to S \otimes U$ in the same way, only we exchange k' for k. By first induction hypothesis it will now suffice to prove that $(k_0, k_0') \in [\![\tau\ \mathtt{ref}]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. Aiming to prove this, we pick $[\Xi_0|\overline{\nu}_0|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$, $(s_0, s_0') \in [\![\Delta_0]\!]^S_{\Xi_0}(\overline{\nu}_0)$ and $(u_0, u_0') \in [\![\tau\ \mathtt{ref}]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. We now branch according to the two clauses of the definition of interpretation of reference types.

It may be the case that $u_0 = u_0' = \lambda_l$ for some $l \in \mathrm{dom}(\Delta_0)$ with $[\![\tau]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1) = [\![\Delta_1(l)]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$ for all $[\Xi_1|\overline{\nu}_1|\Delta_1] \sqsupseteq [\Xi_0|\overline{\nu}_0|\Delta_0]$. We build the map $k_1 : S \otimes U \multimap S \otimes U$ from

$$\lambda(s_1, u_1).\begin{cases} k(\lfloor s_1[l \mapsto u_1], in_{\mathtt{unit}}(*)\rfloor) & l \in \mathrm{dom}(s_1) \\ \bot & \text{otherwise} \end{cases}$$

and $k_1 : S \otimes U \multimap S \otimes U$ similarly with k' exchanged for k. We have $\Xi_0 \supset \Xi' \supset \Xi$, $\overline{\nu}_0 \in \mathbf{SCT}^{\Xi_0}$, $\Delta_0 \sqsupseteq \Delta' \sqsupseteq \Delta$ and for any $x \in \mathrm{dom}(\Gamma)$ we have

$$(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi'}(\overline{\nu}')(\Delta') \subset [\![\Gamma(x)]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$$

by the Monotonicity Lemma. It only remains to prove $(k_1, k_1') \in [\![\tau]\!]^K_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$ to be able to apply the second induction hypothesis and be done with it. We unroll the definition and pick $[\Xi_1|\overline{\nu}_1|\Delta_1] \sqsupseteq [\Xi_0|\overline{\nu}_0|\Delta_0]$ and $(s_1, s_1') \in [\![\Delta_1]\!]^S_{\Xi_1}(\overline{\nu}_1)$ and $(u_1, u_1') \in [\![\tau]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$. But $(s_1[l \mapsto u_1], s_1'[l \mapsto u_1']) \in [\![\Delta_1]\!]^S_{\Xi_1}(\overline{\nu}_1)$ is now within reach and we are done as we obviously have $(in_{\mathtt{unit}}(*), in_{\mathtt{unit}}(*)) \in [\![\mathtt{unit}]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$ and $(k, k') \in [\![\tau]\!]^K_{\Xi'}(\overline{\nu}')(\Delta') \subset [\![\tau]\!]^K_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$.

The second subcase to consider is $u_0 = u_0' = \lambda_l^{n+1}$ for some $l \in \mathrm{dom}(\Delta_0)$ and $n \in \mathbb{N}$ with $[\![\tau]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1) \overset{n}{\subset} [\![\Delta_1(l)]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1) \cup \{(\bot, \bot)\}$ for all $[\Xi_1|\overline{\nu}_1|\Delta_1] \sqsupseteq [\Xi_0|\overline{\nu}_0|\Delta_0]$. Somewhat predictably we build the $k_1 : S \otimes U \multimap S \otimes U$ from

$$\lambda(s_1, u_1).\begin{cases} k(\lfloor s_1[l \mapsto \pi_n(u_1)], in_{\mathtt{unit}}(*)\rfloor) & \left[\begin{array}{l} l \in \mathrm{dom}(s_1), \\ \pi_n(u_1) \neq \bot \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

and $k_1 : S \otimes U \multimap S \otimes U$ similarly with k' exchanged for k. As above it now remains to prove $(k_1, k_1') \in [\![\tau]\!]^K_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$; we pick $[\Xi_1|\overline{\nu}_1|\Delta_1] \sqsupseteq [\Xi_0|\overline{\nu}_0|\Delta_0]$, $(s_1, s_1') \in [\![\Delta_1]\!]^S_{\Xi_1}(\overline{\nu}_1)$ and $(u_1, u_1') \in [\![\tau]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$. But we have $(s_1[l \mapsto \pi_n(u_1)], s_1'[l \mapsto \pi_n(u_1')]) \in [\![\Delta_1]\!]^S_{\Xi_1}(\overline{\nu}_1)$ from the interpretation of the reference type and are done.

We now turn to the case of type abstraction. We look at the rule

$$\frac{\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau}{\Xi \mid \Delta \mid \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau}\,(\Xi \vdash \Delta, \Xi \vdash \Gamma)$$

and as usual assume that the proposition holds for the premise. Following standard procedure we pick arbitrary $\Xi' \supset \Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta' \in \mathbf{World}_{\Xi'}$ with $\Delta' \sqsupseteq \Delta$ and $\rho, \rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ such that $(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi'}(\overline{\nu}')(\Delta')$ for all $x \in \mathrm{dom}(\Gamma)$. Also we take arbitrary $(s, s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k, k') \in [\![\forall\alpha.\tau]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. From the definition of interpretation of type abstraction it suffices to show that

$$\big(in_\forall([\![\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau]\!]^{(-)}_\rho), in_\forall([\![\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau]\!]^{(-)}_{\rho'})\big)$$

lies in $[\![\forall\alpha.\tau]\!]_{\Xi'}(\overline{\nu}')(\Delta')$. This again comes down to proving

$$\big([\![\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau]\!]^{(-)}_\rho, [\![\Xi, \alpha \mid \Delta \mid \Gamma \vdash e : \tau]\!]^{(-)}_{\rho'}\big)$$

a member of $[\![\tau]\!]^T_{\Xi_0,\alpha}(\overline{\nu}_0, \nu)(\Delta_0)$ for $[\Xi_0|\overline{\nu}_0|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$ and $\nu \in \mathbf{SCT}_{\Xi'}$ arbitrary. But as

$$(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi'}(\overline{\nu}')(\Delta') \subset [\![\Gamma(x)]\!]_{\Xi_0,\alpha}(\overline{\nu}_0, \nu)(\Delta_0)$$

holds for all $x \in \mathrm{dom}(\Gamma)$ by the Monotonicity Lemma we are done by application of the induction hypothesis.

We now look into the case of function abstraction, i.e., the rule

$$\frac{\Xi \mid \Delta \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash e : \tau_1}{\Xi \mid \Delta \mid \Gamma \vdash \mathtt{fix}^{\tau_0 \to \tau_1} f(x).e : \tau_0 \to \tau_1}$$

and as usual assume that the proposition holds for the premise. Following standard procedure we pick arbitrary $\Xi' \supset \Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta' \in \mathbf{World}_{\Xi'}$ with $\Delta' \sqsupseteq \Delta$ and $\rho, \rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ such that $(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi'}(\overline{\nu}')(\Delta')$ for all $x \in \mathrm{dom}(\Gamma)$. Also we take arbitrary $(s, s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k, k') \in [\![\tau_0 \to \tau_1]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. From the definition of interpretation of function abstraction it shall suffice to show that the pair

$$\big[(in_\to \circ fix)\big(\lambda\varphi.\lambda(s, u).\,[\![e : \tau_1]\!]^s_{\rho[f \mapsto in_\to(\varphi), x \mapsto u]}\big),$$
$$(in_\to \circ fix)\big(\lambda\varphi'.\lambda(s', u').\,[\![e : \tau_1]\!]^{s'}_{\rho'[f \mapsto in_\to(\varphi'), x \mapsto u']}\big)\big]$$

lies in $[\![\tau_0 \to \tau_1]\!]_{\Xi'}(\overline{\nu}')(\Delta')$. For layout reasons we have omitted part of the typing judgments and the domains of the abstractions: the domain of φ and φ' is $S \otimes U \multimap S \otimes U$ and the domain of (s, u) and (s', u') is $S_\downarrow \times U_\downarrow$. By the interpretation of function types and for admissibility reasons this comes down to proving that for any $[\Xi_0|\overline{\nu}_0|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$ and any $(u, u') \in [\![\tau_0]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$ we have the pair

$$\big[[\![e : \tau_1]\!]^{(-)}_{\rho[f \mapsto in_\to(\varphi), x \mapsto u]}, [\![e : \tau_1]\!]^{(-)}_{\rho'[f \mapsto in_\to(\varphi'), x \mapsto u']}\big]$$

in $[\![\tau_1]\!]^T_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$ under the assumption that $\varphi, \varphi' \in S \otimes U \multimap S \otimes U$ with the property that for any $[\Xi_0|\overline{\nu}_0|\Delta_0] \sqsupseteq [\Xi'|\overline{\nu}'|\Delta']$ and any $(u, u') \in [\![\tau_0]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$ we have

$$\big[\varphi(\lfloor -, u\rfloor), \varphi'(\lfloor -, u'\rfloor)\big] \in [\![\tau_1]\!]^T_{\Xi_0}(\overline{\nu}_0)(\Delta_0).$$

The induction hypothesis together with the Monotonicity Lemma sees to that.

The final case we consider in detail is function application, i.e., we look at the rule

$$\frac{\Xi \mid \Delta \mid \Gamma \vdash e_0 : \tau_0 \to \tau_1 \qquad \Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau_0}{\Xi \mid \Delta \mid \Gamma \vdash e_0(e_1) : \tau_1}$$

and assume that the proposition holds for the premises. For the seventh time we pick arbitrary $\Xi' \supset \Xi$, $\overline{\nu}' \in \mathbf{SCT}^{\Xi'}$, $\Delta' \in \mathbf{World}_{\Xi'}$ with $\Delta' \sqsupseteq \Delta$ and $\rho, \rho' \in \mathrm{dom}(\Gamma) \to U_\downarrow$ such that $(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi'}(\overline{\nu}')(\Delta')$ for all $x \in \mathrm{dom}(\Gamma)$. And we take arbitrary $(s, s') \in [\![\Delta']\!]^S_{\Xi'}(\overline{\nu}')$ and $(k, k') \in [\![\tau_1]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. We build $k_0 : S \otimes U \to S \otimes U$ from the map

$$\lambda(s_0, u_0).\ \begin{cases} k(\varphi(\lfloor s_1, u_1 \rfloor)) & \left[\begin{array}{l} [\![\Xi \mid \Delta \mid \Gamma \vdash e_1 : \tau_0]\!]^{s_0}_\rho = \\ \lfloor s_1, u_1 \rfloor, u_0 = in_\forall(\varphi) \end{array}\right. \\ \bot & \text{otherwise} \end{cases}$$

and naturally build $k'_0 : S \otimes U \to S \otimes U$ in the same way, only we exchange k' for k. By the first induction hypothesis it will now suffice to prove that $(k_0, k'_0) \in [\![\tau_0 \to \tau_1]\!]^K_{\Xi'}(\overline{\nu}')(\Delta')$. Aiming to prove this, we pick $[\Xi_0 | \overline{\nu}_0 | \Delta_0] \sqsupseteq [\Xi' | \overline{\nu}' | \Delta']$, $(s_0, s'_0) \in [\![\Delta_0]\!]^S_{\Xi_0}(\overline{\nu}_0)$ and $(u_0, u'_0) \in [\![\tau_0 \to \tau_1]\!]_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. The latter implies the existence of $\varphi, \varphi' \in S \otimes U \multimap S \otimes U$ such that $u_0 = in_\forall(\varphi)$ and $u'_0 = in_\forall(\varphi')$ and we have

$$[\varphi(\lfloor -, u \rfloor), \varphi'(\lfloor -, u' \rfloor)] \in [\![\tau_1]\!]^T_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$$

for any choice of $[\Xi_1 | \overline{\nu}_1 | \Delta_1] \sqsupseteq [\Xi_0 | \overline{\nu}_0 | \Delta_0]$ and any $(u, u') \in [\![\tau_0]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$. We can now build maps $k_1, k'_1 : S \otimes U \to S \otimes U$, the former form the map $\lambda(s_1, u_1) \in S_\downarrow \times U_\downarrow.\ k(\varphi(\lfloor s_1, u_1 \rfloor))$ and the latter from $\lambda(s'_1, u'_1) \in S_\downarrow \times U_\downarrow.\ k'(\varphi'(\lfloor s'_1, u'_1 \rfloor))$. By the second induction hypothesis and the Monotonicity Lemma it shall now suffice to prove that $(k_1, k'_1) \in [\![\tau_0]\!]^K_{\Xi_0}(\overline{\nu}_0)(\Delta_0)$. Once again we look into the future and pick $[\Xi_1 | \overline{\nu}_1 | \Delta_1] \sqsupseteq [\Xi_0 | \overline{\nu}_0 | \Delta_0]$, $(s_1, s'_1) \in [\![\Delta_1]\!]^S_{\Xi_1}(\overline{\nu}_1)$ and $(u_1, u'_1) \in [\![\tau_0]\!]_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$. All that remains is to remark that we of course have $(k, k') \in [\![\tau_1]\!]^K_{\Xi_1}(\overline{\nu}_1)(\Delta_1)$ too. $\square$

Chapter 3

Realizability Semantics of Parametric Polymorphism, General References, and Recursive Types

Under consideration for publication in Math. Struct. in Comp. Science

Realizability Semantics of Parametric Polymorphism, General References, and Recursive Types

LARS BIRKEDAL
KRISTIAN STØVRING
JACOB THAMSBORG

IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark.

Received 9 May 2010

We present a realizability model for a call-by-value, higher-order programming language with parametric polymorphism, general first-class references, and recursive types. The main novelty is a relational interpretation of open types that include general reference types. The interpretation uses a new approach to modeling references.
The universe of semantic types consists of world-indexed families of logical relations over a universal predomain. In order to model general reference types, worlds are finite maps from locations to semantic types: this introduces a circularity between semantic types and worlds that precludes a direct definition of either. Our solution is to solve a recursive equation in an appropriate category of metric spaces. In effect, types are interpreted using a Kripke logical relation over a recursively defined set of worlds.
We illustrate how the model can be used to prove simple equivalences between different implementations of imperative abstract data types.

1. Introduction

In this article we develop a semantic model of a call-by-value programming language with impredicative and parametric polymorphism, general first-class references, and recursive types. Motivations for conducting this study include:

— Extending the approach to reasoning about abstract data types via relational parametricity from pure languages to more realistic languages with effects, here general references. We discussed this point of view extensively earlier (Birkedal et al. 2009).
— Investigating what semantic structures are needed in general models for effects. Indeed, we see the present work as a pilot study for studying general type theories and models of effects (e.g., Levy 2006; Plotkin and Power 2004), in which we identify key ingredients needed for semantic modeling of general first-class references.
— Paving the way for developing models of separation logic for ML-like languages with reference types. Earlier such models of separation logic (Petersen et al. 2008) only

treat so-called strong references, where the type on the contents of a reference cell can vary: therefore proof rules cannot take advantage of the strong invariants provided by ML-style reference types.

We now give an overview of the conceptual development of the paper. The development is centered around three recursively defined structures, defined in three steps. In slogan form, there is one recursively defined structure for each of the type constructors $\forall$, ref, and μ alluded to in the title.

First, since the language involves impredicative polymorphism, the semantic model is based on a realizability interpretation (Amadio 1991) over a certain recursively defined predomain V. Using this predomain we can give a denotational semantics of an untyped version of the language. This part is mostly standard, except for the fact that we model locations as pairs (l, n), with l a natural number corresponding to a standard location and $n \in \mathbb{N} \cup \{\infty\}$ indicating the "approximation stage" of the location (Birkedal et al. 2009). These pairs, called semantic locations, are needed for modeling reference types in step three. Intuitively, the problem with the more standard approach of modeling locations as natural numbers is that such "flat" locations contain no approximation information that can be used to define relations by induction.

Second, to account for dynamic allocation of typed reference cells, we follow earlier work on modeling simple integer references (Benton and Leperchey 2005) and use a Kripke-style possible worlds model. Here, however, the set of worlds needs to be recursively defined since we treat general references. Semantically, a world maps locations to semantic types, which, following the general realizability idea, are certain world-indexed families of relations on V: this introduces a circularity between semantic types and worlds that precludes a direct definition of either. Thus we need to solve recursive equations of approximately the following form

$$\begin{aligned} \mathcal{W} &= \mathbb{N}_0 \rightharpoonup_{\mathit{fin}} \mathcal{T} \\ \mathcal{T} &= \mathcal{W} \to \mathit{CURel}(V) \end{aligned}$$

even in order to define the space in which types will be modeled. (Here $\mathit{CURel}(V)$ is a set of "good" relations on V.) We formally define the recursive equations in certain ultrametric spaces and show how to solve them using known results from metric-space based semantics. The employed metric on relations on V is well-known from work on interpreting recursive types and impredicative polymorphism (Abadi and Plotkin 1990; Amadio 1991; Amadio and Curien 1998; Cardone 1989; MacQueen et al. 1986); here we extend its use to reference types (combined with these two other features).

Third, having now defined the space in which types should be modeled, the actual semantics of types can be defined. For recursive types, that also involves a recursive definition. Since the space $\mathcal{T}$ of semantic types is a metric space we can employ Banach's fixed point theorem to find a solution as the fixed point of a contractive operator on $\mathcal{T}$.[†] This involves interpreting the various type constructors of the language as non-expansive

[†] We remark that the fixed point could also be found using the technique of Pitts (1996); the proof techniques are very similar because of the particular way the requisite metrics are defined. In this article we do in any case need the metric-space formulation, but not the extra separation of positive

operators. For most type constructors doing so is straightforward, but for the reference-type constructor it is not. That is the reason for introducing the semantic locations mentioned above: using these, we can define a semantic reference-type operator (and show that it is non-expansive). In fact, we give an abstract proof that the probably most natural interpretation of reference types is, under certain assumptions, impossible. Therefore, semantic locations (or some other construct) are indeed necessary.

Finally, having now defined semantics of types using a family of world-indexed logical relations, we define the typed meaning of terms by proving the fundamental theorem of logical relations with respect to the untyped semantics of terms.

Limitations. The model we construct does not validate standard equivalences involving local state; indeed, it can only be used to equate computations that allocate references essentially "in lockstep." More precisely the model can only relate two program states if they contain the same number of locations. Furthermore, a certain technical requirement on the relations we consider ("uniformity") seems to be too restrictive. In recent work (Birkedal et al. 2010b) we have shown that both these problems can be overcome: one can use the techniques presented here to construct a model that validates sophisticated equivalences in the style of Ahmed et al. (2009). One key idea is to weaken the model such that the constructed relations can be thought of as inequalities instead of equalities; then one can prove results about contextual approximation rather than contextual equivalence. The model there is rather more complicated than the one in the present article, however. Here we rather aim to present the fundamental ideas behind Kripke logical relations over recursively defined sets of worlds.

Overview of the rest of the article. The rest of the article is organized as follows. In Section 2 we sketch the language we consider. In Section 3 we present the untyped semantics, corresponding to step one in the outline above. In Section 4 we present the typed semantics, corresponding to the last two steps. In Section 5 we give an abstract proof that a certain simpler interpretation of reference types is impossible. In Section 6 we present a few examples of reasoning using the model. Related work is discussed in Section 7.

2. Language

We consider a standard call-by-value language with universal types, iso-recursive types, ML-style reference types, and a ground type of integers. The language is sketched in Figure 1. Terms are not intrinsically typed; this allows us to give a denotational semantics of untyped terms. The typing rules are standard (Pierce 2002). In the figure, Ξ and Γ range over contexts of type variables and term variables, respectively.

The constructs that involve references have the following informal meaning: The term $\mathsf{ref}\ t$ allocates a new reference cell initialized with the result of evaluating t. The term $!t$

and negative arguments in recursive definitions of relations, and hence we define the meaning of recursive types via Banach's fixed point theorem (Amadio 1991; Amadio and Curien 1998).

looks up t in the current store (and gives an error if t evaluates to something which is not a location). Finally, the term $t_1 := t_2$ assigns the value of t_2 to t_1 (provided that t_1 evaluates to a location).

2.1. *Operational semantics*

We sketch an operational semantics of the untyped term language above. For this purpose, we temporarily add syntactic locations to the language:

$$t \quad ::= \quad \cdots \mid l \qquad (l \in \omega)$$

The set of *syntactic values* is then given by:

$$w \quad ::= \quad x \mid \overline{m} \mid l \mid () \mid (w_1, w_2) \mid \mathsf{inl}\, w \mid \mathsf{inr}\, w \mid \mathsf{fold}\, w \mid \Lambda\alpha.t \mid \lambda x.t$$

A term or syntactic value is *closed* if it contains no free variables and no free type variables.

A *syntactic store* is a finite map from locations to closed syntactic values. Let σ range over syntactic stores. A *configuration* is a pair (σ, t) consisting of a syntactic store σ and a closed term t.

The operational semantics is given by two judgements on configurations. First, $(\sigma, t) \Downarrow (\sigma', w)$ means that evaluation of t together with the store σ terminates with a value w and a possibly modified store σ'. Second, $(\sigma, t) \Downarrow \mathit{error}$ means that evaluation of t together with the store σ results in an error, either due to a memory fault or, e.g., an attempt to apply an integer constant to an argument.

Figure 2 shows some selected rules for the two judgements. As for the last rule, notice that evaluation of allocation is deterministic: the newly allocated location is always the least one not already in the store. In earlier work (Birkedal et al. 2009) we treated a non-deterministic allocation rule; here we stick to deterministic allocation in order to simplify the relationship with the denotational semantics in the next section.

A configuration (σ, t) *converges*, written $(\sigma, t) \Downarrow$, if there exist some σ' and w such that $(\sigma, t) \Downarrow (\sigma', w)$ holds. Two terms t and t' are *contextually equivalent* if for every (many-holed) term context C such that $C[t]$ and $C[t']$ are closed terms,

$$(\emptyset, C[t]) \Downarrow \iff (\emptyset, C[t']) \Downarrow \qquad \text{and} \qquad (\emptyset, C[t]) \Downarrow \mathit{error} \iff (\emptyset, C[t']) \Downarrow \mathit{error}.$$

3. Untyped semantics

We now give a denotational semantics for the untyped term language above. As usual for models of untyped languages, the semantics is given by means of a "universal" complete partial order (cpo) in which one can inject integers, pairs, functions, etc. This universal cpo is obtained by solving a recursive domain equation.

The only non-standard aspect of the semantics is the treatment of store locations: locations are modeled as elements of the cpo $\mathit{Loc} = \mathbb{N}_0 \times \overline{\omega}$ where $\overline{\omega}$ is the "vertical natural numbers" cpo $1 \sqsubset 2 \sqsubset \cdots \sqsubset n \sqsubset \cdots \sqsubset \infty$. (For notational reasons it is convenient to call the least element 1 rather than 0.) The intuitive idea is that locations can be approximated: the element $(l, \infty) \in \mathit{Loc}$ is the "ideal" location numbered l, while the elements

$$\text{Types:}\quad \tau ::= \mathsf{int} \mid 1 \mid \tau_1 \times \tau_2 \mid 0 \mid \tau_1 + \tau_2 \mid \mu\alpha.\tau \mid \forall\alpha.\tau \mid \alpha \mid \tau_1 \to \tau_2 \mid \mathsf{ref}\,\tau$$

$$\begin{aligned}\text{Terms:}\quad t ::= {} & x \mid \overline{m} \mid \mathsf{ifz}\,t_0\,t_1\,t_2 \mid t_1 + t_2 \mid t_1 - t_2 \mid () \mid (t_1, t_2) \mid \mathsf{fst}\,t \mid \mathsf{snd}\,t \\ & \mid \mathsf{void}\,t \mid \mathsf{inl}\,t \mid \mathsf{inr}\,t \mid \mathsf{case}\,t_0\,x_1.t_1\,x_2.t_2 \mid \mathsf{fold}\,t \mid \mathsf{unfold}\,t \\ & \mid \Lambda\alpha.t \mid t\,[\tau] \mid \lambda x.t \mid t_1\,t_2 \mid \mathsf{fix}\,f.\lambda x.t \mid \mathsf{ref}\,t \mid !t \mid t_1 := t_2\end{aligned}$$

Typing rules:

$$\frac{}{\Xi \mid \Gamma \vdash x : \tau}\ (\Xi \vdash \Gamma, \Gamma(x) = \tau) \qquad \frac{}{\Xi \mid \Gamma \vdash \overline{m} : \mathsf{int}}\ (\Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Gamma \vdash t_0 : \mathsf{int} \qquad \Xi \mid \Gamma \vdash t_1 : \tau \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash \mathsf{ifz}\,t_0\,t_1\,t_2 : \tau}$$

$$\frac{\Xi \mid \Gamma \vdash t_1 : \mathsf{int} \qquad \Xi \mid \Gamma \vdash t_2 : \mathsf{int}}{\Xi \mid \Gamma \vdash t_1 \pm t_2 : \mathsf{int}} \qquad \frac{}{\Xi \mid \Gamma \vdash () : 1}\ (\Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Gamma \vdash t_1 : \tau_1 \qquad \Xi \mid \Gamma \vdash t_2 : \tau_2}{\Xi \mid \Gamma \vdash (t_1, t_2) : \tau_1 \times \tau_2} \qquad \frac{\Xi \mid \Gamma \vdash t : 0}{\Xi \mid \Gamma \vdash \mathsf{void}\,t : \tau}\ (\Xi \vdash \tau)$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\,t : \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{snd}\,t : \tau_2}$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{inl}\,t : \tau_1 + \tau_2}\ (\Xi \vdash \tau_2) \qquad \frac{\Xi \mid \Gamma \vdash t : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{inr}\,t : \tau_1 + \tau_2}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Gamma \vdash t_0 : \tau_1 + \tau_2 \qquad \Xi \mid \Gamma, x_i : \tau_i \vdash t_i : \tau \quad (i = 1, 2)}{\Xi \mid \Gamma \vdash \mathsf{case}\,t_0\,x_1.t_1\,x_2.t_2 : \tau}$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau[\mu\alpha.\tau/\alpha]}{\Xi \mid \Gamma \vdash \mathsf{fold}\,t : \mu\alpha.\tau} \qquad \frac{\Xi \mid \Gamma \vdash t : \mu\alpha.\tau}{\Xi \mid \Gamma \vdash \mathsf{unfold}\,t : \tau[\mu\alpha.\tau/\alpha]}$$

$$\frac{\Xi, \alpha \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \Lambda\alpha.t : \forall\alpha.\tau}\ (\Xi \vdash \Gamma) \qquad \frac{\Xi \mid \Gamma \vdash t : \forall\alpha.\tau_0}{\Xi \mid \Gamma \vdash t\,[\tau_1] : \tau_0[\tau_1/\alpha]}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Gamma, x : \tau_0 \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \lambda x.t : \tau_0 \to \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t_1 : \tau \to \tau' \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash t_1\,t_2 : \tau'}$$

$$\frac{\Xi \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{fix}\,f.\lambda x.t : \tau_0 \to \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \mathsf{ref}\,t : \mathsf{ref}\,\tau}$$

$$\frac{\Xi \mid \Gamma \vdash t : \mathsf{ref}\,\tau}{\Xi \mid \Gamma \vdash !t : \tau} \qquad \frac{\Xi \mid \Gamma \vdash t_1 : \mathsf{ref}\,\tau \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash t_1 := t_2 : 1}$$

Fig. 1. Programming language

$$\frac{(\sigma, t_1) \Downarrow (\sigma', \lambda x.t) \quad (\sigma', t_2) \Downarrow (\sigma'', w_2) \quad (\sigma'', t[w_2/x]) \Downarrow (\sigma''', w)}{(\sigma,\ t_1\ t_2) \Downarrow (\sigma''', w)}$$

$$\frac{(\sigma, t_1) \Downarrow (\sigma', w) \quad w \text{ not of the form } \lambda x.t}{(\sigma,\ t_1\ t_2) \Downarrow \mathit{error}}$$

$$\frac{(\sigma, t) \Downarrow (\sigma', l) \quad l \in \mathrm{dom}(\sigma')}{(\sigma, !t) \Downarrow (\sigma', \sigma'(l))}$$

$$\frac{(\sigma, t) \Downarrow (\sigma', l) \quad l \notin \mathrm{dom}(\sigma')}{(\sigma, !t) \Downarrow \mathit{error}}$$

$$\frac{(\sigma, t_1) \Downarrow (\sigma', l) \quad (\sigma', t_2) \Downarrow (\sigma'', w) \quad l \in \mathrm{dom}(\sigma'')}{(\sigma,\ t_1 := t_2) \Downarrow (\sigma''[l \mapsto w], ())}$$

$$\frac{(\sigma, t) \Downarrow (\sigma', w) \quad l = \min\{l \in \omega \mid l \notin \mathrm{dom}(\sigma')\}}{(\sigma, \mathsf{ref}\ t) \Downarrow (\sigma' \uplus [l \mapsto w],\ l)}$$

Fig. 2. Operational semantics (selected rules)

of the form (l, n) for $n < \infty$ are its approximations. It is essential for the construction of the typed semantics in the next section that these "approximate locations" (l, n) are included. (See the remark before Proposition 3.2 below.)

3.1. *Domain-theoretic preliminaries*

We assume that the reader is familiar with basic denotational semantics, as presented for example in Winskel (1993), and with semantics in monadic style (Moggi 1991). Methods for solving recursive domain equations are used in a few of the proofs, but not elsewhere in the article. Familiarity with methods for proving the existence of invariant relations (Pitts 1996) should be useful, but is not assumed.

Let Cpo be the category of ω-cpos and ω-continuous functions. We use the standard notation for products, sums, and function spaces in Cpo. Injections into binary sums are written ι_1 and ι_2. For any set M and any cpo A, the cpo $M \rightharpoonup_{fin} A$ has maps from finite subsets of M to A as elements, and is ordered as follows: $f \sqsubseteq f'$ if and only if f and f' has the same domain M_0 and $f(m) \sqsubseteq f'(m)$ for all $m \in M_0$.

A complete, pointed partial order (cppo) is a cpo containing a least element. We use the notation $A_\perp = \{\lfloor a \rfloor \mid a \in A\} \cup \{\perp\}$ for the cppo obtained by "lifting" a cpo A. The least fixed-point of a continuous function $f : D \to D$ from a cppo D to itself is written $\mathit{fix}\, f$. The cppo of strict, continuous functions from a cpo A to a cppo D is written $A \multimap D$.

We shall also need to work with partial, continuous functions; these will be represented using the Kleisli category for the lifting monad $(-)_\perp$. Let pCpo be the Kleisli category for the lifting monad: objects are cpos, while morphisms from A to B are continuous functions from A to $B_\perp$. The identity maps in pCpo are written $\overline{id}$; they are given by

lifting: $\overline{id} = \lambda a.\lfloor a \rfloor$. Composition in pCpo is written $\mathbin{\overline{\circ}}$:

$$f \mathbin{\overline{\circ}} g = \lambda a. \begin{cases} f\, b, & \text{if } g\, a = \lfloor b \rfloor, \\ \bot, & \text{otherwise.} \end{cases}$$

The semantics below is presented in monadic style (Moggi 1991), i.e., structured using a monad that models the effects of the language. More specifically, we use a continuation-and-state monad (Benton and Leperchey 2005). It is most convenient to define this monad by means of a Kleisli triple: for every cpo S and every cppo Ans, the continuation-and-state monad $T_{S,Ans} : \mathsf{Cpo} \to \mathsf{Cpo}$ over S and Ans is given by

$$\begin{aligned} T_{S,Ans}\, A &= (A \to S \to Ans) \to S \to Ans \\ \eta_A\, a &= \lambda k.\lambda s.\, k\, a\, s \\ c \;\star_{A,B}\; f &= \lambda k.\lambda s.\, c\, (\lambda a.\lambda s'.f\, a\, k\, s')\, s \end{aligned}$$

where $\eta_A : A \to T_{S,Ans}A$ and $\star_{A,B} : T_{S,Ans}A \to (A \to T_{S,Ans}B) \to T_{S,Ans}B$. In the following we omit the type subscripts on η and $\star$. It is easy to verify that $(T_{S,Ans}, \eta, \star)$ satisfies the three monad laws:

$$\eta\, a \star f = f\, a \tag{3.1}$$

$$c \star \eta = c \tag{3.2}$$

$$(c \star f) \star g = c \star (\lambda a.\, f\, a \star g)\,. \tag{3.3}$$

Continuations are included for a technical reason, namely to ensure chain-completeness of the relations that will be used to model computations. (This will be made precise in Lemma 4.27 below.) These relations will be defined by "biorthogonality" (Benton and Hur 2009; Pitts 1998). Informally, computations are related if they map related continuations and related states to related answers, while continuations are in turn related if they map related values and related states to related answers. This approach ensures closure under limits of chains; see also Abadi (2000).

3.2. *Uniform cpos*

The standard methods for solving recursive domain equations give solutions that satisfy certain induction principles (Pitts 1996; Smyth and Plotkin 1982). One aspect of these induction principles is that, loosely speaking, one obtains as a solution not only a cpo A, but also a family of "projection" functions ϖ_n on A (one function for each $n \in \omega$) such that each element a of A is the limit of its projections $\varpi_0(a)$, $\varpi_1(a)$, etc. These functions therefore provide a handle for proving properties about A by induction on n.

Definition 3.1.

1. A *uniform cpo* $(A, (\varpi_n)_{n\in\omega})$ is a cpo A together with a family $(\varpi_n)_{n\in\omega}$ of continuous

functions from A to $A_\perp$, satisfying

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \cdots \sqsubseteq \varpi_n \sqsubseteq \ldots \tag{3.4}$$

$$\bigsqcup_{n\in\omega} \varpi_n = \overline{id_A} = \lambda a.\lfloor a \rfloor \tag{3.5}$$

$$\varpi_m \mathbin{\overline{\circ}} \varpi_n = \varpi_n \mathbin{\overline{\circ}} \varpi_m = \varpi_{\min(m,n)} \tag{3.6}$$

$$\varpi_0 = \lambda e.\, \bot\,. \tag{3.7}$$

2. A *uniform cppo* $(D, (\varpi_n)_{n\in\omega})$ is a cppo D together with a family $(\varpi_n)_{n\in\omega}$ of strict, continuous functions from D to itself, satisfying

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \cdots \sqsubseteq \varpi_n \sqsubseteq \ldots \tag{3.8}$$

$$\bigsqcup_{n\in\omega} \varpi_n = id_D \tag{3.9}$$

$$\varpi_m \circ \varpi_n = \varpi_n \circ \varpi_m = \varpi_{\min(m,n)} \tag{3.10}$$

$$\varpi_0 = \lambda e.\, \bot\,. \tag{3.11}$$

Remark. Uniform cppos are exactly the algebras for a certain monad on the category of cppos and strict, continuous functions. The monad is given by a natural monoid structure on $\overline{\omega}$: $T_{\mathrm{u}}D = \overline{\omega} \otimes D$, $\eta_{\mathrm{u}} e = (\infty, e)$, $\mu_{\mathrm{u}}(m,(n,e)) = (\min(m,n), e)$. Our locations are modeled using a free algebra for this monad: $Loc_\perp \cong \overline{\omega} \otimes (\mathbb{N}_0)_\perp$.

Uniform cppos are called *rank-ordered cpos* in earlier work by Baier and Majster-Cederbaum (1997).

3.3. *A universal uniform cpo*

We are now ready to construct a uniform cpo $(V, (\pi_n)_{n\in\omega})$ such that V is a suitable "universal" cpo. The exact requirements on the functions π_n are written down rather verbosely in the proposition below. This is not only convenient for proofs of properties about V: the functions π_n are also used in the definition of the untyped semantics. Intuitively, if one for example looks up the approximate location $(l,\, n+1)$ in a store s, one only obtains the approximate element $\pi_n(s(l))$ as a result.

More abstractly, in order to construct the typed interpretation in the next section we *need* the property that if one looks up the $n+1$-th approximation of location l in a store s, one only obtains $\pi_n(s(l))$ as a result. This is one reason for introducing approximated locations: the property would not hold if locations were modelled as standard integers.

Proposition 3.2. There exists a uniform cpo $(V, (\pi_n)_{n\in\omega})$ satisfying the following two properties:

1. The following isomorphism holds in Cpo:

$$V \;\cong\; \mathbb{Z} + Loc + 1 + (V \times V) + (V + V) + V \\ + T_{S,Ans}V + (V \to T_{S,Ans}V) \tag{3.12}$$

where

$$\begin{aligned} T_{S,Ans}V &= (V \to S \to Ans) \to S \to Ans \\ S &= \mathbb{N}_0 \rightharpoonup_{fin} V \\ Ans &= (\mathbb{Z} + Err)_\perp \end{aligned}$$

and

$$\begin{aligned} Loc &= \mathbb{N}_0 \times \overline{\omega} \\ Err &= 1\,. \end{aligned}$$

2. Abbreviate $TV = T_{S,Ans}V$ and $K = V \to S \to Ans$. Define the following injection functions corresponding to the summands on the right-hand side of the isomorphism (3.12):

$$\begin{aligned} in_{\mathbb{Z}} &: \mathbb{Z} \to V & \qquad in_{+} &: V + V \to V \\ in_{Loc} &: Loc \to V & \qquad in_{\to} &: (V \to TV) \to V \\ in_{1} &: 1 \to V & \qquad in_{\mu} &: V \to V \\ in_{\times} &: V \times V \to V & \qquad in_{\forall} &: TV \to V \end{aligned}$$

   With that notation, the functions $\pi_n : V \to V_\perp$ satisfy (and are determined by) the equations shown in Figure 3.

These two properties determine V uniquely, up to isomorphism in Cpo.

Proof (sketch) One solves the predomain equation (3.12) as usual (Smyth and Plotkin 1982); this gives a uniform cpo $(V, (\varpi_n)_{n\in\omega})$ which is almost right, except that the values of the ϖ_n on locations are wrong:

$$\varpi_{n+1}(in_{Loc}\, p) = in_{Loc}\, p.$$

Now define the functions π_n (and π_n^T etc.) as in the proposition (by induction on n). All the requirements in the definition of a uniform cpo except the fact that $\sqcup_n \pi_n = \overline{id}$ are easy to show. To show that $\sqcup_n \pi_n = \overline{id}$, one first shows by induction on m that $\pi_n \mathbin{\overline{\circ}} \varpi_m = \varpi_m \mathbin{\overline{\circ}} \pi_n$ for all n, and that

$$(\sqcup_n \pi_n) \mathbin{\overline{\circ}} \varpi_m = \varpi_m\,.$$

The conclusion then follows from the fact that $\sqcup_m \varpi_m = \overline{id}$ since $(V, (\varpi)_{n\in\omega})$ is a uniform cpo. □

As for the choice of answer type Ans in the continuation-and-state monad, we include an explicit "error" answer in order to show later that well-typed programs do not give errors (Corollary 4.37).

From here on, let V and $(\pi_n)_{n\in\omega}$ be as in the proposition above. We furthermore use the abbreviations, notation for injections, etc. introduced in the proposition; in particular, $TV = (V \to S \to Ans) \to S \to Ans$. Additionally, abbreviate $\lambda_l = in_{Loc}(l, \infty)$ and $\lambda_l^n = in_{Loc}(l, n)$; recall that here we have $n \geq 1$. Let $error_{Ans} \in Ans$ be the "error

$$\pi_0 = \lambda v.\bot \tag{3.13}$$

$$\pi_{n+1}(in_{\mathbb{Z}}(m)) = \lfloor in_{\mathbb{Z}}(m) \rfloor \tag{3.14}$$

$$\pi_{n+1}(in_1(*)) = \lfloor in_1(*) \rfloor \tag{3.15}$$

$$\pi_{n+1}(in_{Loc}(l,\infty)) = \lfloor in_{Loc}(l, n+1) \rfloor \tag{3.16}$$

$$\pi_{n+1}(in_{Loc}(l,m)) = \lfloor in_{Loc}(l, \min(n+1, m)) \rfloor \tag{3.17}$$

$$\pi_{n+1}(in_{\times}(v_1,v_2)) = \begin{cases} \lfloor in_{\times}(v_1', v_2') \rfloor & \text{if } \pi_n\, v_1 = \lfloor v_1' \rfloor \text{ and } \pi_n\, v_2 = \lfloor v_2' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{3.18}$$

$$\pi_{n+1}(in_{+}(\iota_i\, v)) = \begin{cases} \lfloor in_{+}(\iota_i\, v') \rfloor & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \qquad (i = 1,2) \tag{3.19}$$

$$\pi_{n+1}(in_{\mu}\, v) = \begin{cases} \lfloor in_{\mu}\, v' \rfloor & \text{if } \pi_n v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{3.20}$$

$$\pi_{n+1}(in_{\forall}\, c) = \lfloor in_{\forall}(\pi_{n+1}^T\, c) \rfloor \tag{3.21}$$

$$\pi_{n+1}(in_{\to}\, f) = \left\lfloor in_{\to}\left(\lambda v.\begin{cases} \pi_{n+1}^T\,(f\, v') & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases}\right)\right\rfloor \tag{3.22}$$

Here the functions $\pi_n^S : S \to S_\bot$ and $\pi_n^K : K \to K$ and $\pi_n^T : TV \to TV$ are defined as follows:

$$\pi_0^S = \lambda s.\bot \qquad \pi_0^K = \lambda k.\bot \qquad \pi_0^T = \lambda c.\bot \tag{3.23}$$

$$\pi_{n+1}^S(s) = \begin{cases} \lfloor s' \rfloor & \text{if } \pi_n \circ s = \lambda l.\lfloor s'(l) \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{3.24}$$

$$\pi_{n+1}^K(k) = \lambda v.\lambda s.\begin{cases} k\, v'\, s' & \text{if } \pi_n\, v = \lfloor v' \rfloor \text{ and } \pi_{n+1}^S\, s = \lfloor s' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{3.25}$$

$$\pi_{n+1}^T(c) = \lambda k.\lambda s.\begin{cases} c\,(\pi_{n+1}^K\, k)\, s' & \text{if } \pi_{n+1}^S\, s = s' \\ \bot & \text{otherwise}\,. \end{cases} \tag{3.26}$$

Fig. 3. Characterization of the projection functions $\pi_n : V \to V_\bot$.

answer" and let $error \in TV$ be the "error computation":

$$error_{Ans} = \lfloor \iota_2 * \rfloor$$
$$error = \lambda k.\lambda s.\, error_{Ans}\,.$$

We shall later need:

Proposition 3.3.

1. $(S, (\pi_n^S)_{n\in\omega})$ is a uniform cpo.
2. $(K, (\pi_n^K)_{n\in\omega})$ and $(TV, (\pi_n^T)_{n\in\omega})$ are uniform cppos.

In order to model the three operations of the untyped language that involve references, we define the three functions *alloc*, *lookup*, and *assign* in Figure 4.

$$\mathit{alloc} : V \to TV, \quad \mathit{lookup} : V \to TV, \quad \mathit{assign} : V \to V \to TV.$$

$$\mathit{alloc}\, v = \lambda k\, \lambda s.\, k\, (\lambda_{\mathit{free}(s)})\, (s[\mathit{free}(s) \mapsto v])$$
$$(\text{where } \mathit{free}(s) = \min\{n \in \mathbb{N}_0 \mid n \notin \mathrm{dom}(s)\})$$

$$\mathit{lookup}\, v = \lambda k\, \lambda s. \begin{cases} k\, s(l)\, s & \text{if } v = \lambda_l \text{ and } l \in \mathrm{dom}(s) \\ k\, v'\, s & \text{if } v = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n(s(l)) = \lfloor v' \rfloor \\ \bot_{Ans} & \text{if } v = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n(s(l)) = \bot \\ \mathit{error}_{Ans} & \text{otherwise} \end{cases}$$

$$\mathit{assign}\, v_1\, v_2 = \lambda k\, \lambda s. \begin{cases} k\, (\mathit{in}_1 *)\, (s[l \mapsto v_2]) & \text{if } v_1 = \lambda_l \text{ and } l \in \mathrm{dom}(s) \\ k\, (\mathit{in}_1 *)\, (s[l \mapsto v_2']) & \text{if } v_1 = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n(v_2) = \lfloor v_2' \rfloor \\ \bot_{Ans} & \text{if } v_1 = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n(v_2) = \bot \\ \mathit{error}_{Ans} & \text{otherwise} \end{cases}$$

Fig. 4. Functions used for interpreting reference operations.

Lemma 3.4. The functions *alloc*, *lookup*, and *assign* are continuous.

Notice that it would not suffice to define, e.g., $\mathit{lookup}(\lambda_l^{n+1})(k)(s) = \bot$ for $l \in \mathrm{dom}(s)$, and hence avoid mentioning the projection functions: *lookup* would then not be continuous.

We are now ready to define the untyped semantics.

Definition 3.5. Let t be a term and let X be a set of variables such that $\mathrm{FV}(t) \subseteq X$. The *untyped semantics of t with respect to X* is the continuous function $[\![t]\!]_X : V^X \to TV$ defined by induction on t in Figure 5.

The semantics of a complete program, i.e., a term with no free term variables or type variables, is defined by supplying an initial continuation and the empty store:

Definition 3.6. Let t be a term with no free term variables or type variables. The *program semantics of t* is the element $[\![t]\!]^{\mathrm{p}}$ of Ans defined by

$$[\![t]\!]^{\mathrm{p}} = [\![t]\!]_\emptyset \emptyset\, k_{init}\, s_{init}$$

where

$$k_{init} = \lambda v.\lambda s. \begin{cases} \lfloor \iota_1\, m \rfloor & \text{if } v = \mathit{in}_{\mathbb{Z}}(m) \\ \mathit{error}_{Ans} & \text{otherwise} \end{cases}$$

and where $s_{init} \in S$ is the empty store.

For every t with $\mathrm{FV}(t) \subseteq X$, define the continuous $[\![t]\!]_X : V^X \to TV$ by induction on t:

$$
\begin{aligned}
[\![x]\!]_X\rho &= \eta(\rho(x)) \\
[\![\overline{m}]\!]_X\rho &= \eta(in_{\mathbb{Z}}\, m) \\
[\![\mathsf{ifz}\, t_0\, t_1\, t_2]\!]_X\rho &= [\![t_0]\!]_X\rho \star \lambda v_0. \begin{cases} [\![t_1]\!]_X\rho & \text{if } v_0 = in_{\mathbb{Z}}\, 0 \\ [\![t_2]\!]_X\rho & \text{if } v_0 = in_{\mathbb{Z}}\, m \text{ where } m \neq 0 \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![t_1 \pm t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2. \begin{cases} \eta(in_{\mathbb{Z}}(m_1 \pm m_2)) \\ \qquad \text{if } v_1 = in_{\mathbb{Z}}\, m_1 \\ \qquad \text{and } v_2 = in_{\mathbb{Z}}\, m_2 \\ \mathit{error} \quad \text{otherwise} \end{cases} \\
[\![()]\!]_X\rho &= \eta(in_1\, *) \\
[\![(t_1,\, t_2)]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2.\, \eta(in_{\times}(v_1, v_2)) \\
[\![\mathsf{fst}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_1) & \text{if } v = in_{\times}(v_1, v_2) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{snd}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_2) & \text{if } v = in_{\times}(v_1, v_2) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{void}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \mathit{error} \\
[\![\mathsf{inl}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{+}(\iota_1\, v)) \\
[\![\mathsf{inr}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{+}(\iota_2\, v)) \\
[\![\mathsf{case}\, t_0\ x_1.t_1\ x_2.t_2]\!]_X\rho &= [\![t_0]\!]_X\rho \star \lambda v_0. \begin{cases} [\![t_1]\!]_{X,x_1}(\rho[x_1 \mapsto v]) & \text{if } v_0 = in_{+}(\iota_1\, v) \\ [\![t_2]\!]_{X,x_2}(\rho[x_2 \mapsto v]) & \text{if } v_0 = in_{+}(\iota_2\, v) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\lambda x.t]\!]_X\rho &= \eta(in_{\to}(\lambda v.\, [\![t]\!]_{X,x}(\rho[x \mapsto v]))) \\
[\![t_1\, t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2. \begin{cases} g\, v_2 & \text{if } v_1 = in_{\to}\, g \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{fix}\, f.\lambda x.t]\!]_X\rho &= \eta(in_{\to}(\mathit{fix}(\lambda g^{V \to TV}.\, \lambda v.\, [\![t]\!]_{X,f,x}(\rho[f \mapsto in_{\to}\, g,\, x \mapsto v])))) \\
[\![\mathsf{fold}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{\mu}\, v) \\
[\![\mathsf{unfold}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_0) & \text{if } v = in_{\mu}\, v_0 \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\Lambda\alpha.t]\!]_X\rho &= \eta(in_{\forall}\, ([\![t]\!]_X\rho)) \\
[\![t\, [\tau]]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} c & \text{if } v = in_{\forall}\, c \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{ref}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \mathit{alloc}\, v \\
[\![!\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \mathit{lookup}\, v \\
[\![t_1 := t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2.\, \mathit{assign}\, v_1\, v_2
\end{aligned}
$$

Fig. 5. Untyped semantics of terms.

3.4. *Soundness and adequacy of the untyped semantics*

In order to formulate soundness and adequacy of the denotational semantics with respect to the operational semantics, we extend the denotational semantics to location constants:

$$[\![l]\!]_X = \eta(\lambda_l).$$

That is, the meaning of a location constant is the corresponding *ideal* location. The meaning of a syntactic store is given pointwise: $[\![\sigma]\!] = \lambda l \in \mathrm{dom}(\sigma).[\![\sigma(l)]\!]_\emptyset$.

We then have soundness:

1. If $(\sigma, t) \Downarrow (\sigma', w)$, then for all continuations k we have $[\![t]\!]\, k\, [\![\sigma]\!] = k\, v\, [\![\sigma']\!]$ where $[\![w]\!] = \eta(v)$.
2. If $(\sigma, t) \Downarrow \mathit{error}$, then for all continuations k we have $[\![t]\!]\, k\, [\![\sigma]\!] = \mathit{err}_{Ans}$.

This is shown in the usual way, by induction on evaluation derivations. Notice that no approximate locations occur in the soundness proof since location constants are interpreted as ideal locations.

Computational adequacy can be stated as follows:

1. If $[\![t]\!]^{\mathrm{p}} = \lfloor \iota_1(m) \rfloor$, then $(\emptyset, t) \Downarrow (\sigma, \overline{m})$ for some σ.
2. If $[\![t]\!]^{\mathrm{p}} = \mathit{error}_{Ans}$, then $(\emptyset, t) \Downarrow \mathit{error}$.

It follows easily from the combination of soundness and adequacy that (possibly open) terms with the same denotation are contextually equivalent.

We expect that computational adequacy can be shown using the standard technique (Pitts 1996) also for our non-standard semantics of locations, lookup and assignment. We have earlier shown a computational-adequacy result for a similar untyped semantics that also contains approximate locations (Birkedal et al. 2009).

4. Typed semantics

In this section we present a "typed semantics", i.e., an interpretation of types and typed terms. As described in the introduction, types will be interpreted as world-indexed families of binary relations on the universal cpo V. Since worlds depend on semantic types, the space of semantic types is obtained by solving a recursive metric-space equation, i.e., by finding a fixed-point of a functor on metric spaces.

The rest of this section is structured as follows. Section 4.1 presents the necessary material on metric spaces. In Section 4.2 we construct an appropriate space of semantic types. Then, in Section 4.3, we interpret each type of the language as a semantic type. Based on that interpretation of types, we introduce a notion of semantic relatedness of typed terms in Section 4.4. We then show that all the term constructs of the language respect semantic relatedness; as a corollary, we have a "fundamental lemma" stating that every well-typed term is semantically related to itself. It follows that well-typed terms do not denote "error". More interestingly, well-typed terms of polymorphic type satisfy a relational parametricity principle. In fact, *all* well-typed terms satisfy a relational parametricity principle involving the store: this principle results from Kripke-style quantification over all future "semantic store typings".

The reader is assumed to be familiar with basic properties of metric spaces (Smyth 1992), although the relevant definitions are repeated below.

4.1. *Ultrametric spaces*

Let $\mathbb{R}^+$ be the set of non-negative real numbers.

Definition 4.1. A *metric space* (X, d) is a set X together with a function $d : X \times X \to \mathbb{R}^+$ satisfying the following three conditions:

1. $d(x, y) = 0 \iff x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$.

An *ultrametric space* is a metric space (X, d) that satisfies the stronger *ultrametric inequality* instead of (iii):

(iii') $d(x, z) \leq \max(d(x, y), d(y, z))$.

A metric space (X, d) is 1-*bounded* if $d(x, y) \leq 1$ for all x and y in X.

By a *sequence* in a metric space (X, d) we mean an ω-indexed sequence $(x_n)_{n\in\omega}$ of elements of X.

Definition 4.2.

1. A *Cauchy sequence* in a metric space (X, d) is a sequence $(x_n)_{n\in\omega}$ of elements of X such that for all $\epsilon > 0$, there exists an $N \in \omega$ such that $d(x_m, x_n) < \epsilon$ for all $m, n \geq N$.
2. A *limit* of a sequence $(x_n)_{n\in\omega}$ in a metric space (X, d) is an element x of X such that for all $\epsilon > 0$, there exists an $N \in \omega$ such that $d(x_n, x) < \epsilon$ for all $n \geq N$.
3. A *complete metric space* is a metric space in which every Cauchy sequence has a limit.

In the following we shall consider complete, 1-bounded ultrametric spaces. As a canonical example of such a metric space, consider the set $\mathbb{N}^\omega$ of infinite sequences of natural numbers, with distance function d given by:

$$d(x, y) = \begin{cases} 2^{-\max\{n\in\omega \mid \forall m\leq n.\, x(m)=y(m)\}} & \text{if } x \neq y \\ 0 & \text{if } x = y. \end{cases}$$

To avoid confusion, call the elements of $\mathbb{N}^\omega$ *strings* instead of sequences. Here the ultrametric inequality simply states that if x and y agree on the first n "characters" and y and z also agree on the first n characters, then x and z agree on the first n characters. A Cauchy sequence in $\mathbb{N}^\omega$ is a sequence of strings $(x_n)_{n\in\omega}$ in which the individual characters "stabilize": for all m, there exists $N \in \omega$ such that $x_{n_1}(m) = x_{n_2}(m)$ for all $n_1, n_2 \geq N$. In other words, there is a number k such that $x_n(m) = k$ for almost all n, i.e., all but finitely many n. The limit of the sequence $(x_n)_{n\in\omega}$ is therefore the string x defined by

$$x(m) = k \quad \text{where } x_n(m) = k \text{ for almost all } n.$$

As illustrated by the above example, it might be helpful to think of the function d of a complete, 1-bounded ultrametric space (X, d) not as a measure of (euclidean) distance between elements, but rather as a measure of the degree of similarity between elements.

Definition 4.3.

1. A function $f : X_1 \to X_2$ from a metric space (X_1, d_1) to a metric space (X_2, d_2) is *non-expansive* if $d_2(f(x), f(y)) \leq d_1(x, y)$ for all x and y in X_1.
2. A function $f : X_1 \to X_2$ from a metric space (X_1, d_1) to a metric space (X_2, d_2) is *contractive* if there exists $c < 1$ such that $d_2(f(x), f(y)) \leq c \cdot d_1(x, y)$ for all x and y in X_1.

Let CBUlt be the category with complete, 1-bounded ultrametric spaces as objects and non-expansive functions as morphisms. This category is cartesian closed (Wagner 1994). Products are defined in the natural way: $(X_1, d_1) \times (X_2, d_2) = (X_1 \times X_2, d_{X_1 \times X_2})$ where

$$d_{X_1 \times X_2}((x_1, x_2), (y_1, y_2)) = \max(d_1(x_1, y_1), d_2(x_2, y_2)) \,.$$

The exponential $(X_1, d_1) \to (X_2, d_2)$ has the set of non-expansive maps from (X_1, d_1) to (X_2, d_2) as the underlying set, and the "sup"-metric $d_{X_1 \to X_2}$ as distance function:

$$d_{X_1 \to X_2}(f, g) = \sup\{d_2(f(x), g(x)) \mid x \in X_1\} \,.$$

For both products and exponentials, limits are pointwise.

Note that the category of (not necessarily ultra-) metric spaces and non-expansive maps is not cartesian closed: the ultrametric inequality is required in order for the evaluation maps (corresponding to the exponentials) to be non-expansive (Wagner 1994).

If X_0 is a subset of the underlying set X of a metric space (X, d), then the restriction $d_0 = d|_{X_0 \times X_0}$ of d turns (X_0, d_0) into a metric space. If X_0 is closed, then (X_0, d_0) is complete:

Definition 4.4. Let (X, d) be a metric space. A subset X_0 of X is *closed* (with respect to d) if whenever $(x_n)_{n \in \omega}$ is a sequence of elements of X_0 with limit x, the limit element x belongs to X_0.

Proposition 4.5. Let (X, d) be a complete, 1-bounded ultrametric space, and let X_0 be a closed subset of X. The restriction $d_0 = d|_{X_0 \times X_0}$ of d turns (X_0, d_0) into a complete, 1-bounded ultrametric space.

4.1.1. *Banach's fixed-point theorem* We need the following classical result:

Theorem 4.6 (Banach's fixed-point theorem). Let (X, d) be a non-empty, complete metric space, and let f be a contractive function from (X, d) to itself. There exists a unique fixed-point of f, i.e., a unique element x of X such that $f(x) = x$.

For a given complete metric space, consider the function *fix* that maps every contractive operator to its unique fixed-point. On complete ultrametric spaces, *fix* is non-expansive in the following sense (Amadio 1991):

Proposition 4.7. Let (X, d) be a non-empty, complete ultrametric space. For all contractive functions f and g from (X, d) to itself, $d(\mathit{fix}\ f, \mathit{fix}\ g) \leq d(f, g)$.

Proof. Let $c < 1$ be a non-negative number such that $d(f(x), f(y)) \leq c \cdot d(x, y)$ for all x and y in X. Now let $x = \mathit{fix}\ f$ and $y = \mathit{fix}\ g$. By the ultrametric inequality,

$$\begin{aligned} d(x, y) &= d(f(x), g(y)) \\ &\leq \max(d(f(x), f(y)), d(f(y), g(y)) \\ &\leq \max(d(f(x), f(y)), d(f, g)) \\ &\leq \max(c \cdot d(x, y), d(f, g))\,. \end{aligned}$$

If $\max(c \cdot d(x, y), d(f, g)) = c \cdot d(x, y)$ we have $d(x, y) \leq c \cdot d(x, y)$, and hence $d(x, y) = 0 \leq d(f, g)$. Otherwise, $\max(c \cdot d(x, y), d(f, g)) = d(f, g)$, and hence $d(x, y) \leq d(f, g)$. □

4.1.2. *Solving recursive metric-space equations* The inverse-limit method for solving recursive domain equations can be adapted from Cpo to CBUlt (America and Rutten 1989). We sketch an account that suffices for this article.

In CBUlt, one finds fixed points of *locally contractive* functors instead of locally continuous functors.

Definition 4.8.

1. A functor $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ is *locally non-expansive* if
$$d(F(f, g), F(f', g')) \leq \max(d(f, f'), d(g, g'))$$
for all non-expansive functions f, f', g, and g'.
2. A functor $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ is *locally contractive* if there exists $c < 1$ such that
$$d(F(f, g), F(f', g')) \leq c \cdot \max(d(f, f'), d(g, g'))$$
for all non-expansive functions f, f', g, and g'.

One can obtain a locally contractive functor from a locally non-expansive one by multiplying with a "shrinking" factor (America and Rutten 1989):

Proposition 4.9. Let $0 < c < 1$.

1. Let $(X, d) \in \mathsf{CBUlt}$, and define $c \cdot (X, d) = (X, c \cdot d)$ where $c \cdot d : X \times X \to \mathbb{R}^+$ is given by $(c \cdot d)(x, y) = c \cdot d(x, y)$. We have $c \cdot (X, d) \in \mathsf{CBUlt}$.
2. Let $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ be a locally non-expansive functor. The functor $c \cdot F$ given by
$$\begin{aligned} (c \cdot F)((X_1, d_1), (X_2, d_2)) &= c \cdot F((X_1, d_1), (X_2, d_2)) \\ (c \cdot F)(f, g) &= F(f, g) \end{aligned}$$
is locally contractive.

The main theorem about existence and uniqueness of fixed points of locally contractive functors is actually most conveniently phrased in terms of the category of *non-empty*, complete, 1-bounded ultrametric spaces. The reason is the essential use of Banach's fixed-point theorem in the proof. Rather than considering this subcategory, we impose a technical requirement on the given mixed-variance functor F on CBUlt, namely that

$F(1,1) \neq \emptyset$ where 1 is the one-point metric space. It is not hard to see that this requirement holds if and only if F restricts to the full subcategory of non-empty metric spaces.

By a well-known adaptation of the inverse-limit method, in the style of America and Rutten (1989), one can then show:

Theorem 4.10. Let $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ be a locally contractive functor satisfying that $F(1,1) \neq \emptyset$. There exists a unique (up to isomorphism) non-empty $(X,d) \in \mathsf{CBUlt}$ such that $F((X,d),(X,d)) \cong (X,d)$.

4.2. *The space of semantic types*

The space of semantic types is obtained by applying Theorem 4.10 above to a functor that maps metric spaces to world-indexed binary relations on V. First, some standard definitions:

Definition 4.11. For every cpo A, let $Rel(A)$ be the set of binary relations $R \subseteq A \times A$ on A.

1. A relation $R \in Rel(A)$ is *complete* if for all chains $(a_n)_{n\in\omega}$ and $(a'_n)_{n\in\omega}$ such that $(a_n, a'_n) \in R$ for all n, also $(\sqcup_{n\in\omega} a_n, \sqcup_{n\in\omega} a'_n) \in R$. Let $CRel(A)$ be the set of complete relations on A.
2. A relation $R \in Rel(D)$ on a cppo D is *pointed* if $(\bot,\bot) \in R$ and *admissible* if it is pointed and complete. Let $ARel(D)$ be the set of admissible relations on D.
3. For every cpo A and every relation $R \in Rel(A)$, define the relation $R_\bot \in Rel(A_\bot)$ by $R_\bot = \{\,(\bot,\bot)\,\} \cup \{\,(\lfloor a\rfloor, \lfloor a'\rfloor) \mid (a,a') \in R\,\}$.
4. For $R \in Rel(A)$ and $S \in Rel(B)$, let $R \to S$ be the set of continuous functions f from A to B satisfying that for all $(a,a') \in R$, $(f\,a, f\,a') \in S$.

On uniform cpos and uniform cppos, we furthermore define the set of *uniform* binary relations (Abadi and Plotkin 1990; Amadio 1991). The key point is that a uniform and complete relation on a uniform cppo $(D,(\varpi_n)_{n\in\omega})$ is completely determined by its elements of the form $(\varpi_n\,e, \varpi_n\,e')$.

Definition 4.12.

1. Let $(A,(\varpi_n)_{n\in\omega})$ be a uniform cpo. A relation $R \in Rel(A)$ is *uniform with respect to* $(\varpi_n)_{n\in\omega}$ if $\varpi_n \in R \to R_\bot$ for all n. Let $CURel(A,(\varpi_n)_{n\in\omega})$ be the set of binary relations on A that are uniform with respect to $(\varpi_n)_{n\in\omega}$ and complete.
2. Let $(D,(\varpi_n)_{n\in\omega})$ be a uniform cppo. A relation $R \in Rel(D)$ is *uniform with respect to* $(\varpi_n)_{n\in\omega}$ if $\varpi_n \in R \to R$ for all n. Let $AURel(D,(\varpi_n)_{n\in\omega})$ be the set of binary relations on D that are uniform with respect to $(\varpi_n)_{n\in\omega}$ and admissible.

Proposition 4.13. Let $(D,(\varpi_n)_{n\in\omega})$ be a uniform cppo, and assume that $R,S \in AURel(D,(\varpi_n)_{n\in\omega})$.

1. If $\varpi_n \in R \to S$, then $\varpi_{n'} \in R \to S$ for all $n' \leq n$.
2. If $\varpi_n \in R \to S$ for all n, then $R \subseteq S$.

We now define a number of metric spaces that will be used in constructing the universe of semantic types. After defining one of these metric spaces (X, d), the "distance function" d will be fixed, so we usually omit it and call X itself a metric space.

First, as in Amadio (1991), we obtain:

Proposition 4.14. Let $(D, (\varpi_n)_{n\in\omega})$ be a uniform cppo. Then $AURel(D, (\varpi_n)_{n\in\omega})$ is a complete, 1-bounded ultrametric space with the distance function given by

$$d(R,S) = \begin{cases} 2^{-\max\{\, n\in\omega \mid \varpi_n\in R\to S \,\wedge\, \varpi_n\in S\to R\,\}} & \text{if } R \neq S \\ 0 & \text{if } R = S. \end{cases}$$

Proof. First we show that the function d is well-defined: if $R \neq S$, then there exists a greatest n in ω such that $\varpi_n \in R \to S$ and $\varpi_n \in S \to R$. Assume that $R \neq S$. By (3.11) we always have $\varpi_0 \in R \to S$ and $\varpi_0 \in S \to R$, so there is at least one such n. Now assume that there are infinitely many such n; then Proposition 4.13 implies that $R \subseteq S$ and $S \subseteq R$, i.e., that $R = S$, a contradiction.

Proposition 4.13(1) implies the following property, which we shall need below:

$$d(R,S) \leq 2^{-n} \text{ if and only if } \varpi_n \in R \to S \text{ and } \varpi_n \in S \to R. \tag{4.1}$$

It is easy to see that the function d defines a 1-bounded ultrametric. To see that it is complete, let $(R_m)_{m\in\omega}$ be a Cauchy sequence. Then for all n there exists a number M_n such that $d(R_m, R_{m'}) \leq 2^{-n}$ for all $m, m' \geq M_n$. For all $m, m' \geq M_n$, (4.1) then implies that $\varpi_n \in R_m \to R_{m'}$. Therefore, for all $e, e' \in D$,

$$\begin{aligned} (\varpi_n\, e, \varpi_n\, e') \in R_m &\implies ((\varpi_n \circ \varpi_n)\, e, (\varpi_n \circ \varpi_n)\, e') \in R_{m'} && \text{(by definition of } d) \\ &\implies (\varpi_n\, e, \varpi_n\, e') \in R_{m'}\,, && \text{(by (3.10))} \end{aligned}$$

and the other way around by symmetry. This means that the set of related elements of the form $(\varpi_n\, e, \varpi_n\, e')$ is the same in the relations R_{M_n}, R_{M_n+1}, etc. Now define the relation R by

$$(e, e') \in R \iff \text{for all } n,\ (\varpi_n\, e,\ \varpi_n\, e') \in R_{M_n}.$$

We first show that R is admissible and uniform, and then that R is the limit of $(R_m)_{m\in\omega}$. First, R is pointed by (3.11) and the fact that each R_n is pointed. R is complete since it is an intersection of inverse images of the continuous functions ϖ_n with respect to the complete relations R_{M_n}. R is also uniform: let $(e, e') \in R$; then for all m and n, uniformity of R_{M_n} and (3.10) imply that $(\varpi_n(\varpi_m\, e), \varpi_n(\varpi_m\, e')) = (\varpi_m(\varpi_n\, e), \varpi_m(\varpi_n\, e')) \in R_{M_n}$, and hence $(\varpi_m\, e, \varpi_m\, e') \in R$.

It remains to show that R is the limit of $(R_m)_{m\in\omega}$. It suffices to show: for all n and all $m \geq M_n$,

$$\varpi_n \in R \to R_m \quad \text{and} \quad \varpi_n \in R_m \to R\,.$$

First, let $(e, e') \in R$. Then $(\varpi_n\, e,\ \varpi_n\, e') \in R_{M_n}$ by definition on R, and hence

$$(\varpi_n\, e,\ \varpi_n\, e') \in R_m$$

since $m \geq M_n$. Second, let $(e, e') \in R_m$. By uniformity of R_m also $(\varpi_n\, e, \varpi_n\, e') \in R_m$. But then $(\varpi_n\, e, \varpi_n\, e')$ belongs to R_{M_n} since $m \geq M_n$. It then follows easily from (3.10) and the definition of R that $(\varpi_n\, e, \varpi_n\, e') \in R$. □

Proposition 4.15. Let (X, d) be a complete, 1-bounded ultrametric space. The set $\mathbb{N}_0 \rightharpoonup_{fin} X$ of finite maps from natural numbers to elements of X is a complete, 1-bounded ultrametric space with the distance function given by

$$d'(\Delta, \Delta') = \begin{cases} \max\{d(\Delta(l), \Delta'(l)) \mid l \in \mathrm{dom}(\Delta)\} & \text{if } \mathrm{dom}(\Delta) = \mathrm{dom}(\Delta') \\ 1 & \text{otherwise.} \end{cases}$$

Proof (sketch) Standard. CBUlt has all products and sums. Then, the set $\mathbb{N}_0 \rightharpoonup_{fin} X$ can be viewed as a sum of products: $\sum_{L \subseteq_{fin} \mathbb{N}_0} X^L$ and the distance function above reflects that fact. In general, two elements of different summands are given the maximal possible distance 1. □

Definition 4.16. For every $(X, d) \in \mathit{CBUlt}$, define an "extension" ordering $\leq$ on the set $\mathbb{N}_0 \rightharpoonup_{fin} X$ by

$$\Delta \leq \Delta' \iff \mathrm{dom}(\Delta) \subseteq \mathrm{dom}(\Delta') \wedge \forall l \in \mathrm{dom}(\Delta).\, \Delta(l) = \Delta'(l)\,.$$

Proposition 4.17. Let $(X, d) \in \mathit{CBUlt}$, let $(D, (\varpi_n)_{n\in\omega})$ be a uniform cppo, and let

$$(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} \mathit{AURel}(D, (\varpi_n)_{n\in\omega})$$

be the set of functions ν from $\mathbb{N}_0 \rightharpoonup_{fin} X$ to $\mathit{AURel}(D, (\varpi_n)_{n\in\omega})$ that are both non-expansive and monotone in the sense that $\Delta \leq \Delta'$ implies $\nu(\Delta) \subseteq \nu(\Delta')$. This set is a complete, 1-bounded ultrametric space with the "sup"-metric, given by

$$d'(\nu, \nu') = \sup\{d(\nu(\Delta), \nu'(\Delta)) \mid \Delta \in \mathbb{N}_0 \rightharpoonup_{fin} X\}\,.$$

Proof. The set $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} \mathit{AURel}(D, (\varpi_n)_{n\in\omega})$ is a subset of the underlying set of the exponential $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to \mathit{AURel}(D, (\varpi_n)_{n\in\omega})$ in CBUlt, namely the subset of monotone as well as non-expansive functions, and the distance function d defined above is the same as for the larger set. By Proposition 4.5 it therefore suffices to show that the set of monotone and non-expansive functions is a closed subset of the (complete) metric space of all non-expansive functions.

Let $(\nu_m)_{m\in\omega}$ be a sequence of monotone and non-expansive functions from $(\mathbb{N}_0 \rightharpoonup_{fin} X)$ to $\mathit{AURel}(D, (\varpi_n)_{n\in\omega})$ with limit ν (for some function ν which is non-expansive). We must show that ν is monotone. To that end, let Δ and Δ' be elements of $\mathbb{N}_0 \rightharpoonup_{fin} X$ such that $\Delta \leq \Delta'$; we must show that $\nu(\Delta) \subseteq \nu(\Delta')$. By Proposition 4.13(2) it suffices to show that $\varpi_n \in \nu(\Delta) \to \nu(\Delta')$ for all n. So let n be given. Since $(\nu_m)_{m\in\omega}$ has limit ν, there exists an m such that $d(\nu, \nu_m) \leq 2^{-n}$. By definition of the metric on exponentials, this implies that $d(\nu(\Delta), \nu_m(\Delta)) \leq 2^{-n}$, and hence that $\varpi_n \in \nu(\Delta) \to \nu_m(\Delta)$ by Proposition 4.13(1). But ν_m is assumed to be monotone, so $\nu_m(\Delta) \subseteq \nu_m(\Delta')$ and therefore $\varpi_n \in \nu(\Delta) \to \nu_m(\Delta')$. Since also $d(\nu(\Delta'), \nu_m(\Delta')) \leq 2^{-n}$, we have $\varpi_n \in \nu_m(\Delta') \to \nu(\Delta')$, and conclude by (3.10) that $\varpi_n \in \nu(\Delta) \to \nu(\Delta')$. □

Propositions 4.14 and 4.17 and a little extra work give analogous results for uniform cpos:

Proposition 4.18. Let $(A, (\varpi_n)_{n\in\omega})$ be a uniform cpo. Below, abbreviate $\mathit{CURel}(A) = \mathit{CURel}(A, (\varpi_n)_{n\in\omega})$.

1. The set $\mathit{CURel}(A)$ is a complete, 1-bounded ultrametric space with the distance function given by
$$d(R,S) = \begin{cases} 2^{-\max\{\, n\in\omega \mid \varpi_n\in R\to S_\perp \,\wedge\, \varpi_n\in S\to R_\perp \,\}} & \text{if } R \neq S \\ 0 & \text{if } R = S. \end{cases}$$
2. Let $(X,d) \in \mathit{CBUlt}$, and let $(\mathbb{N}_0 \rightharpoonup_{\mathit{fin}} X) \to_{\mathit{mon}} \mathit{CURel}(A)$ be the set of functions ν from $\mathbb{N}_0 \rightharpoonup_{\mathit{fin}} X$ to $\mathit{CURel}(A)$ that are both non-expansive and monotone in the sense that $\Delta \leq \Delta'$ implies $\nu(\Delta) \subseteq \nu(\Delta')$. This set is a complete, 1-bounded ultrametric space with the "sup"-metric, given by
$$d'(\nu,\nu') = \sup\{d(\nu(\Delta),\nu'(\Delta)) \mid \Delta \in \mathbb{N}_0 \rightharpoonup_{\mathit{fin}} D\}.$$

Proof.

1: It is easy to see that the family of strict extensions $\varpi_n^\dagger : A_\perp \to A_\perp$ of the projection functions $\varpi_n : A \to A_\perp$ turns $(A_\perp, (\varpi_n^\dagger)_{n\in\omega})$ into a uniform cppo. Abbreviate $\mathit{AURel}(A_\perp) = \mathit{AURel}(A_\perp, (\varpi_n^\dagger)_{n\in\omega})$. By definition of uniform relations,
$$R \in \mathit{CURel}(A) \quad \text{if and only if} \quad R_\perp \in \mathit{AURel}(A_\perp) \tag{4.2}$$
for all R in $\mathit{Rel}(A)$. Furthermore, Proposition 4.14 gives a metric on the set $\mathit{AURel}(A_\perp)$, and it is easy to see that the distance function on $\mathit{CURel}(A)$ defined in Part 1 above is induced by the lifting operator, i.e., $d(R,S) = d(R_\perp, S_\perp)$. Since the lifting operator is injective, this induced distance function turns $\mathit{CURel}(A)$ into a 1-bounded ultrametric space.

However, not every S in $\mathit{AURel}(A_\perp)$ has the form $R_\perp$ for some R in $\mathit{CURel}(A)$: unless A is empty, some relations in $\mathit{AURel}(A_\perp)$ relate $\perp$ to elements different from $\perp$. In other words, the lifting operator from $\mathit{CURel}(A)$ to $\mathit{AURel}(A_\perp)$ is not surjective. Therefore, completeness of $\mathit{AURel}(A_\perp)$ does not immediately imply completeness of $\mathit{CURel}(A)$. What we need to show is that the subset of $\mathit{AURel}(A_\perp)$ consisting of *strict* relations, i.e., relations S for which $(a,\perp) \in S$ or $(\perp,a) \in S$ implies $a = \perp$, is a closed subset of $\mathit{AURel}(A_\perp)$. Proposition 4.5 then implies that the subset of strict relations is a complete metric space, and (4.2) implies that it is isomorphic to $\mathit{CURel}(A)$, which is therefore also complete.

More generally, let $(D, (\varpi'_n)_{n\in\omega})$ be a uniform cppo, and abbreviate $\mathit{AURel}(D) = \mathit{AURel}(D, (\varpi'_n)_{n\in\omega})$; we show that the subset $\mathit{SAURel}(D) \subseteq \mathit{AURel}(D)$ of strict relations is closed. So let $(R_m)_{m\in\omega}$ be a sequence of strict relations (elements of $\mathit{SAURel}(D)$) with limit R for some $R \in \mathit{AURel}(D)$. We must show that R is strict. So let $(\perp, e) \in R$: we show that $e = \perp$. (The case where $(e,\perp) \in R$ is completely symmetric.) By (3.9) it suffices to show that $\varpi'_n\, e = \perp$ for all n. Given n, choose m large enough that $d(R,R_m) \leq 2^{-n}$. Then $\varpi'_n \in R \to R_m$ by Proposition 4.13(1), and therefore $(\perp, \varpi'_n\, e) = (\varpi'_n\, \perp, \varpi'_n\, e) \in R_m$. But this implies that $\varpi'_n\, e = \perp$ since R_m is strict. In conclusion, R is strict.

2: In the proof of Part 1 we showed that $\mathit{CURel}(A)$ is isomorphic to the complete, 1-bounded metric space $\mathit{SAURel}(A_\perp)$ of strict, uniform, and admissible relations on $A_\perp$. The isomorphism is the lifting operator on relations, and this operator clearly preserves

and reflects set-theoretic inclusion, i.e., $R \subseteq S$ if and only if $R_\perp \subseteq S_\perp$. It therefore suffices to show that the set $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} SAURel(A_\perp)$ of non-expansive and monotone functions from $\mathbb{N}_0 \rightharpoonup_{fin} X$ to $SAURel(A_\perp)$ is a complete metric space with the "sup" metric on functions:

$$d'(\nu, \nu') = \sup \{d(\nu(\Delta), \nu'(\Delta)) \mid \Delta \in \mathbb{N}_0 \rightharpoonup_{fin} D\}\,.$$

By Proposition 4.5 it is enough to show that $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} SAURel(A_\perp)$ is a closed subset of $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} AURel(A_\perp)$. But this follows immediately from the fact that $SAURel(A_\perp)$ is a closed subset of $CURel(A_\perp)$, as shown in Part 1, since limits with respect to the "sup" metric on functions are pointwise. □

In the rest of this section we do not need the extra generality of uniform cpos: recall that V is the cpo obtained from Proposition 3.2 and abbreviate $CURel(V) = CURel(V, (\pi_n)_{n\in\omega})$.

Proposition 4.19. The operation mapping each $(X, d) \in CBUlt$ to the monotone function space $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} CURel(V)$ (as given by the previous proposition) can be extended to a locally non-expansive functor $F : CBUlt^{\mathrm{op}} \to CBUlt$ in the natural way:

$$F(X, d) = (\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} CURel(V)$$
$$F(f) = \lambda\nu.\, \lambda\Delta.\, \nu(f \circ \Delta)$$

Proof. Let (X_1, d_1) and (X_2, d_2) be complete, 1-bounded ultrametric spaces. For every non-expansive function f from X_2 to X_1, the $F(f)$ given above is clearly a well-defined function from $(\mathbb{N}_0 \rightharpoonup_{fin} X_1) \to_{mon} CURel(V)$ to the set of functions from $(\mathbb{N}_0 \rightharpoonup_{fin} X_2)$ to $CURel(V)$. It is also easy to see that $F(f)(\nu)$ is monotone for every ν in $F(X_1, d_1)$: let $\Delta, \Delta' \in (\mathbb{N}_0 \rightharpoonup_{fin} X_2)$ such that $\Delta \leq \Delta'$; then $f \circ \Delta \leq f \circ \Delta'$ by definition of $\leq$, and therefore

$$F(f)(\nu)(\Delta) = \nu(f \circ \Delta) \subseteq \nu(f \circ \Delta') = F(f)(\nu)(\Delta')$$

since ν is monotone.

We now show the following property: for all non-expansive functions f and f' from X_2 to X_1, all ν and ν' in $(\mathbb{N}_0 \rightharpoonup_{fin} X_1) \to_{mon} CURel(V)$, and all Δ and Δ' in $(\mathbb{N}_0 \rightharpoonup_{fin} X_2)$,

$$d(F(f)(\nu)(\Delta),\, F(f')(\nu')(\Delta')) \leq \max(d(f, f'), d(\nu, \nu'), d(\Delta, \Delta'))\,. \tag{4.3}$$

By definition, $F(f)(\nu)(\Delta) = \nu(f \circ \Delta)$ and $F(f')(\nu')(\Delta') = \nu'(f' \circ \Delta')$. By the ultrametric inequality,

$$d(f \circ \Delta, f' \circ \Delta') \leq \max(d(f \circ \Delta, f' \circ \Delta), d(f' \circ \Delta, f' \circ \Delta'))$$

But $d(f \circ \Delta, f' \circ \Delta) \leq d(f, f')$ by definition of the metric on $(\mathbb{N}_0 \rightharpoonup_{fin} X_2)$ and $d(f' \circ \Delta, f' \circ \Delta') \leq d(\Delta, \Delta')$ by the fact that f' is non-expansive. Therefore,

$$d(f \circ \Delta, f' \circ \Delta') \leq \max(d(f, f'), d(\Delta, \Delta'))\,.$$

Then, by the ultrametric inequality and the fact that ν' is non-expansive,

$$\begin{aligned} d(\nu(f \circ \Delta), \nu'(f' \circ \Delta')) &\leq \max(d(\nu(f \circ \Delta), \nu'(f \circ \Delta)), d(\nu'(f \circ \Delta), \nu'(f' \circ \Delta'))) \\ &\leq \max(d(\nu, \nu'), d(f \circ \Delta, f' \circ \Delta')) \\ &\leq \max(d(\nu, \nu'), d(f, f'), d(\Delta, \Delta'))\,, \end{aligned}$$

which shows (4.3).

Now, for all f and ν, taking $f' = f$ and $\nu' = \nu$ in (4.3) shows that $F(f)(\nu)$ is non-expansive. Similarly, taking $f' = f$ and $\Delta' = \Delta$ in (4.3) shows that $F(f)$ is non-expansive. All in all, we have now shown that $F(f)$ is a morphism from $F(X_1, d_1)$ to $F(X_2, d_2)$ when f is a morphism from (X_2, d_2) to (X_1, d_1).

The functor laws are then easily verified:

$$F(id_X) = \lambda\nu.\,\lambda\Delta.\,\nu(id_X \circ \Delta) = \lambda\nu.\,\lambda\Delta.\,\nu(\Delta) = id_{F(X,d)}\,.$$

$$\begin{aligned} (F(g) \circ F(f))(\nu) &= ((\lambda\nu.\,\lambda\Delta.\,\nu(g \circ \Delta)) \circ (\lambda\nu.\,\lambda\Delta.\,\nu(f \circ \Delta)))(\nu) \\ &= \lambda\Delta.\,(\lambda\Delta'.\,\nu(f \circ \Delta'))(g \circ \Delta)) \\ &= \lambda\Delta.\,\nu(f \circ g \circ \Delta) \\ &= F(f \circ g)(\Delta)\,. \end{aligned}$$

It remains to show that F is locally non-expansive, i.e., that

$$d(F(f), F(f')) \leq d(f, f')$$

for all non-expansive functions f and f'. But that follows from (4.3) by taking $\nu' = \nu$ and $\Delta' = \Delta'$. □

Proposition 4.19, Proposition 4.9 (with $c = 1/2$), and Theorem 4.10 now immediately imply:

Theorem 4.20. There exists a complete, 1-bounded ultrametric space $\widehat{\mathcal{T}}$ such that the isomorphism

$$\widehat{\mathcal{T}} \cong \tfrac{1}{2}((\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}) \to_{mon} \mathit{CURel}(V)) \tag{4.4}$$

holds in *CBUlt*.

Remark 4.21. Since in general the underlying sets of $1/2 \cdot (X, d)$ and (X, d) are the same, the theorem above gives a continuous, but not distance-preserving, bijection

$$\widehat{\mathcal{T}} \rightleftarrows ((\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}) \to_{mon} \mathit{CURel}(V))\,.$$

We implicitly use that bijection below. Notice that the function space $(\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}) \to_{mon} \mathit{CURel}(V)$ consists of non-expansive functions, so one cannot simply forget about the metric, i.e., generalize to the category of sets and functions and view $\widehat{\mathcal{T}}$ as a solution to an equation like (4.4) but without the "1/2". Likewise, one cannot view $\widehat{\mathcal{T}}$ as a solution to such an equation in the category of metric spaces and *continuous* functions.

4.3. *Interpretation of types*

Let in the following $\widehat{\mathcal{T}}$ be a complete, 1-bounded ultrametric space satisfying (4.4), and let $i : \widehat{\mathcal{T}} \to \frac{1}{2}((\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}) \to_{mon} CURel(V))$ be an isomorphism with inverse $i^{-1} : \frac{1}{2}((\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}) \to_{mon} CURel(V)) \to \widehat{\mathcal{T}}$. For convenience, we use the following abbreviations (where the names $\mathcal{W}$ and $\mathcal{T}$ are intended to indicate "worlds" and "types", respectively):

$$\begin{aligned} \mathcal{W} &= \mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}} \\ \mathcal{T} &= \mathcal{W} \to_{mon} CURel(V)\,. \end{aligned}$$

With that notation, (4.4) expresses that $\widehat{\mathcal{T}}$ is isomorphic to $\frac{1}{2}\mathcal{T}$.

We choose $\mathcal{T}$ as our space of semantic types: types of the language will be interpreted as elements of $\mathcal{T}$, i.e., as certain world-indexed families of relations on V. We additionally define families of relations on "states" (elements of S), "continuations" (elements of $K = V \to S \to Ans$), and "computations" (elements of TV).

Definition 4.22. Abbreviate

$$\begin{aligned} AURel(TV) &= AURel(TV, (\pi_n^T)_{n\in\omega}), \\ AURel(K) &= AURel(K, (\pi_n^K)_{n\in\omega}), \quad \text{and} \\ CURel(S) &= CURel(S, (\pi_n^S)_{n\in\omega})\,. \end{aligned}$$

Let

$$\begin{aligned} \mathcal{T}_T &= \mathcal{W} \to_{mon} AURel(TV) \\ \mathcal{T}_K &= \mathcal{W} \to_{mon} AURel(K) \end{aligned}$$

be the complete, uniform 1-bounded ultrametric spaces given by Proposition 4.17. Furthermore, let

$$\mathcal{T}_S = \mathcal{W} \to CURel(S)$$

be the complete, uniform 1-bounded ultrametric space obtained from Propositions 4.15 and 4.18 and the exponential in CBUlt. (The elements of $\mathcal{T}_S$ are non-expansive but not necessarily monotone functions.)

In all the ultrametric spaces we consider here, all non-zero distances have the form 2^{-m} for some m. For such ultrametric spaces, there is a useful notion of n-approximated equality of elements:

Definition 4.23. For every complete, 1-bounded ultrametric space (D, d), every natural number $n \geq 0$, and all elements $x, y \in D$, the notation $x \overset{n}{=}_d y$ means that $d(x, y) \leq 2^{-n}$. When the distance function d is clear from the context, we shall just write $x \overset{n}{=} y$ for $x \overset{n}{=}_d y$.

(In general, such approximated equality relations can of course also be defined for numbers not of the form 2^{-n}.) The ultrametric inequality implies that each relation $\overset{n}{=}_d$ is transitive, and therefore an equivalence relation:

Proposition 4.24. If $x \stackrel{n}{=}_d y$ and $y \stackrel{n}{=}_d z$, then $x \stackrel{n}{=}_d z$.

The fact that the evaluation map corresponding to a given exponential is non-expansive can now be expressed as a congruence property for approximated equality: for non-expansive maps $f, f' : (D_1, d_1) \to (D_2, d_2)$ and elements $x, x' \in D_1$,

$$f \stackrel{n}{=} f' \ \wedge \ x \stackrel{n}{=} x' \implies f(x) \stackrel{n}{=} f'(x')\,. \tag{4.5}$$

That property will be used frequently below.

To interpret types of the language as elements of $\mathcal{T}$, it remains to define a number of operators on $\mathcal{T}$ (and $\mathcal{T}_T$ and $\mathcal{T}_K$) that will be used to interpret the various type constructors of the language; these operators are shown in the lower part of Figure 6. Notice that the operator *ref* is defined in terms of n-approximated equality $\stackrel{n}{=}$ on $\mathit{CURel}(V)$, as defined above.

In order to interpret the fragment of the language without recursive types, it suffices to verify that these operators are well-defined (e.g., *ref* actually maps elements of $\mathcal{T}$ into $\mathcal{T}$.) In order to interpret recursive types, however, we furthermore need to verify that the operators are non-expansive.

The proofs below depend on a number of lemmas that give more concrete descriptions of the metric spaces involved; these lemmas can be found in Appendix A. In particular, the factor $1/2$ in (4.4) implies that worlds that are "$(n+1)$-equal" only contain "n-equal" semantic types.

Lemma 4.25. The function *states* from $\mathcal{W}$ to $\mathit{Rel}(S)$ defined in the lower part of Figure 6 is an element of $\mathcal{T}_S$.

Proof. First, for every $\Delta \in \mathcal{W}$, the relation $\mathit{states}(\Delta)$ is complete: this follows from the fact that $i(\Delta(l))\,(\Delta)$ is complete for all $l \in \mathrm{dom}(\Delta)$. We now show that

$$\Delta \stackrel{n}{=} \Delta' \implies \pi_n^S \in \mathit{states}(\Delta) \to \mathit{states}(\Delta')_\perp$$

for all $\Delta, \Delta' \in \mathcal{W}$. From this implication, uniformity follows by taking $\Delta' = \Delta$ and using Lemma A.2(1), and non-expansiveness of *states* follows from Lemma A.2(1) and symmetry. So, let $\Delta \stackrel{n}{=} \Delta'$ and let $(s, s') \in \mathit{states}(\Delta)$; we must show that either $\pi_n^S(s) = \pi_n^S(s') = \perp$, or $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ where $(s_0, s_0') \in \mathit{states}(\Delta')$. If $n = 0$ we are done by (3.23); assume therefore that $n > 0$. Then $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ by the definition of the metric on $\mathcal{W}$, and furthermore, for every $l \in \mathrm{dom}(\Delta)$,

$$\begin{aligned} i(\Delta(l))(\Delta) &\stackrel{n}{=} i(\Delta(l))\,(\Delta') && (i(\Delta(l)) \text{ non-expansive}) \\ &\stackrel{n-1}{=} i(\Delta'(l))\,(\Delta')\,. && (\text{Lemma A.1}) \end{aligned}$$

By transitivity (Proposition 4.24),

$$i(\Delta(l))\,(\Delta) \stackrel{n-1}{=} i(\Delta'(l))\,(\Delta')\,,$$

and therefore Lemma A.2(1) gives that

$$\pi_{n-1} \in i(\Delta(l))\,(\Delta) \to (i(\Delta'(l))\,(\Delta'))_\perp\,.$$

Since the above holds for every $l \in \mathrm{dom}(\Delta)$, Equation (3.24) gives that either $\pi_n^S(s) =$

For every $\Xi \vdash \tau$, define the non-expansive $[\![\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ by induction on τ:

$$
\begin{aligned}
[\![\alpha]\!]_\Xi\varphi &= \varphi(\alpha)\\
[\![\mathsf{int}]\!]_\Xi\varphi &= \lambda\Delta.\,\{\,(in_{\mathbb{Z}}\,k, in_{\mathbb{Z}}\,k) \mid k \in \mathbb{Z}\,\}\\
[\![1]\!]_\Xi\varphi &= \lambda\Delta.\,\{\,(in_1\,*, in_1\,*)\,\}\\
[\![\tau_1 \times \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi \times [\![\tau_2]\!]_\Xi\varphi\\
[\![0]\!]_\Xi\varphi &= \lambda\Delta.\,\emptyset\\
[\![\tau_1 + \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi + [\![\tau_2]\!]_\Xi\varphi\\
[\![\mathsf{ref}\,\tau]\!]_\Xi\varphi &= \mathit{ref}([\![\tau]\!]_\Xi\varphi)\\
[\![\forall\alpha.\tau]\!]_\Xi\varphi &= \lambda\Delta.\,\{\,(in_\forall\,c, in_\forall\,c') \mid \forall\nu \in \mathcal{T}.\,(c,c') \in \mathit{comp}([\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])(\Delta)\,\}\\
[\![\mu\alpha.\tau]\!]_\Xi\varphi &= \mathit{fix}\,\big(\lambda\nu.\,\lambda\Delta.\,\{\,(in_\mu\,v, in_\mu\,v') \mid (v,v') \in [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu]\,(\Delta)\,\}\big)\\
&\qquad \text{(see Theorem 4.29)}\\
[\![\tau_1 \to \tau_2]\!]_\Xi\varphi &= ([\![\tau_1]\!]_\Xi\varphi) \to (\mathit{comp}([\![\tau_2]\!]_\Xi\varphi))
\end{aligned}
$$

The following operators and elements are used above:

$$
\begin{aligned}
\times &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & \mathit{comp} &: \mathcal{T} \to \mathcal{T}_T\\
+ &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & \mathit{cont} &: \mathcal{T} \to \mathcal{T}_K\\
\mathit{ref} &: \mathcal{T} \to \mathcal{T} & \mathit{states} &\in \mathcal{T}_S\\
\to &: \mathcal{T} \times \mathcal{T}_T \to \mathcal{T} & R_{Ans} &\in \mathit{CRel}(Ans)
\end{aligned}
$$

$$
(\nu_1 \times \nu_2)(\Delta) = \{\,(in_\times(v_1,v_2), in_\times(v_1',v_2')) \mid (v_1,v_1') \in \nu_1(\Delta) \wedge (v_2,v_2') \in \nu_2(\Delta)\,\}
$$

$$
\begin{aligned}
(\nu_1 + \nu_2)(\Delta) = {} & \{\,(in_+(\iota_1\,v_1), in_+(\iota_1\,v_1')) \mid (v_1,v_1') \in \nu_1(\Delta)\,\} \cup\\
& \{\,(in_+(\iota_2\,v_2), in_+(\iota_2\,v_2')) \mid (v_2,v_2') \in \nu_2(\Delta)\,\}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{ref}(\nu)(\Delta) = {} & \{\,(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \wedge \forall\Delta_1 \geq \Delta.\ i(\Delta(l))\,(\Delta_1) = \nu(\Delta_1)\,\} \cup\\
& \{\,(\lambda_l^{n+1}, \lambda_l^{n+1}) \mid l \in \mathrm{dom}(\Delta) \wedge \forall\Delta_1 \geq \Delta.\ i(\Delta(l))\,(\Delta_1) \stackrel{n}{=} \nu(\Delta_1)\,\}
\end{aligned}
$$

$$
(\nu \to \xi)(\Delta) = \{\,(in_\to\,f, in_\to\,f') \mid \forall\Delta_1 \geq \Delta.\,\forall(v,v') \in \nu(\Delta_1)\,.(f\,v, f'\,v') \in \xi(\Delta_1)\,\}
$$

$$
\begin{aligned}
\mathit{cont}(\nu)(\Delta) = \{\,(k,k') \mid {} & \forall\Delta_1 \geq \Delta.\,\forall(v,v') \in \nu(\Delta_1).\\
& \forall(s,s') \in \mathit{states}(\Delta_1).\,(k\,v\,s, k'\,v'\,s') \in R_{Ans}\,\}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{comp}(\nu)(\Delta) = \{\,(c,c') \mid {} & \forall\Delta_1 \geq \Delta.\,\forall(k,k') \in \mathit{cont}(\nu)(\Delta_1).\\
& \forall(s,s') \in \mathit{states}(\Delta_1).\,(c\,k\,s, c'\,k'\,s') \in R_{Ans}\,\}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{states}(\Delta) = \{\,(s,s') \mid {} & \mathrm{dom}(s) = \mathrm{dom}(s') = \mathrm{dom}(\Delta)\\
& \wedge\ \ \forall l \in \mathrm{dom}(\Delta).\,(s(l), s'(l)) \in i(\Delta(l))\,(\Delta)\,\}
\end{aligned}
$$

$$
R_{Ans} = \{\,(\bot,\bot)\,\} \cup \{\,(\lfloor\iota_1\,m\rfloor, \lfloor\iota_1\,m\rfloor) \mid m \in \mathbb{Z}\,\}
$$

Fig. 6. Interpretation of types.

$\pi_n^S(s') = \bot$, and we are done, or $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ for some s_0 and s_0' such that $(s_0(l), s_0'(l)) \in i(\Delta'(l))\,(\Delta')$ for all $l \in \mathrm{dom}(\Delta')$. But the latter means exactly that $(s_0, s_0') \in \mathit{states}(\Delta')$. □

Lemma 4.26. Let Δ, Δ', and Δ_1 be elements of $\mathcal{W}$ such that $\Delta \stackrel{n}{=} \Delta'$ and $\Delta \leq \Delta_1$. There exists a Δ_1' such that $\Delta_1 \stackrel{n}{=} \Delta_1'$ and $\Delta' \leq \Delta_1'$.

Proof. If $n = 0$ we can take $\Delta_1' = \Delta'$; in fact, any extension of Δ' would do. If $n > 0$ we have $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ by definition of the metric on $\mathcal{W}$. Now define $\Delta_1' \in \mathcal{W}$ with $\mathrm{dom}(\Delta_1') = \mathrm{dom}(\Delta_1)$ by

$$\Delta_1'(l) = \begin{cases} \Delta'(l) & \text{if } l \in \mathrm{dom}(\Delta) \\ \Delta_1(l) & \text{if } l \in \mathrm{dom}(\Delta_1) \setminus \mathrm{dom}(\Delta). \end{cases}$$

Clearly $\Delta' \leq \Delta_1'$ since $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$. Also, by definition of the metric on $\mathcal{W}$ (as a maximum of the distances for each "l"), $d(\Delta_1, \Delta_1') = d(\Delta, \Delta') \leq 2^{-n}$. □

Lemma 4.27. The operators $\times$, $+$, *ref*, $\to$, *cont*, and *comp* defined in the lower part of Figure 6 are non-expansive.

Proof. We show that each operator maps into the appropriate codomain and that it is non-expansive.

$\times : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$:
It is easy to see that $(\nu_1 \times \nu_2)(\Delta)$ is complete for all $\Delta \in \mathcal{W}$. To see that $\nu_1 \times \nu_2$ belongs to $\mathcal{T}$, it therefore suffices to verify the two conditions of Lemma A.2(3). Condition (a), monotonicity, is immediate. As for Condition (b), we show a more general fact which furthermore implies non-expansiveness of $\times$: for all ν_1, ν_2, ν_1', and ν_2' in $\mathcal{T}$ and all Δ and Δ' in $\mathbb{N}_0 \rightharpoonup_{\mathit{fin}} \widehat{\mathcal{T}}$,

$$\nu_1 \stackrel{n}{=} \nu_1' \ \wedge\ \nu_2 \stackrel{n}{=} \nu_2' \ \wedge\ \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in (\nu_1 \times \nu_2)(\Delta) \to (\nu_1' \times \nu_2')(\Delta)_\bot\,.$$

Condition (b) then follows by taking $\nu_1 = \nu_1'$ and $\nu_2 = \nu_2'$. Non-expansiveness of $\times$ follows by taking $\Delta = \Delta'$ and using parts 1 and 2 of Lemma A.2 (and symmetry).

So, assume that $\nu_1 \stackrel{n}{=} \nu_1'$ and $\nu_2 \stackrel{n}{=} \nu_2'$ and $\Delta \stackrel{n}{=} \Delta'$, and let

$$(\mathit{in}_\times(v_1, v_2), \mathit{in}_\times(v_1', v_2')) \in (\nu_1 \times \nu_2)(\Delta)\,.$$

We must show that either (1) $\pi_n(\mathit{in}_\times(v_1, v_2)) = \pi_n(\mathit{in}_\times(v_1', v_2')) = \bot$ or otherwise (2) $\pi_n(\mathit{in}_\times(v_1, v_2)) = \lfloor w \rfloor$ and $\pi_n(\mathit{in}_\times(v_1', v_2')) = \lfloor w' \rfloor$ for some w and w' such that $(w, w') \in (\nu_1' \times \nu_2')(\Delta')$. If $n = 0$ we are done by Equation (3.4); assume therefore that $n > 0$. By definition of $(\nu_1 \times \nu_2)(\Delta)$ we know that $(v_1, v_1') \in \nu_1(\Delta)$ and $(v_2, v_2') \in \nu_2(\Delta)$. Since ν_1 and ν_2 are non-expansive functions, (4.5) gives that

$$\nu_1(\Delta) \stackrel{n-1}{=} \nu_1'(\Delta') \quad \text{and} \quad \nu_2(\Delta) \stackrel{n-1}{=} \nu_2'(\Delta')\,.$$

Therefore $\pi_{n-1} \in \nu_1(\Delta) \to \nu_1'(\Delta')_\bot$ and $\pi_{n-1} \in \nu_2(\Delta) \to \nu_2'(\Delta')_\bot$ by Lemma A.2(1). By definition of $\nu_i(\Delta) \to \nu_i'(\Delta')_\bot$ (for $i = 1, 2$) there are now two cases:

1. $\pi_{n-1}(v_1) = \pi_{n-1}(v_1') = \bot$ or $\pi_{n-1}(v_2) = \pi_{n-1}(v_2') = \bot$.

2. There exist $(w_1, w_1') \in \nu_1'(\Delta')$ and $(w_2, w_2') \in \nu_2'(\Delta')$ where $\pi_{n-1}(v_1) = \lfloor w_1 \rfloor$ and $\pi_{n-1}(v_1') = \lfloor w_1' \rfloor$ and $\pi_{n-1}(v_2) = \lfloor w_2 \rfloor$ and $\pi_{n-1}(v_2') = \lfloor w_2' \rfloor$.

In case (1), (3.18) gives that $\pi_n(in_\times(v_1, v_2)) = \pi_n(in_\times(v_1', v_2')) = \bot$ and we are done. In case (2), (3.18) gives that $\pi_n(in_\times(v_1, v_2)) = \lfloor in_\times(w_1, w_2) \rfloor$ and $\pi_n(in_\times(v_1', v_2')) = \lfloor in_\times(w_1', w_2') \rfloor$. By definition of $(\nu_1' \times \nu_2')(\Delta')$ we have that $(in_\times(w_1, w_2), in_\times(w_1', w_2')) \in (\nu_1' \times \nu_2')(\Delta')$ and we are done.

$ref : \mathcal{T} \to \mathcal{T}$:
First, $ref(\nu)(\Delta)$ is complete for all Δ: this follows from the general fact that if $R \stackrel{n}{=} S$ for all $n \in \omega$, then $d(R, S) = 0$ and hence $R = S$. It is also easy to see that $ref(\nu)$ is monotone. Similarly to the previous case, we then show that $ref(\nu)$ belongs to $\mathcal{T}$ and that ref is non-expansive by showing that

$$\nu \stackrel{n}{=} \nu' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in ref(\nu)(\Delta) \to ref(\nu')(\Delta')_\bot$$

for all ν and ν' in $\mathcal{T}$ and all Δ and Δ' in $\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}$.

So, assume that $\nu \stackrel{n}{=} \nu'$ and $\Delta \stackrel{n}{=} \Delta'$, and let $(\lambda_l^m, \lambda_l^m) \in ref(\nu)(\Delta)$. (The case where $(\lambda_l, \lambda_l) \in ref(\nu)(\Delta)$ is similar, but slightly easier.) If $n = 0$ we are done by Equation (3.4). If $n > 0$, (3.17) gives that $\pi_n(\lambda_l^m) = \lfloor \lambda_l^{\min(n,m)} \rfloor$, and it therefore remains to show that $(\lambda_l^{\min(n,m)}, \lambda_l^{\min(n,m)}) \in ref(\nu')(\Delta')$. To that end, let $l \in \mathrm{dom}(\Delta')$ and $\Delta_1' \geq \Delta'$; we must show that $i\,(\Delta'(l))\,(\Delta_1') \stackrel{\min(n,m)-1}{=} \nu'(\Delta_1')$. Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta_1'$. Then:

$$\begin{aligned}
i(\Delta'(l))\,(\Delta_1') &\stackrel{n}{=} i(\Delta'(l))\,(\Delta_1) && (i(\Delta'(l)) \text{ non-expansive}) \\
&\stackrel{n-1}{=} i(\Delta(l))\,(\Delta_1) && (\text{Lemma A.1}) \\
&\stackrel{m-1}{=} \nu(\Delta_1) && (\text{since } (\lambda_l^m, \lambda_l^m) \in ref(\nu)(\Delta)) \\
&\stackrel{n}{=} \nu'(\Delta_1) && (\text{Lemma A.2(2)}) \\
&\stackrel{n}{=} \nu'(\Delta_1')\,. && (\nu' \text{ non-expansive})
\end{aligned}$$

Hence by transitivity $i(\Delta'(l))\,(\Delta_1') \stackrel{\min(n,m)-1}{=} \nu'(\Delta_1')$.

$+ : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$:
It is easy to see that $(\nu_1 + \nu_2)(\Delta)$ is complete for all $\Delta \in W$, and that $\nu_1 + \nu_2$ is monotone. It then suffices to show that

$$\nu_1 \stackrel{n}{=} \nu_1' \ \wedge \ \nu_2 \stackrel{n}{=} \nu_2' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in (\nu_1 + \nu_2)(\Delta) \to (\nu_1' + \nu_2')(\Delta')_\bot$$

for all ν_1, ν_2, ν_1', and ν_2' in $\mathcal{T}$ and all Δ and Δ' in $\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathcal{T}}$.

So, assume that $\nu_1 \stackrel{n}{=} \nu_1'$ and $\nu_2 \stackrel{n}{=} \nu_2'$ and $\Delta \stackrel{n}{=} \Delta'$, and let

$$(in_+(\iota_1\, v), in_+(\iota_1 v')) \in (\nu_1 + \nu_2)(\Delta)\,.$$

(The case with ι_2 instead of ι_1 is completely symmetric.) If $n = 0$ we are done by Equation (3.4); assume therefore that $n > 0$. By definition of $(\nu_1 + \nu_2)(\Delta)$ we have $(v, v') \in \nu_1(\Delta)$. Then, since $\nu_1 \stackrel{n}{=} \nu_1'$ and $\Delta \stackrel{n}{=} \Delta'$ implies $\nu_1(\Delta) \stackrel{n}{=} \nu_1'(\Delta')$, there are two

cases: either $\pi_{n-1}(v) = \pi_{n-1}(v') = \bot$, and we are done, or $\pi_{n-1}(v) = \lfloor w \rfloor$ and $\pi_{n-1}(v') = \lfloor w' \rfloor$ where $(w, w') \in \nu_1'(\Delta')$. But then (3.19) gives that $\pi_n(in_+(\iota_1 v)) = \lfloor in_+(\iota_1 w) \rfloor$ and $\pi_n(in_+(\iota_1 v')) = \lfloor in_+(\iota_1 w') \rfloor$ with $(in_+(\iota_1 w), in_+(\iota_1 w')) \in (\nu_1' + \nu_2')(\Delta')$.

$cont : \mathcal{T} \to \mathcal{T}_K$:
First, $cont(\nu)(\Delta)$ is admissible for each $\Delta \in \mathcal{W}$ since R_{Ans} is admissible. Also, $cont(\nu)$ is monotone. By Lemma A.3(3), it then suffices to show that

$$\nu \stackrel{n}{=} \nu' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n^K \in cont(\nu)(\Delta) \to cont(\nu')(\Delta')$$

for all ν and ν' in $\mathcal{T}$ and all Δ and Δ' in $\mathcal{W}$. So, assume that $\nu \stackrel{n}{=} \nu'$ and $\Delta \stackrel{n}{=} \Delta'$ and let $(k, k') \in cont(\nu)(\Delta)$; we must show that $(\pi_n^K(k), \pi_n^K(k')) \in cont(\nu')(\Delta')$. If $n = 0$ this follows from (3.23) and the fact that $cont(\nu')(\Delta')$ is pointed. Otherwise, let $\Delta_1' \geq \Delta'$ and $(v, v') \in \nu'(\Delta_1')$ and $(s, s') \in states(\Delta_1')$. We must show that $(\pi_n^K(k)\, v\, s,\ \pi_n^K(k')\, v'\, s') \in R_{Ans}$. First, Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta_1'$. By (4.5), $\nu(\Delta_1) \stackrel{n}{=} \nu'(\Delta_1')$. Furthermore, the fact that *states* belongs to $\mathcal{T}_S$, shown above, implies that $states(\Delta_1) \stackrel{n}{=} states(\Delta_1')$. Therefore, by (3.25), either $\pi_n^K(k)\, v\, s = \pi_n^K(k')\, v'\, s' = \bot$, and we are done, or $\pi_n^K(k)\, v\, s = k\, w\, s_0$ and $\pi_n^K(k')\, v'\, s' = k'\, w'\, s_0'$ where $\pi_n(v) = \lfloor w \rfloor$ and $\pi_n(v') = \lfloor w' \rfloor$ and $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ with $(w, w') \in \nu(\Delta_1)$ and $(s_0, s_0') \in states(\Delta_1)$. In the latter case, $(k\, w\, s_0, k'\, w'\, s_0') \in R_{Ans}$ since $(k, k') \in cont(\nu)(\Delta)$ and $\Delta \leq \Delta_1$.

$comp : \mathcal{T} \to \mathcal{T}_T$:
Similar to *cont*. First, $comp(\nu)(\Delta)$ is admissible for each $\Delta \in \mathcal{W}$ since R_{Ans} is admissible. Also, $comp(\nu)$ is monotone. By Lemma A.3(3), it then suffices to show that

$$\nu \stackrel{n}{=} \nu' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n^T \in comp(\nu)(\Delta) \to comp(\nu')(\Delta')$$

for all ν and ν' in $\mathcal{T}$ and all Δ and Δ' in $\mathcal{W}$. So, assume that $\nu \stackrel{n}{=} \nu'$ and $\Delta \stackrel{n}{=} \Delta'$ and let $(c, c') \in comp(\nu)(\Delta)$; we must show that $(\pi_n^T(c), \pi_n^T(c')) \in comp(\nu')(\Delta')$. If $n = 0$ this follows from (3.23) and the fact that $comp(\nu')(\Delta')$ is pointed. Otherwise, let $\Delta_1' \geq \Delta'$ and $(k, k') \in cont(\nu')(\Delta_1')$ and $(s, s') \in states(\Delta_1')$. We must show that $(\pi_n^T(c)\, k\, s,\ \pi_n^T(c')\, k'\, s') \in R_{Ans}$. Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta_1'$. Since *cont* is non-expansive,

$$cont(\nu)(\Delta_1) \stackrel{n}{=} cont(\nu')(\Delta_1) \stackrel{n}{=} cont(\nu')(\Delta_1')\,.$$

Furthermore, the fact that *states* belongs to $\mathcal{T}_S$ implies that $states(\Delta_1) \stackrel{n}{=} states(\Delta_1')$. Therefore, by (3.26), either $\pi_n^T(c)\, k\, s = \pi_n^T(c')\, k'\, s' = \bot$, and we are done, or $\pi_n^T(c)\, k\, s = c\,(\pi_n^K(k))\, s_0$ and $\pi_n^T(c')\, k'\, s' = c'\,(\pi_n^K(k'))\, s_0'$ where $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ with $(\pi_n^K(k), \pi_n^K(k')) \in cont(\nu)(\Delta_1)$ and $(s_0, s_0') \in states(\Delta_1)$. In the latter case,

$$(c\,(\pi_n^K(k))\, s_0, c'\,(\pi_n^K(k'))\, s_0') \in R_{Ans}$$

since $(c, c') \in comp(\nu)(\Delta)$ and $\Delta \leq \Delta_1$.

$\to : \mathcal{T} \times \mathcal{T}_T \to \mathcal{T}$:
It is easy to see that $(\nu \to \xi)(\Delta)$ is admissible for all $\Delta \in \mathcal{W}$ since ξ maps worlds

to admissible relations. Also, $\nu \to \xi$ is obviously monotone. By Lemma A.3(3), it then suffices to show that

$$\nu \stackrel{n}{=} \nu' \ \wedge \ \xi \stackrel{n}{=} \xi' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in (\nu \to \xi)(\Delta) \to (\nu' \to \xi')(\Delta')_\perp$$

for all ν and ν' in $\mathcal{T}$, all ξ and ξ' in $\mathcal{T}_T$, and all Δ and Δ' in $\mathcal{W}$. So, assume that $\nu \stackrel{n}{=} \nu'$ and $\xi \stackrel{n}{=} \xi'$ and $\Delta \stackrel{n}{=} \Delta'$, and let $(in_\to f, in_\to f') \in (\nu \to \xi)(\Delta)$. If $n = 0$ we are done by Equation (3.4); assume therefore that $n > 0$. Define the two functions

$$g = \lambda v. \begin{cases} \pi_n^T\,(f\,w) & \text{if } \pi_{n-1}\,v = \lfloor w \rfloor \\ \perp & \text{otherwise} \end{cases}$$

$$g' = \lambda v'. \begin{cases} \pi_n^T\,(f'\,w') & \text{if } \pi_{n-1}\,v' = \lfloor w' \rfloor \\ \perp & \text{otherwise} \end{cases}$$

By (3.22), it suffices to show that $(in_\to(g), in_\to(g')) \in (\nu' \to \xi')(\Delta')$. To that end, let $\Delta'_1 \geq \Delta'$ and let $(v, v') \in \nu'(\Delta'_1)$; we must show that $(g(v), g'(v')) \in \xi'(\Delta'_1)$. Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta'_1$. Then $\nu(\Delta_1) \stackrel{n-1}{=} \nu'(\Delta'_1)$, and there are therefore two cases: either $\pi_{n-1}\,v = \pi_{n-1}\,v' = \perp$, and we are done, or $\pi_{n-1}\,v = \lfloor w \rfloor$ and $\pi_{n-1}\,v' = \lfloor w' \rfloor$ for some w, w' such that $(w, w') \in \nu(\Delta_1)$. In the latter case $(f\,w, f'\,w') \in \xi(\Delta_1)$ since $(f, f') \in (\nu \to \xi)(\Delta)$ and $\Delta_1 \geq \Delta$. Then by (4.5), $\xi(\Delta_1) \stackrel{n}{=} \xi'(\Delta'_1)$, and therefore $(\pi_n^T(f\,w), \pi_n^T(f'\,w')) \in \xi'(\Delta'_1)$. But this means exactly that $(g(v), g'(v')) \in \xi'(\Delta'_1)$.

□

It is here, in order to show that *ref* is well-defined (and non-expansive), that we need the approximate locations λ_l^n. Suppose for the sake of argument that locations were modeled simply using a flat cpo of natural numbers, i.e., suppose that $Loc = \mathbb{N}_0$ and that $\pi_1(in_{Loc}\,l) = \lfloor in_{Loc}\,l \rfloor$ for all $l \in \mathbb{N}_0$. The definition of *ref* would then have the form $ref(\nu)(\Delta) = \{(in_{Loc}\,l, in_{Loc}\,l) \mid l \in \mathrm{dom}(\Delta) \wedge \ldots\}$. The function $ref(\nu)$ from worlds to relations must be non-expansive. But assume then that $\Delta =_1 \Delta'$; then $ref(\nu)(\Delta) =_1 ref(\nu)(\Delta')$ by non-expansiveness, and hence $ref(\nu)(\Delta) = ref(\nu)(\Delta')$ since π_1 is the (lifted) identity on locations. In other words, $ref(\nu)$ would only depend on the "first approximation" of its argument world Δ: this can never be right, no matter what the particular definition of *ref* is.[‡] This observation generalizes to variants where $\pi_n(in_{Loc}\,l) = \lfloor in_{Loc}\,l \rfloor$) for some arbitrary finite n.

For any finite set Ξ of type variables, the set $\mathcal{T}^\Xi$ of functions from Ξ to $\mathcal{T}$ is a metric space with the product metric:

$$d'(\varphi, \varphi') = \max\{\, d(\varphi(\alpha), \varphi'(\alpha)) \mid \alpha \in \Xi \,\}\,.$$

We are now ready to formulate the interpretation of types:

Definition 4.28. Let τ be a type and let Ξ be a type environment such that $\Xi \vdash \tau$. The *relational interpretation of τ with respect to* Ξ is the non-expansive function $[\![\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ defined by induction on τ in Figure 6. The interpretation of recursive types is by appeal to Banach's fixed-point theorem (see Theorem 4.29).

[‡] In particular, the obvious definition of *ref* as $ref(\nu)(\Delta) = \{(in_{Loc}\,l, in_{Loc}\,l) \mid l \in \mathrm{dom}(\Delta) \wedge \forall \Delta_1 \geq \Delta.\ i(\Delta(l))\ (\Delta_1) = \nu(\Delta_1)\}$ would *not* be well-defined, since it would not be non-expansive in Δ.

In more detail, well-definedness of $[\![\tau]\!]_\Xi$ must be argued together with non-expansiveness, by induction on τ (see below). This is similar to the more familiar situation with the untyped semantics of terms presented in Section 3: there, well-definedness must be argued together with continuity because of the use of Kleene's fixed-point theorem in the interpretation of $\mathsf{fix}\, f.\lambda x.t$.

Theorem 4.29. Let τ be a type such that $\Xi \vdash \tau$.

1. The function $[\![\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ defined in Figure 6 is non-expansive.
2. If $\Xi = \Xi', \alpha$ then for all $\varphi' \in \mathcal{T}^{\Xi'}$ we have that $\lambda\nu.\,\lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau]\!]_{\Xi',\alpha}\varphi'[\alpha \mapsto \nu]\,\Delta\,\}$ is a contractive function from $\mathcal{T}$ to $\mathcal{T}$. In particular, $[\![\mu\alpha.\tau]\!]$ is well-defined.

Proof. First, generalize Part 2 above:

2'. $\lambda\varphi\,\lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau]\!]_\Xi\varphi\,\Delta\,\}$ is a contractive function from $\mathcal{T}^\Xi$ to $\mathcal{T}$.

By the definition of the product metric, 2' implies 2.

We now show 1 and 2' by simultaneous induction on n.

1: If τ is int, 1, or 0, then $[\![\tau]\!]_\Xi$ is a constant function and hence trivially non-expansive. If τ is a type variable α, then non-expansiveness of $[\![\tau]\!]_\Xi$ follows directly from the definition of the product metric. In the cases where τ is $\tau_1 \times \tau_2$, $\tau_1 + \tau_2$, *ref* τ', or $\tau_1 \to \tau_2$, non-expansiveness follows directly from Lemma 4.27 and the induction hypothesis.

It remains to consider the cases where τ is $\mu\alpha.\tau'$ or $\forall\alpha.\tau'$. First, assume that τ is $\mu\alpha.\tau'$ for some τ' such that $\Xi, \alpha \vdash \tau'$. We know from 2' and the induction hypothesis that $[\![\mu\alpha.\tau']\!]_\Xi$ is a (well-defined) function from $\mathcal{T}^\Xi$ to $\mathcal{T}$. To show that $[\![\mu\alpha.\tau']\!]_\Xi$ is non-expansive, let $\varphi \overset{n}{=} \varphi'$; we must show that $[\![\mu\alpha.\tau']\!]_\Xi\varphi \overset{n}{=} [\![\mu\alpha.\tau']\!]_\Xi\varphi'$. By Proposition 4.7 it suffices to show that the two contractive functions $g, g' : \mathcal{T} \to \mathcal{T}$ defined by

$$g = \lambda\nu.\,\lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu]\,\Delta\,\}$$
$$g' = \lambda\nu.\,\lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto \nu]\,\Delta\,\}$$

satisfy that $g \overset{n}{=} g'$. So let $\nu \in \mathcal{T}$ be given; we must show that $g\,\nu \overset{n}{=} g'\,\nu$. But this follows from 2' and the induction hypothesis. Therefore, $[\![\mu\alpha.\tau']\!]_\Xi$ is non-expansive.

Now assume that τ is $\forall\alpha.\tau'$ for some τ' such that $\Xi, \alpha \vdash \tau'$. First, $[\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta$ is complete for all $\Delta \in \mathcal{W}$ since arbitrary intersections of complete relations are complete. It is also easy to see that $[\![\forall\alpha.\tau']\!]_\Xi\varphi$ is monotone since $comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \to \nu])$ is monotone for all $\nu \in \mathcal{T}$. By Lemma A.2(3), it then suffices to show that

$$\varphi \overset{n}{=} \varphi' \;\wedge\; \Delta \overset{n}{=} \Delta' \implies \pi_n \in [\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta \to ([\![\forall\alpha.\tau']\!]_\Xi\varphi'\,\Delta')_\perp$$

for all φ and φ' in $\mathcal{T}^\Xi$ and all Δ and Δ' in $\mathcal{W}$. So, let $(in_\forall\, c, in_\forall\, c') \in [\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta$. If $n = 0$ we are done by Equation (3.4); assume therefore that $n > 0$. By (3.21) it then suffices to show that $(in_\forall(\pi_n^T\, c),\ in_\forall(\pi_n^T\, c')) \in [\![\forall\alpha.\tau']\!]_\Xi\varphi'\,\Delta'$. To this end, let $\Delta_1' \geq \Delta'$ and $\nu \in \mathcal{T}$; we must show that $(\pi_n^T\, c, \pi_n^T\, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto \nu])\Delta_1'$. Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \overset{n}{=} \Delta_1'$. Then $(c, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])\Delta_1$ since $(in_\forall\, c, in_\forall\, c') \in [\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta$. By the induction hypothesis, $[\![\tau']\!]_{\Xi,\alpha}$ is non-expansive, and

therefore

$$[\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu] \stackrel{n}{=} [\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto \nu]$$

by the definition of the product metric. Now, the operator *comp* is non-expansive by Lemma 4.27, and therefore

$$comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu]) \stackrel{n}{=} comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto \nu])\,.$$

Finally, by (4.5),

$$comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])\Delta_1 \stackrel{n}{=} comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto \nu])\Delta_1'\,,$$

and we conclude that

$$(\pi_n^T\, c, \pi_n^T\, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto \nu])\Delta_1'$$

by Lemma A.3(1) and the fact that $(c, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])\Delta_1$.

2': Let $G = \lambda\varphi\,\lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau]\!]_\Xi\varphi\,\Delta\,\}$; we must show that G is a contractive function from $\mathcal{T}^\Xi$ to $\mathcal{T}$. First, it is easy to see that $G(\varphi)$ is monotone and that $G(\varphi)(\Delta)$ is admissible for all φ and Δ. To show that G has codomain $\mathcal{T}$ it therefore remains to verify Condition (b) of Lemma A.2(3). We show the following more general property which furthermore implies that G is contractive: for all φ and φ' in $\mathcal{T}^\Xi$ and all Δ and Δ' in $\mathcal{W}$,

$$\varphi \stackrel{n}{=} \varphi' \;\wedge\; \Delta \stackrel{n}{=} \Delta' \implies \pi_{n+1} \in G(\varphi)(\Delta) \to G(\varphi')(\Delta')_\perp\,.$$

Notice the $n+1$ on the right-hand side: the above property implies that G is contractive with factor $\delta = 1/2$ (by taking $\Delta = \Delta'$ and using Lemma A.2(1) and symmetry).

So, let $\varphi \stackrel{n}{=} \varphi'$ and $\Delta \stackrel{n}{=} \Delta'$, and let $(in_\mu\, v, in_\mu\, v') \in G(\varphi)(\Delta)$. We know that $(v, v') \in [\![\tau]\!]_\Xi\varphi\,\Delta$ by definition of G. Part 1 gives that $[\![\tau]\!]_\Xi$ is non-expansive, and therefore $[\![\tau]\!]_\Xi\varphi \stackrel{n}{=} [\![\tau]\!]_\Xi\varphi'$. By (4.5), $[\![\tau]\!]_\Xi\varphi\,\Delta \stackrel{n}{=} [\![\tau]\!]_\Xi\varphi'\,\Delta'$, and there are therefore two cases: either $\pi_n\, v = \pi_n\, v' = \perp$, in which case we are done by (3.20), or there exists $(w, w') \in [\![\tau]\!]_\Xi\varphi'\,\Delta'$ such that $\pi_n\, v = \lfloor w \rfloor$ and $\pi_n\, v' = \lfloor w' \rfloor$. But in the latter case, (3.20) gives that $\pi_{n+1}(in_\mu\, v) = \lfloor in_\mu\, w \rfloor$ and $\pi_{n+1}(in_\mu\, v') = \lfloor in_\mu\, w' \rfloor$ where $(in_\mu\, w, in_\mu\, w') \in G(\varphi')(\Delta')$.

Finally, to appeal to Banach's fixed-point theorem and conclude that the interpretation of recursive types is well-defined, we need to ensure that the complete metric space $\mathcal{T}$ is non-empty. We have already observed that, e.g., the constant function $\lambda\Delta.\emptyset$, used to interpret the type $\mathsf{0}$, belongs to $\mathcal{T}$. □

We need the following weakening and substitution properties, easily proved by induction on τ:

Proposition 4.30.

1. Let τ be a type such that $\Xi \vdash \tau$, and let $\alpha \notin \Xi$. For all φ in $\mathcal{T}^\Xi$ and $\nu \in \mathcal{T}$,

$$[\![\tau]\!]_\Xi\varphi = [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu]\,.$$

2. Let τ and τ' be types such that $\Xi, \alpha \vdash \tau$ and $\Xi \vdash \tau'$. For all φ in $\mathcal{T}^\Xi$,

$$[\![\tau[\tau'/\alpha]]\!]_\Xi\varphi = [\![\tau]\!]_{\Xi,\alpha}(\varphi[\alpha \mapsto [\![\tau']\!]_\Xi\varphi])\,.$$

Corollary 4.31. For $\Xi, \alpha \vdash \tau$ and $\varphi \in \mathcal{T}^{\Xi}$,

$$[\![\mu\alpha.\tau]\!]_{\Xi}\varphi = \lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi\,\Delta\,\}\,.$$

4.4. *Interpretation of terms*

As for the interpretation of terms, we must show that the untyped meaning of a typed term is related to itself at the appropriate type. We first show that *comp* respects the operations η and $\star$ of the monad T (defined on page 7).

Definition 4.32. For $\nu \in \mathcal{T}$ and $\xi \in \mathcal{T}_T$ and $\Delta \in \mathcal{W}$, let $\nu \stackrel{\Delta}{\to} \xi$ be the binary relation on functions $V \to TV$ defined by

$$\nu \stackrel{\Delta}{\to} \xi = \{\,(f, f') \mid \forall\Delta_1 \geq \Delta.\forall(v, v') \in \nu(\Delta_1).\,(f\, v, f'\, v') \in \xi(\Delta_1)\,\}\,.$$

Proposition 4.33. Let $\nu, \nu_1, \nu_2 \in \mathcal{T}$ and $\Delta \in \mathcal{W}$.

1. If $(v, v') \in \nu(\Delta)$, then $(\eta\, v, \eta\, v') \in \mathit{comp}(\nu)(\Delta)$.
2. If $(c, c') \in \mathit{comp}(\nu_1)(\Delta)$ and $(f, f') \in \nu_1 \stackrel{\Delta}{\to} \mathit{comp}(\nu_2)$, then

$$(c \star f,\, c' \star f') \in \mathit{comp}(\nu_2)(\Delta)\,.$$

Proof.

1: Assume that $(v, v') \in \nu(\Delta)$. By definition, $\eta\, v = \lambda k.\lambda s.\, k\, v\, s$, and similarly for $\eta\, v'$. To show that $(\eta\, v, \eta\, v') \in \mathit{comp}(\nu)(\Delta)$, let $\Delta_1 \geq \Delta$ and $(k, k') \in \mathit{cont}(\nu)(\Delta_1)$ and $(s, s') \in \mathit{states}(\Delta_1)$; we must show that $((\eta\, v)\, k\, s,\, (\eta\, v')\, k'\, s') \in R_{Ans}$, i.e., that $(k\, v\, s,\, k'\, v'\, s') \in R_{Ans}$. But this follows directly from the definition of $\mathit{cont}(\nu)(\Delta_1)$ since $(v, v') \in \nu(\Delta) \subseteq \nu(\Delta_1)$ by monotonicity.

2: Assume that $(c, c') \in \mathit{comp}(\nu_1)(\Delta)$ and $(f, f') \in \nu_1 \stackrel{\Delta}{\to} \mathit{comp}(\nu_2)$. By definition, $c \star f = \lambda k.\lambda s.\, c\,(\lambda v.\lambda s_1.f\, v\, k\, s_1)\, s$, and similarly for $c' \star f'$. To show that $(c \star f, c' \star f') \in \mathit{comp}(\nu_2)(\Delta)$, let $\Delta_1 \geq \Delta$ and $(k, k') \in \mathit{cont}(\nu_2)(\Delta_1)$ and $(s, s') \in \mathit{states}(\Delta_1)$; we must show that $((c \star f)\, k\, s,\, (c' \star f')\, k'\, s') \in R_{Ans}$, i.e., that

$$(c\,(\lambda v.\lambda s_1.f\, v\, k\, s_1)\, s,\, c'\,(\lambda v'.\lambda s_1'.f'\, v'\, k'\, s_1')\, s') \in R_{Ans}\,.$$

Since $(c, c') \in \mathit{comp}(\nu_1)(\Delta)$ and $\Delta_1 \geq \Delta$ and $(s, s') \in \mathit{states}(\Delta_1)$, it suffices to show that $(\lambda v.\lambda s_1.f\, v\, k\, s_1,\, \lambda v'.\lambda s_1'.f'\, v'\, k'\, s_1') \in \mathit{cont}(\nu_1)(\Delta_1)$. So, let $\Delta_2 \geq \Delta_1$ and $(v, v') \in \nu_1(\Delta_2)$ and $(s_1, s_1') \in \mathit{states}(\Delta_2)$; we must show that $(f\, v\, k\, s_1,\, f'\, v'\, k'\, s_1') \in R_{Ans}$. First, $(f\, v, f'\, v') \in \mathit{comp}(\nu_2)(\Delta_2)$ by assumption on (f, f'). By monotonicity of $\mathit{cont}(\nu_2)$ we have $(k, k') \in \mathit{cont}(\nu_2)(\Delta_2)$, and by assumption, $(s_1, s_1') \in \mathit{states}(\Delta_2)$. Therefore, it follows from the definition of $\mathit{cont}(\nu_2)(\Delta_2)$ that $(f\, v\, k\, s_1,\, f'\, v'\, k'\, s_1') \in R_{Ans}$. □

Definition 4.34. For every term environment $\Xi \vdash \Gamma$, every $\varphi \in \mathcal{T}^{\Xi}$, and every $\Delta \in \mathcal{W}$, let $[\![\Gamma]\!]_{\Xi}\varphi\,\Delta$ be the binary relation on $V^{\mathrm{dom}(\Gamma)}$ defined by

$$[\![\Gamma]\!]_{\Xi}\varphi\,\Delta = \{\,(\rho, \rho') \mid \forall x \in \mathrm{dom}(\Gamma).\,(\rho(x), \rho'(x)) \in [\![\Gamma(x)]\!]_{\Xi}\varphi\,\Delta\,\}\,.$$

Definition 4.35. Two typed terms $\Xi \mid \Gamma \vdash t : \tau$ and $\Xi \mid \Gamma \vdash t' : \tau$ of the same type are *semantically related*, written $\Xi \mid \Gamma \models t \sim t' : \tau$, if for all $\varphi \in \mathcal{T}^{\Xi}$, all $\Delta \in \mathcal{W}$, and all $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi}\varphi\,\Delta$,

$$\big([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho, [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho'\big) \in \mathit{comp}([\![\tau]\!]_{\Xi}\varphi)(\Delta)\,.$$

Theorem 4.36 (Fundamental Theorem). Every typed term is semantically related to itself: if $\Xi \mid \Gamma \vdash t : \tau$, then $\Xi \mid \Gamma \models t \sim t : \tau$.

Proof. By showing the stronger property that semantic relatedness is preserved by all the term constructs. We use Proposition 4.33 to avoid tedious reasoning about continuations and states for the term constructs that do not directly involve references. Below are some illustrative cases.

1. If $\Gamma(x) = \tau$, then $\Xi \mid \Gamma \models x \sim x : \tau$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi}\varphi\,\Delta$ be given. Then $(\rho(x), \rho'(x)) \in [\![\tau]\!]_{\Xi}\varphi\,\Delta$. Therefore, by Proposition 4.33(1),

$$([\![x]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![x]\!]_{\mathrm{dom}(\Gamma)}\rho') = (\eta(\rho(x)),\ \eta(\rho'(x))) \in \mathit{comp}([\![\tau]\!]_{\Xi}\varphi)(\Delta)\,,$$

as required.

2. If $\Xi \mid \Gamma \models t \sim t' : \mu\alpha.\tau$, then $\Xi \mid \Gamma \models \mathsf{unfold}\, t \sim \mathsf{unfold}\, t' : \tau[\mu\alpha.\tau/\alpha]$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi}\varphi\,\Delta$ be given. Recall that $[\![\mathsf{unfold}\, t]\!]_{\mathrm{dom}(\Gamma)}\rho = [\![t]\!]_{\mathrm{dom}(\Gamma)}\rho \star f$ and $[\![\mathsf{unfold}\, t']\!]_{\mathrm{dom}(\Gamma)}\rho' = [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho' \star f$ where $f : V \to TV$ is given by

$$f\,v = \begin{cases} \eta(v_0) & \text{if } v = \mathit{in}_{\mu}\, v_0 \\ \mathit{error} & \text{otherwise.} \end{cases}$$

By assumption, $([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in \mathit{comp}([\![\mu\alpha.\tau]\!]_{\Xi}\varphi)(\Delta)$. Therefore, by Proposition 4.33(2), it suffices to show that

$$(f, f) \in [\![\mu\alpha.\tau]\!]_{\Xi}\varphi \overset{\Delta}{\to} \mathit{comp}([\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi)\,.$$

To see this, let $\Delta_1 \geq \Delta$ and $(v, v') \in [\![\mu\alpha.\tau]\!]_{\Xi}\varphi\,\Delta_1$ be given; we must show that $(f\,v, f\,v') \in \mathit{comp}([\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi)(\Delta_1)$. By Corollary 4.31, $(v, v') = (\mathit{in}_{\mu}\, v_0,\ \mathit{in}_{\mu}\, v_0')$ for some $(v_0, v_0') \in [\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi\,\Delta_1$. But then by Proposition 4.33(1),

$$(f\,v, f\,v') = (\eta(v_0), \eta(v_0')) \in \mathit{comp}([\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi)(\Delta_1)\,,$$

as required.

3. If $\Xi, \alpha \mid \Gamma \models t \sim t' : \tau$ and $\Xi \vdash \Gamma$, then $\Xi \mid \Gamma \models \Lambda\alpha.t \sim \Lambda\alpha.t' : \forall\alpha.\tau$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi}\varphi\,\Delta$ be given. Recall that $[\![\Lambda\alpha.t]\!]_{\mathrm{dom}(\Gamma)}\rho = \eta(\mathit{in}_{\forall}([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho))$ and that $[\![\Lambda\alpha.t']\!]_{\mathrm{dom}(\Gamma)}\rho' = \eta(\mathit{in}_{\forall}([\![t']\!]_{\mathrm{dom}(\Gamma)}\rho'))$. We must therefore show that

$$\big(\eta(\mathit{in}_{\forall}([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho)),\ \eta(\mathit{in}_{\forall}([\![t']\!]_{\mathrm{dom}(\Gamma)}\rho'))\big) \in \mathit{comp}([\![\forall\alpha.\tau]\!]_{\Xi}\varphi)(\Delta)\,.$$

By Proposition 4.33(1) it suffices to show that

$$\big(\mathit{in}_{\forall}([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho),\ \mathit{in}_{\forall}([\![t']\!]_{\mathrm{dom}(\Gamma)}\rho')\big) \in [\![\forall\alpha.\tau]\!]_{\Xi}\varphi\,\Delta\,.$$

We proceed according to the definition of $[\![\forall\alpha.\tau]\!]_{\Xi}$. Let $\nu \in \mathcal{T}$ be given; we must show that $([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in \mathit{comp}([\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])(\Delta)$. But this follows from the

assumption that $\Xi, \alpha \mid \Gamma \models t \sim t' : \tau$ since Proposition 4.30(1) (weakening) gives that $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu]$.

4. If $\Xi \mid \Gamma \models t \sim t' : \forall\alpha.\tau_0$ and $\Xi \vdash \tau_1$, then $\Xi \mid \Gamma \models t\,[\tau_1] \sim t'\,[\tau_1] : \tau_0[\tau_1/\alpha]$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_\Xi\varphi\,\Delta$ be given. Recall that $[\![t\,[\tau_1]]\!]_{\mathrm{dom}(\Gamma)}\rho = [\![t]\!]_{\mathrm{dom}(\Gamma)}\rho \star g$ and $[\![t'\,[\tau_1]]\!]_{\mathrm{dom}(\Gamma)}\rho' = [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho' \star g$ where $g : V \to TV$ is given by

$$g\,v = \begin{cases} c & \text{if } v = in_\forall\, c \\ \mathit{error} & \text{otherwise.} \end{cases}$$

By assumption, $([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in \mathit{comp}([\![\forall\alpha.\tau]\!]_\Xi\varphi)(\Delta)$. Therefore, by Proposition 4.33(2), it suffices to show that

$$(g, g) \in [\![\forall\alpha.\tau]\!]_\Xi\varphi \stackrel{\Delta}{\to} \mathit{comp}([\![\tau_0[\tau_1/\alpha]]\!]_\Xi\varphi)\,.$$

To see this, let $\Delta_1 \geq \Delta$ and $(v, v') \in [\![\forall\alpha.\tau]\!]_\Xi\varphi\,\Delta_1$ be given; we must show that $(g\,v, g\,v') \in \mathit{comp}([\![\tau_0[\tau_1/\alpha]]\!]_\Xi\varphi)(\Delta_1)$. By the definition of $[\![\forall\alpha.\tau_0]\!]_\Xi$ we know that $(v, v') = (in_\forall\, c,\ in_\forall\, c')$ for some c and c' satisfying that $(c, c') \in \mathit{comp}([\![\tau_0]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])(\Delta_1)$ for all $\nu \in \mathcal{T}$. Now choose $\nu = [\![\tau_1]\!]_\Xi\varphi$: Proposition 4.30(2) (substitution) gives that

$$(g\,v, g\,v') = (c, c') \in \mathit{comp}([\![\tau_0[\tau_1/\alpha]]\!]_\Xi\varphi)(\Delta_1)\,,$$

as required.

5. If $\Xi \mid \Gamma \models t \sim t' : \tau$, then $\Xi \mid \Gamma \models \mathsf{ref}\,t \sim \mathsf{ref}\,t' : \mathsf{ref}\,\tau$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_\Xi\varphi\,\Delta$ be given. Recall that $[\![\mathsf{ref}\,t]\!]_{\mathrm{dom}(\Gamma)}\rho = [\![t]\!]_{\mathrm{dom}(\Gamma)}\rho \star \lambda v.\ \mathit{alloc}\,v$ and $[\![\mathsf{ref}\,t']\!]_{\mathrm{dom}(\Gamma)}\rho' = [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho' \star \lambda v.\ \mathit{alloc}\,v$. By assumption, $([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in \mathit{comp}([\![\tau]\!]_\Xi\varphi)(\Delta)$. Therefore, by Proposition 4.33(2), it suffices to show that

$$(\mathit{alloc}, \mathit{alloc}) \in [\![\tau]\!]_\Xi\varphi \stackrel{\Delta}{\to} \mathit{comp}([\![\mathsf{ref}\,\tau]\!]_\Xi\varphi)\,.$$

To see this, let $\Delta_1 \geq \Delta$ and $(v, v') \in [\![\tau]\!]_\Xi\varphi\,\Delta_1$ be given; we must show that $(\mathit{alloc}\ v, \mathit{alloc}\ v') \in \mathit{comp}([\![\mathsf{ref}\,\tau]\!]_\Xi\varphi)(\Delta_1)$. We proceed according to the definition of *comp*. Let $\Delta_2 \geq \Delta_1$ and $(k, k') \in \mathit{cont}([\![\mathsf{ref}\,\tau]\!]_\Xi\varphi)(\Delta_2)$ and $(s, s') \in \mathit{states}(\Delta_2)$ be given; we must show that

$$(\mathit{alloc}\ v\,k\,s,\ \mathit{alloc}\ v'\,k'\,s') \in R_{Ans}\,. \tag{4.6}$$

We know that $\mathrm{dom}(s) = \mathrm{dom}(s') = \mathrm{dom}(\Delta_2)$. Let $l_0 \in \mathbb{N}_0$ be the least number such that $l_0 \notin \mathrm{dom}(\Delta_2)$; then

$$\mathit{alloc}\ v\,k\,s = k\ \lambda_{l_0}\,(s[l_0 \mapsto v]) \tag{4.7}$$

$$\mathit{alloc}\ v'\,k'\,s' = k'\ \lambda_{l_0}\,(s'[l_0 \mapsto v'])\,. \tag{4.8}$$

We now aim to use the assumption that $(k, k') \in \mathit{cont}([\![\mathsf{ref}\,\tau]\!]_\Xi\varphi)(\Delta_2)$. Define $\Delta_3 = \Delta_2[l_0 \mapsto i^{-1}([\![\tau]\!]_\Xi\varphi)]$ (where $i^{-1} = i^{-1}$ is the isomorphism associated with the recursive metric-space equation.) Clearly $\Delta_3 \geq \Delta_2$ since $l_0 \notin \mathrm{dom}(\Delta_2)$. Then $(\lambda_{l_0}, \lambda_{l_0}) \in [\![\mathsf{ref}\,\tau]\!]_\Xi\varphi\,\Delta_3 = \mathit{ref}([\![\tau]\!]_\Xi\varphi)(\Delta_3)$ since for all $\Delta_4 \geq \Delta_3$ we have

$$i(\Delta_4(l_0)) = i(\Delta_3(l_0)) = i(i^{-1}([\![\tau]\!]_\Xi\varphi)) = [\![\tau]\!]_\Xi\varphi\,.$$

Furthermore, $(s[l_0 \mapsto v],\, s'[l_0 \mapsto v']) \in \mathit{states}(\Delta_3)$: for $l \in \mathrm{dom}(\Delta_2)$ we have

$$\begin{aligned}((s[l_0 \mapsto v])(l),\, (s'[l_0 \mapsto v'])(l)) = (s(l), s'(l)) &\in i(\Delta_2(l))\,(\Delta_2)\\ &= i(\Delta_3(l))\,(\Delta_2)\\ &\subseteq i(\Delta_3(l))\,(\Delta_3)\,,\end{aligned}$$

by monotonicity of $i(\Delta_3(l)) \in \mathcal{W} \to_{mon} \mathit{CURel}(V)$, and also,

$$\begin{aligned}((s[l_0 \mapsto v])(l_0),\, (s'[l_0 \mapsto v'])(l_0)) = (v, v') &\in [\![\tau]\!]_{\Xi\varphi}\,\Delta_1\\ &\subseteq [\![\tau]\!]_{\Xi\varphi}\,\Delta_3\,,\end{aligned}$$

by monotonicity of $[\![\tau]\!]_{\Xi\varphi}$. All in all, $(s[l_0 \mapsto v],\, s'[l_0 \mapsto v']) \in \mathit{states}(\Delta_3)$. Therefore, (4.7), (4.8), and the assumption that $(k, k') \in \mathit{cont}([\![\mathsf{ref}\,\tau]\!]_{\Xi\varphi})(\Delta_2)$ gives (4.6), as required.

6. If $\Xi \mid \Gamma \models t \sim t' : \mathsf{ref}\,\tau$, then $\Xi \mid \Gamma \models\, !t \sim\, !t' : \tau$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi\varphi}\,\Delta$ be given. Using exactly the same reasoning as in the previous case, we see that it suffices to show that

$$(\mathit{lookup}, \mathit{lookup}) \in [\![\mathsf{ref}\,\tau]\!]_{\Xi\varphi} \xrightarrow{\Delta} \mathit{comp}([\![\tau]\!]_{\Xi\varphi})\,.$$

Therefore, let $\Delta_1 \geq \Delta$ and $(v, v') \in [\![\mathsf{ref}\,\tau]\!]_{\Xi\varphi}\,\Delta_1$ be given; we must show that

$$(\mathit{lookup}\ v, \mathit{lookup}\ v') \in \mathit{comp}([\![\tau]\!]_{\Xi\varphi})(\Delta_1)$$

. According to the definition of $[\![\mathsf{ref}\,\tau]\!]_{\Xi\varphi} = \mathit{ref}([\![\tau]\!]_{\Xi\varphi})$ there are two cases: either $v = v' = \lambda_l$ for some $l \in \mathrm{dom}(\Delta_1)$, or $v = v' = \lambda_l^{n+1}$ for some $n \in \omega$ and $l \in \mathrm{dom}(\Delta_1)$. Assume that we are in the latter case; the former case is similar, but easier.

We proceed according to the definition of *comp*. So, let $\Delta_2 \geq \Delta_1$ and $(k, k') \in \mathit{cont}([\![\tau]\!]_{\Xi\varphi})(\Delta_2)$ and $(s, s') \in \mathit{states}(\Delta_2)$ be given; we must show that

$$(\mathit{lookup}\ v\ k\ s,\ \mathit{lookup}\ v'\ k'\ s') \in R_{Ans}\,. \tag{4.9}$$

By the definition of *ref* we have $v = v' = \lambda_l^{n+1}$ where $i(\Delta_1(l))(\Delta_2) \stackrel{n}{=} [\![\tau]\!]_{\Xi\varphi}\,\Delta_2$. Since $(s, s') \in \mathit{states}(\Delta_2)$ we know that

$$(s(l), s'(l)) \in i(\Delta_2(l))(\Delta_2)\,.$$

Since $l \in \mathrm{dom}(\Delta_1)$ and $\Delta_2 \geq \Delta_1$, we have

$$i(\Delta_2(l))(\Delta_2) = i(\Delta_1(l))(\Delta_2) \stackrel{n}{=} [\![\tau]\!]_{\Xi\varphi}\,\Delta_2\,.$$

There are therefore two cases: either $\pi_n(s(l)) = \pi_n(s'(l)) = \bot$, or $(\pi_n(s(l)),\, \pi_n(s'(l))) = (\lfloor v_0 \rfloor, \lfloor v_0' \rfloor)$ where $(v_0, v_0') \in [\![\tau]\!]_{\Xi\varphi}\,\Delta_2$. In the first case, the definition of *lookup* gives

$$(\mathit{lookup}\ v\ k\ s,\ \mathit{lookup}\ v'\ k'\ s') = (\bot, \bot) \in R_{Ans}\,.$$

and we are done. In the second case, the definition of *lookup* gives

$$(\mathit{lookup}\ v\ k\ s,\ \mathit{lookup}\ v'\ k'\ s') = (k\ v_0\ s,\ k'\ v_0'\ s')\,. \tag{4.10}$$

By assumption, $(k, k') \in \mathit{cont}([\![\tau]\!]_{\Xi\varphi})(\Delta_2)$. Therefore, by definition of *cont*, we have $(k\ v_0\ s,\ k'\ v_0'\ s') \in R_{Ans}$. From (4.10) we then conclude (4.9), and we are done.

7. For many of the remaining cases, the core of the proof is to show that *comp* preserves

logical relatedness. For example:

$$(\llbracket t_1 \rrbracket\rho,\, \llbracket t_1' \rrbracket\rho') \in \mathit{comp}(\llbracket \tau \to \tau' \rrbracket_\Xi \phi)(\Delta) \;\wedge\; (\llbracket t_2 \rrbracket\rho,\, \llbracket t_2' \rrbracket\rho') \in \mathit{comp}(\llbracket \tau \rrbracket_\Xi \phi)(\Delta) \implies (\llbracket t_1\, t_2 \rrbracket\rho,\, \llbracket t_1'\, t_2' \rrbracket\rho') \in \mathit{comp}(\llbracket \tau' \rrbracket_\Xi \phi)(\Delta) \quad (4.11)$$

$$(\llbracket t_1 \rrbracket\rho,\, \llbracket t_1' \rrbracket\rho') \in \mathit{comp}(\llbracket \tau_1 \rrbracket_\Xi \phi)(\Delta) \;\wedge\; (\llbracket t_2 \rrbracket\rho,\, \llbracket t_2' \rrbracket\rho') \in \mathit{comp}(\llbracket \tau_2 \rrbracket_\Xi \phi)(\Delta) \implies (\llbracket (t_1,\, t_2) \rrbracket\rho,\, \llbracket (t_1',\, t_2') \rrbracket\rho') \in \mathit{comp}(\llbracket \tau_1 \times \tau_2 \rrbracket_\Xi \phi)(\Delta) \quad (4.12)$$

□

Corollary 4.37.

1. If $\emptyset \mid \emptyset \vdash t : \tau$ is a closed term of type τ, then $\llbracket t \rrbracket_\emptyset \emptyset \neq \mathit{error}$.
2. If $\emptyset \mid \emptyset \vdash t : \mathsf{int}$ is a closed term of type int, then $\llbracket t \rrbracket^{\mathrm{p}} \neq \mathit{error}_{Ans}$.

Proof. 1. The theorem gives that $(\llbracket t \rrbracket_\emptyset \emptyset,\, \llbracket t \rrbracket_\emptyset \emptyset) \in \mathit{comp}(\llbracket \tau \rrbracket_\emptyset \emptyset)(\emptyset)$. Now let $s_{init} \in S$ be the empty store, and let $k_0 \in K$ be the continuation that always gives the answer 0, i.e., $k_0\, v\, s = \iota_1\, 0$ for all v and s. It follows immediately from the definitions that $(k_0, k_0) \in \mathit{cont}(\llbracket \tau \rrbracket_\emptyset \emptyset)(\emptyset)$ and that $(s_{init}, s_{init}) \in \mathit{states}(\emptyset)$. Therefore, $(\llbracket t \rrbracket_\emptyset \emptyset\, k_0\, s_{init}, \llbracket t \rrbracket_\emptyset \emptyset\, k_0\, s_{init}) \in R_{Ans}$. By the definition of R_{Ans} we must then have $\llbracket t \rrbracket_\emptyset \emptyset\, k_0\, s_{init} \neq \mathit{error}_{Ans}$ which implies that $\llbracket t \rrbracket_\emptyset \emptyset \neq \mathit{error}$.

2. Recall that $\llbracket t \rrbracket^{\mathrm{p}} = \llbracket t \rrbracket_\emptyset \emptyset\, k_{init}\, s_{init}$ where $s_{init} \in S$ is the empty store and where

$$k_{init} = \lambda v.\lambda s. \begin{cases} \lfloor \iota_1\, m \rfloor & \text{if } v = \mathit{in}_{\mathbb{Z}}(m) \\ \mathit{error}_{Ans} & \text{otherwise}\,. \end{cases}$$

It is easy to show that $(k_{init}, k_{init}) \in \mathit{cont}(\llbracket \mathsf{int} \rrbracket_\emptyset \emptyset)(\emptyset)$. Furthermore, as already argued, $(s_{init}, s_{init}) \in \mathit{states}(\emptyset)$.

The theorem gives that $(\llbracket t \rrbracket_\emptyset \emptyset,\, \llbracket t \rrbracket_\emptyset \emptyset) \in \mathit{comp}(\llbracket \tau \rrbracket_\emptyset \emptyset)(\emptyset)$, which then implies that

$$(\llbracket t \rrbracket^{\mathrm{p}},\, \llbracket t \rrbracket^{\mathrm{p}}) = (\llbracket t \rrbracket_\emptyset \emptyset\, k_{init}\, s_{init},\, \llbracket t \rrbracket_\emptyset \emptyset\, k_{init}\, s_{init}) \in R_{Ans}\,.$$

Therefore $\llbracket t \rrbracket^{\mathrm{p}} \neq \mathit{error}_{Ans}$ by definition of R_{Ans}. □

5. On the need for approximate locations

Consider the following, tentative interpretation of reference types:

$$\mathit{ref}(\nu)(\Delta) = \{(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \;\wedge\; \forall \Delta_1 \geq \Delta.\, i(\Delta(l))(\Delta_1) = \nu(\Delta_1)\}\,.$$

As discussed on page 29, this function $\mathit{ref}(\nu)$ is not non-expansive and hence does not belong to the space $\mathcal{T}$ of semantic types. Accordingly, we have introduced approximate locations and settled for a more complicated interpretation of reference types.

Still, it might be that one could interpret semantic types and worlds differently, perhaps by solving a recursive equation in a different category, in order to achieve a "simple" interpretation of reference types above.

In this section we show, in an abstract setup, and under some relatively mild assumptions, that such a simple interpretation is impossible. The proof works for an arbitrary set V of "syntactic values", not required to be a domain, and there is no requirement that

the elements λ_l are particular values such as integers. The intuitive idea of the proof is that that semantic worlds (as expressed by a solution to the recursive equations between types and worlds) can in a certain sense be used to encode recursive types, and that recursive types and "simple" reference types are incompatible.

Let V be a set, and let $GRel(V)$ be a set of subsets of $V \times V$ which is closed under finite intersections and contains the empty set. We think of $GRel(V)$ as the set of "good" relations on V. Example: In the setup of the paper, V is the universal predomain from Proposition 3.2 and $GRel(V)$ is $CURel(V)$.

Now assume that $\hat{\mathcal{T}}$ is a set and that $i : \hat{\mathcal{T}} \to \mathcal{T}$ is a bijection into some subset $\mathcal{T}$ of $(Loc \rightharpoonup_{fin} \hat{\mathcal{T}}) \to_{mon} GRel(V)$. That is, $\mathcal{T}$ is a subset of those functions that are monotone with respect to the "extension" order on $\mathcal{W} = Loc \rightharpoonup_{fin} \hat{\mathcal{T}}$ and the inclusion order on $GRel(V)$. We assume that $\mathcal{T}$ contains all the constant functions, and that for all $\nu_1 \in \mathcal{T}$ and $\nu_2 \in \mathcal{T}$ the monotone function $\nu_1 \wedge \nu_2$ given by

$$(\nu_1 \wedge \nu_2)(\Delta) = \nu_1(\Delta) \cap \nu_2(\Delta)$$

belongs to $\mathcal{T}$.

The idea is that $\mathcal{T}$ should be thought of as the set of "good" monotone functions. In the concrete metric-space setup, $\mathcal{T}$ consists of the non-expansive, monotone functions.

We now add the following assumption:

Assumption 5.1. There is an injective function $c : V \to V$ such that for every $l \in \omega$, the monotone function ν_l given by

$$\nu_l(\Delta) = \begin{cases} \{(c(v), c(v')) \mid (v, v') \in i(\Delta(l))(\Delta)\} & \text{if } l \in \mathrm{dom}(\Delta) \\ \emptyset & \text{otherwise.} \end{cases}$$

belongs to $\mathcal{T}$.

For every $R \in GRel(V)$ we define $c(R) = \{(c(v), c(v') \mid (v, v') \in R\}$; with that notation, the assumption implies that $\nu_l(\Delta) = c(i(\Delta(l))(\Delta))$ when $l \in \mathrm{dom}(\Delta)$. The idea is that ν_l is a "self-application" operator, up to some function c which could potentially be the identity function. We furthermore assume that for every $R \in GRel(V)$, the relation $c(R)$ belongs to $GRel(V)$.

In the concrete model presented in this paper, we can choose $c = in_\mu$. Then the function ν_l defined above is indeed non-expansive (and hence belongs to $\mathcal{T}$): Lemma A.1 gives that if $\Delta =_n \Delta'$ for some $n > 0$, then $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$. The same lemma furthermore gives that if l belongs to these two domains, then $i(\Delta) =_{n-1} i(\Delta')$ and hence $i(\Delta)(\Delta) =_{n-1} i(\Delta')(\Delta')$. Finally, by the definition of the projection functions, $in_\mu(i(\Delta)(\Delta)) =_n in_\mu(i(\Delta')(\Delta'))$, as desired.

Returning to the abstract setup, we now show that ν_l has the following "universality" property:

Proposition 5.2. For every $\nu \in \mathcal{T}$ there exists $\Delta_0 \in \mathcal{W}$ such that

$$\nu_l(\Delta) = c(\nu(\Delta)) \qquad \text{for all } \Delta \geq \Delta_0.$$

Proof. Choose $\Delta_0 = [l \mapsto i^{-1}(\nu)]$. Then for all $\Delta \geq \Delta_0$ we have

$$\nu_l(\Delta) = c(i(\Delta(l))(\Delta) = \nu(\Delta). \qquad \square$$

Even though the proof of the proposition is about as simple as it could be, the proposition itself is in our opinion slightly counterintuitive: how can it possibly be that every type $\nu \in \mathcal{T}$ "agrees with ν_l" somewhere? One answer is that the set of worlds $\mathcal{W}$ is not just some abstract preorder; it is a set which is so large that it includes $\mathcal{T}$ itself.

It follows that *all* endofunctions on $\mathcal{T}$ have "fixed points up to c" in a certain sense:

Corollary 5.3. For every function F from $\mathcal{T}$ to $\mathcal{T}$ there exists some $\nu' \in \mathcal{T}$ and $\Delta_0 \in \mathcal{W}$ such that

$$\nu'(\Delta) = c(F(\nu')(\Delta)) \qquad \text{for all } \Delta \geq \Delta_0.$$

Furthermore, for every location $l \in \omega$ one can choose the ν' and Δ_0 above such that $\mathrm{dom}(\Delta_0) = \{l\}$.

Proof. Choose $\nu = F(\nu_l)$ in the previous proposition. $\square$

Using these results, we show that the simple interpretation of reference types is impossible:

Proposition 5.4. Let $\lambda_{(-)} : \omega \to V$ be an arbitrary, injective function. There is no function $\mathit{ref} : \mathcal{T} \to \mathcal{T}$ satisfying

$$\mathit{ref}(\nu)(\Delta) = \{(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \ \wedge \ \forall \Delta_1 \geq \Delta.\, i(\Delta(l))(\Delta_1) = \nu(\Delta_1)\}$$

for all $\nu \in \mathcal{T}$ and $\Delta \in \mathcal{W}$.

Proof. Assume that *ref* is such a function. We derive a contradiction.

First, observe that every singleton relation $\{(\lambda_l, \lambda_l)\}$ belongs to $\mathit{GRel}(V)$: for an arbitrary $\nu \in \mathcal{T}$ we have $\{(\lambda_l, \lambda_l)\} = \mathit{ref}(\nu)([l \mapsto i^{-1}(\nu)])$, so $\{(\lambda_l, \lambda_l)\}$ is in the image of $\mathit{ref}(\nu) \in \mathcal{T}$. Then, by assumption on c, the relation $\{(c(\lambda_l),\, c(\lambda_l))\}$ also belongs to $\mathit{GRel}(V)$. Finally, the function $\mathit{single}(c(\lambda_l)) = \lambda\Delta.\{(c(\lambda_l),\, c(\lambda_l))\}$ belongs to $\mathcal{T}$ since we have assumed that $\mathcal{T}$ contains all the constant functions.

Let $l \in \omega$ be an arbitrarily chosen number and define $F : \mathcal{T} \to \mathcal{T}$ by $F(\nu) = \mathit{ref}(\nu \wedge \mathit{single}(c(\lambda_l)))$. Corollary 5.3 gives $\nu \in \mathcal{T}$ and $\Delta_0 \in \mathcal{W}$ such that

$$\nu(\Delta) = \mathit{in}_\mu(F(\nu)(\Delta)) = \mathit{in}_\mu(\mathit{ref}(\nu)(\Delta) \cap \{(c(\lambda_l),\, c(\lambda_l))\})$$

for all $\Delta \geq \Delta_0$. By the last clause of the corollary we can assume that $l \notin \mathrm{dom}(\Delta_0)$ (by choosing $\mathrm{dom}(\Delta_0) = \{l'\}$ for some $l' \neq l$). Now let $\Delta_0' = \Delta_0[l \mapsto i^{-1}(\emptyset)]$; we aim to show that

$$(\lambda_l, \lambda_l) \notin F(\nu)(\Delta) \qquad \text{for all } \Delta \geq \Delta_0'. \tag{5.1}$$

To that end, assume that $(\lambda_l, \lambda_l) \in F(\nu)(\Delta)$ for some $\Delta \geq \Delta_0$'. First, the definition of F and the assumption on *ref* give that $i(\Delta(l))(\Delta) = (\nu \wedge \mathit{single}(c(\lambda_l))(\Delta)$ (here we use

that $\lambda_{(-)}$ is injective). Then, since $\Delta \geq \Delta'_0$,

$$\begin{aligned}\nu(\Delta) \cap \{(c(\lambda_l),\, c(\lambda_l))\} &= (\nu \wedge \mathit{single}(c(\lambda_l)))(\Delta)\\ &= i(\Delta(l))(\Delta)\\ &= i(\Delta'_0(l))(\Delta)\\ &= \emptyset,\end{aligned}$$

which means that $(c(\lambda_l),\, c(\lambda_l)) \notin \nu(\Delta)$. But $\nu(\Delta) = c(F(\nu)(\Delta))$ since $\Delta \geq \Delta'_0 \geq \Delta_0$, and we therefore conclude that $(\lambda_l, \lambda_l) \notin F(\nu)(\Delta)$, a contradiction. This shows (5.1).

In particular, taking $\Delta = \Delta'_0$ in (5.1), we get that $(\lambda_l, \lambda_l) \notin F(\nu)(\Delta'_0)$. But then, by definition of F and the assumption on *ref*, there is some $\Delta \geq \Delta'_0$ such that $i(\Delta'_0(l))(\Delta) \neq (\nu \wedge \mathit{single}(c(\lambda_l))(\Delta)$. Since $i(\Delta'_0(l))(\Delta) = \emptyset$ this means that $(\nu \wedge \mathit{single}(c(\lambda_l))(\Delta)$ is non-empty; in other words, that $(c(\lambda_l),\, c(\lambda_l)) \in \nu(\Delta)$. Therefore, since $\nu(\Delta) = c(F(\nu)(\Delta))$ and since c is injective, $(\lambda_l \lambda_l) \in F(\nu)(\Delta)$. But this contradicts (5.1). □

The proof above only depends on the existence of a particular "recursive type" $\nu(\Delta) = \mathit{in}_\mu(F(\nu)(\Delta))$. Accordingly, one can show a more syntactic variant of the proposition above that does *not* depend on Assumption 5.1: below we merely assume that $c : V \to V$ is some injective function satisfying that $c(R) \in \mathit{GRel}(V)$ for all $R \in \mathit{GRel}(V)$. The semantic type ν will then obtained as the interpretation of a syntactic recursive type.

In order to express the syntactic counterpart of the semantic type ν we add finite intersection types $\tau_1 \wedge \tau_2$ to the language.

Proposition 5.5. Let $\lambda_{(-)} : \omega \to V$ be an arbitrary, injective function. There is no interpretation function $[\![-]\!]_{(-)}$, mapping types in context Ξ to functions from $\mathcal{T}^\Xi$ to $\mathcal{T}$, that satisfies

$$\begin{aligned}[\![\alpha]\!]_\Xi\phi(\Delta) &= \phi(\alpha)(\Delta)\\ [\![\mathsf{ref}\,\tau]\!]_\Xi\phi(\Delta) &= \{(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \ \wedge\ \forall \Delta_1 \geq \Delta.\, i(\Delta(l))(\Delta_1) = [\![\tau]\!]_\Xi\phi(\Delta_1)\}\\ [\![\mu\alpha.\tau]\!]_\Xi\phi(\Delta) &= \{(c(v),\, c(v')) \mid (v, v') \in [\![\tau[\mu\alpha.\tau/\alpha]]\!]_\phi(\Delta)\}\\ [\![\tau_1 \wedge \tau_2]\!]_\phi(\Delta) &= [\![\tau_1]\!]_\Xi\phi(\Delta) \cap [\![\tau_2]\!]_\Xi\phi(\Delta)\end{aligned}$$

for all τ and Δ.

To show this, one essentially repeats the argument in the previous proof by taking

$$\nu = [\![\mu\alpha.(\mathsf{ref}\,\alpha \wedge \beta)]\!]_\beta[\beta \mapsto \mathit{single}(c(\lambda_l))].$$

6. Examples

The model can be used to prove the equivalences in Section 5 of our earlier work (Birkedal et al. 2009). More specifically, one can use the model to prove that some equivalences between different *functional* implementations of abstract data types are still valid in the presence of general references, and also prove some simple equivalences involving *imperative* abstract data types. (See Section 7 for more about extending the model to account properly for local state.) Here we only sketch two of these examples, as well

as a "non-example": an equivalence that cannot be shown because of the existence of approximated locations in the model.

6.1. *Encoding of existential types*

We use the usual encoding of existential types by means of universal types (Crary and Harper 2007): $\exists\alpha.\tau = \forall\beta.(\forall\alpha.\,\tau \to \beta) \to \beta$ where β does not occur free in τ. On the term level, define $\mathsf{pack}(\tau', t) = \Lambda\beta.\lambda x.x\,[\tau']\,t$ where x does not occur free in t.

The natural proof principle for relatedness at existential types is then derivable:

Proposition 6.1. Let $\Xi \mid \Gamma \vdash \mathsf{pack}(\tau_1, t) : \exists\alpha.\tau$ and $\Xi \mid \Gamma \vdash \mathsf{pack}(\tau_2, t') : \exists\alpha.\tau$. Assume that $\nu \in \mathcal{T}$ satisfies: for all $\Delta \in \mathcal{W}$, all $\phi \in \mathcal{T}^{\Xi}$, and all $(\rho, \rho') \in [\![\Gamma]\!]\phi(\Delta)$,

$$([\![t]\!]\rho,\, [\![t']\!]\rho') \in \mathit{comp}([\![\tau]\!](\phi[\alpha \mapsto \nu]))(\Delta).$$

Then

$$\Xi \mid \Gamma \models \mathsf{pack}(\tau_1, t) \sim \mathsf{pack}(\tau_2, t') : \exists\alpha.\tau.$$

Proof. By the congruence properties of semantic relatedness (see the proof of Theorem 4.36) it suffices to show

$$\Xi, \beta \mid \Gamma,\, x : \forall\alpha.\tau \to \beta \models x\,[\tau_1]\,t \sim x\,[\tau_2]\,t' : \beta.$$

(Here we have renamed β and x to ensure that they do not occur in Ξ and Γ, respectively.) To that end, let $\Delta \in \mathcal{W}$ and $\phi \in \mathcal{T}^{\Xi,\beta}$ and $(\rho, \rho') \in [\![\Gamma,\, x : \forall\alpha.\tau \to \beta]\!]\phi(\Delta)$ be given; we must show that

$$([\![x\,[\tau_1]\,t]\!]\rho,\, [\![x\,[\tau_2]\,t']\!]\rho') \in \mathit{comp}(\phi(\beta))(\Delta). \tag{6.1}$$

Since ρ and ρ' are related we have $\rho(c) = \mathit{in}_\forall\, c$ and $\rho'(c) = \mathit{in}_\forall\, c'$ for some c and c' with $(\mathit{in}_\forall\, c,\, \mathit{in}_\forall\, c') \in [\![\forall\alpha.\tau \to \beta]\!]\phi(\Delta)$. The interpretation of universal types then gives that

$$([\![x\,[\tau_1]]\!]\rho,\, [\![x\,[\tau_2]]\!]\rho') = (c, c') \in \mathit{comp}([\![\tau \to \beta]\!](\phi[\alpha \mapsto \nu]))(\Delta). \tag{6.2}$$

The assumption of the proposition (and weakening with respect to β and x) gives

$$([\![t]\!]\rho,\, [\![t']\!]\rho') \in \mathit{comp}([\![\tau]\!](\phi[\alpha \mapsto \nu]))(\Delta) \tag{6.3}$$

Then, using (4.11) on page 36, we get that (6.2) and (6.3) imply (6.1). □

6.2. *Example: imperative counter module*

Here we use the encoding of existential types presented just above. The type

$$\tau_{\mathrm{m}} = \exists\alpha.\,(1 \to \alpha) \times (\alpha \to 1) \times (\alpha \to \mathtt{int})$$

can be used to model imperative counter modules: the idea is that a value of type τ_{m} consists of some hidden type α, used to represent imperative counters, as well as three operations for creating a new counter, incrementing a counter, and reading the value of a counter, respectively.

Consider the following two module implementations, i.e., closed terms of type τ_{m}: $J = \mathsf{pack}(\mathsf{ref\,int},\, I)$ and $J' = \mathsf{pack}(\mathsf{ref\,int},\, I')$ where

$$I = (\lambda x.\mathsf{ref}\,0,\ \lambda x.\ x := !x + 1,\ \lambda x.!x)$$
$$I' = (\lambda x.\mathsf{ref}\,0,\ \lambda x.\ x := !x - 1,\ \lambda x.-(!x)$$

We now show by parametricity reasoning, i.e., by exploiting the universal quantification in the interpretation of universal types, that $\emptyset \mid \emptyset \models J \sim J' : \tau_{\mathrm{m}}$.

First, abbreviate $\tau = (1 \to \alpha) \times (\alpha \to 1) \times (\alpha \to \mathsf{int})$. By Proposition 6.1 it suffices to find $\nu \in \mathcal{T}$ such that $(\llbracket I \rrbracket, \llbracket I' \rrbracket) \in \mathit{comp}(\llbracket \tau \rrbracket[\alpha \to \nu])(\Delta)$ for all Δ.

Let $R = \{(\mathit{in}_Z(n), \mathit{in}_Z(-n)) \mid n \in \omega\}$ and choose $\nu = \mathit{ref}(\lambda\Delta.R)$. That is, we choose ν to be the semantic type of locations that point to R-related values. Now let $\Delta \in \mathcal{W}$ be given. By (4.12) on page 36 it suffices to show:

$$(\llbracket \lambda x.\mathsf{ref}\,0 \rrbracket,\ \llbracket \lambda x.\mathsf{ref}\,0 \rrbracket) \in \mathit{comp}(\llbracket 1 \to \alpha \rrbracket[\alpha \mapsto \nu])(\Delta) \tag{6.4}$$
$$(\llbracket \lambda x.\ x := !x + 1 \rrbracket,\ \llbracket \lambda x.\ x := !x - 1 \rrbracket) \in \mathit{comp}(\llbracket \alpha \to 1 \rrbracket[\alpha \mapsto \nu])(\Delta) \tag{6.5}$$
$$(\llbracket \lambda x.!x \rrbracket,\ \llbracket \lambda x.-!x \rrbracket) \in \mathit{comp}(\llbracket \alpha \to \mathsf{int} \rrbracket[\alpha \mapsto \nu])(\Delta) \tag{6.6}$$

In all three cases we use the first part of Proposition 4.33. Then, (6.5) and (6.6) follow straightforwardly from the definitions given our choice of ν.

Now consider (6.4). Let $\eta(\mathit{in}_{\to} f) = \llbracket \lambda x.\mathsf{ref}\,0 \rrbracket$; we must show that

$$(\mathit{in}_{\to} f,\ \mathit{in}_{\to} f) \in \llbracket 1 \to \alpha \rrbracket[\alpha \mapsto \nu](\Delta). \tag{6.7}$$

Let $\Delta_1 \geq \Delta$ and $(v, v') \in \llbracket 1 \rrbracket(\Delta)$ be given. We must now show that $(f(v),\, f(v')) \in \mathit{comp}(\nu)(\Delta_1)$. To that end, we unfold the definition of *comp*: let $\Delta_2 \geq \Delta_1$ and $(k, k') \in \mathit{cont}(\nu)(\Delta_2)$ and $(s, s') \in \mathit{states}(\Delta_2)$ be given. We must show that

$$(f(v)\,k\,s,\ f(v')\,k'\,s') \in R_{\mathit{Ans}}.$$

By the interpretation of allocation, $f(v)\,k\,s = k\,\lambda_l\,(s[l \mapsto \mathit{in}_Z 0])$ where l is the smallest number such that $l \notin \mathrm{dom}(s)$. Since $(s, s') \in \mathit{states}(\Delta_2)$ we have $\mathrm{dom}(s) = \mathrm{dom}(s') = \mathrm{dom}(\Delta_2)$, and therefore $f(v')\,k'\,s' = k'\,\lambda_l\,(s'[l \mapsto \mathit{in}_Z 0])$. (That is, the new location is the same one as above.) Since $(k, k') \in \mathit{cont}(\nu)(\Delta_2)$ it therefore suffices to find $\Delta_3 \geq \Delta_2$ such that

$$(\lambda_l, \lambda_l) \in \nu(\Delta_3) \tag{6.8}$$
$$(s[l \mapsto \mathit{in}_Z 0],\ s'[l \mapsto \mathit{in}_Z 0]) \in \mathit{states}(\Delta_3). \tag{6.9}$$

Choose $\Delta_3 = \Delta_2[l \mapsto i^{-1}(\lambda\Delta.R)]$. Then (6.8) follows directly from the fact that $\nu = \mathit{ref}(\lambda\Delta.R)$: indeed, for all $\Delta_4 \geq \Delta_3$we have

$$i(\Delta_4(l))(\Delta_4) = R = (\lambda\Delta.R)(\Delta_4).$$

As for (6.9), it is clear that

$$((s[l \mapsto \mathit{in}_Z 0])(l),\ (s'[l \mapsto \mathit{in}_Z 0])(l)) = (\mathit{in}_Z 0, \mathit{in}_Z 0) \in i(\Delta_3(l))(\Delta_3).$$

For $l' \neq l$, the fact that $(s, s') \in \mathit{states}(\Delta_2)$ (and monotonicity) gives

$$\begin{aligned}((s[l \mapsto in_Z(0)])(l'),\, (s'[l \mapsto in_Z(0)])(l')) = (s(l'), s'(l')) &\in i(\Delta_2(l))(\Delta_2)\\ &= i(\Delta_3(l))(\Delta_2)\\ &\subseteq i(\Delta_3(l))(\Delta_3).\end{aligned}$$

We conclude that (6.9) holds, and hence that $\emptyset \mid \emptyset \models J \sim J' : \tau_{\mathrm{m}}$.

6.3. *Example: local references and closures*

One can alternatively implement an imperative counter module by means of a local reference and two closures. Consider the type $\tau_{\mathrm{lr}} = 1 \to ((1 \to 1) \times (1 \to \mathtt{int}))$ and the two counter implementations

$$\begin{aligned}M &= \lambda x : 1.\, \mathtt{let}\, r \;=\; \mathtt{ref}\, 0\, \mathtt{in}\; (\lambda y : 1.\, r := !r + 1,\; \lambda y : 1.\, !r)\\ M' &= \lambda x : 1.\, \mathtt{let}\, r \;=\; \mathtt{ref}\, 0\, \mathtt{in}\; (\lambda y : 1.\, r := !r - 1,\; \lambda y : 1.\, -(!r))\end{aligned}$$

where the $\mathtt{let} \ldots \mathtt{in}$ construct is syntactic sugar for a β-redex in the usual way. Both M and M' are closed terms of type τ_{lr}. By "store parametricity" reasoning, i.e., by exploiting the universal quantification over all larger worlds in the definition of *cont*, one can show that $\emptyset \mid \emptyset \models M \sim M' : \tau_{\mathrm{lr}}$.

In a little more detail, let $\eta(in_\to f) = [\![M]\!]$ and $\eta(in_\to f') = [\![M']\!]$. Let $\Delta \in \mathcal{W}$ be given; we must show that $(in_\to f, in_\to f') \in [\![\tau_{\mathrm{lr}}]\!](\Delta)$. To that end, let $\Delta_1 \geq \Delta$ and $(v, v') \in [\![1]\!](\Delta_1)$ be given; we must show that

$$(f(v), f'(v')) \in \mathit{comp}([\![(1 \to 1) \times (1 \to \mathtt{int})]\!])(\Delta_1). \tag{6.10}$$

We unfold the definition of *comp*: let $\Delta_2 \geq \Delta_1$ and

$$(k, k') \in \mathit{cont}([\![(1 \to 1) \times (1 \to \mathtt{int})]\!])(\Delta_2)$$

and $(s, s') \in \mathit{states}(\Delta_2)$ be given. We must show that

$$(f(v)\, k\, s,\; f'(v')\, k'\, s') \in R_{Ans}. \tag{6.11}$$

Let $l \in \omega$ be the least number not in $\mathrm{dom}(\Delta_2)$. Then

$$\begin{aligned}f(v)\, k\, s &= k\, (in_\times(in_\to g_1, in_\to g_2))\, (s[l \mapsto in_Z(0)])\\ f'(v')\, k'\, s' &= k'\, (in_\times(in_\to g_1', in_\to g_2'))\, (s'[l \mapsto in_Z(0)])\end{aligned}$$

where $\eta(in_\to g_1) = [\![\lambda y : 1.\, r := !r + 1]\!][r \mapsto \lambda_l]$ and $\eta(in_\to g_2) = [\![\lambda y : 1.\, !r]\!][r \mapsto \lambda_l]$; similarly for g_1' and g_2'. In order to show (6.11), it therefore suffices to find $\Delta_3 \geq \Delta_2$ such that

$$\begin{aligned}(in_\to g_1,\; in_\to g_1') &\in [\![1 \to 1]\!](\Delta_3)\\ (in_\to g_2,\; in_\to g_2') &\in [\![1 \to \mathtt{int}]\!](\Delta_3)\\ (s[l \mapsto in_Z(0)],\; s'[l' \mapsto in_Z(0)]) &\in \mathit{states}(\Delta_3).\end{aligned}$$

It is not hard to see that these conditions can be satisfied by choosing $\Delta_3 = \Delta_2[l \mapsto \lambda\Delta.R]$ where $R = \{(in_Z(n), in_Z(-n)) \mid n \in \omega\}$ as in the previous example.

6.4. *Example: references to the empty type*

Consider the two terms $K = \lambda x.2$ and $K' = \lambda x.3$ of type $\mathsf{ref}\,0 \to \mathsf{int}$. Given a standard operational semantics for the language, a simple bisimulation-style argument should suffice to show that K and K' are contextually equivalent: no reference cell can ever contain a value of type 0, and therefore neither function can ever be applied. However, as we will show next, the equivalence $\emptyset \mid \emptyset \models K \sim K' : \mathsf{ref}\,0 \to \mathsf{int}$ does not hold. Briefly, the reason is the existence of approximate locations in the model.

First, by the congruence property of semantic relatedness (Theorem 4.36), if $\emptyset \mid \emptyset \models K \sim K' : \mathsf{ref}\,0 \to \mathsf{int}$ holds, then so does $\emptyset \mid x : \mathsf{ref}\,0 \models K\,x \sim K'\,x : \mathsf{int}$. Therefore it suffices to show that the latter does not hold.

Let $R \in \mathit{CURel}(V)$ be the singleton relation $\{(\mathit{in}_1 *, \mathit{in}_1 *)\}$. (Any other non-empty relation in $\mathit{CURel}(V)$ would also do.) Also, let $l \in \omega$ be an arbitrary location and define $\Delta = [l \mapsto i^{-1}(\lambda\Delta'.R)]$. Then

$$(\lambda^1_l, \lambda^1_l) \in [\![\mathsf{ref}\,0]\!](\Delta) \tag{6.12}$$

since $l \in \mathrm{dom}(\Delta)$ and since $i(\Delta(l))(\Delta_1) \overset{0}{=} [\![0]\!](\Delta_1)$ holds vacuously for all Δ_1. Now let $\rho = [x \mapsto \lambda^1_l]$; then $(\rho, \rho) \in [\![x : \mathsf{ref}\,0]\!](\Delta)$. We claim that

$$([\![K\,x]\!]\rho, [\![K'\,x]\!]\rho) \notin \mathit{comp}([\![\mathsf{int}]\!])(\Delta) \tag{6.13}$$

which shows that $\emptyset \mid x : \mathsf{ref}\,0 \models K\,x \sim K'\,x : \mathsf{int}$ does not hold. Indeed, $[\![K\,x]\!]\rho = \eta(\mathit{in}_Z 2)$ while $[\![K'\,x]\!]\rho = \eta(\mathit{in}_Z 3)$, so

$$([\![K\,x]\!]\rho\,k_{\mathit{init}}\,[l \mapsto \mathit{in}_1 *],\ [\![K'\,x]\!]\rho\,k_{\mathit{init}}\,[l \mapsto \mathit{in}_1 *]) = (\lfloor \iota_1(2) \rfloor, \lfloor \iota_1(3) \rfloor) \notin \mathit{Ans}$$

which shows (6.13). We conclude that $\emptyset \mid x : \mathsf{ref}\,0 \models K\,x \sim K'\,x : \mathsf{int}$ does not hold and hence that $\emptyset \mid \emptyset \models K \sim K' : \mathsf{ref}\,0 \to \mathsf{int}$ does not hold.

Looking back, the problem is indeed the presence of approximate locations: equation (6.12) intuitively implies that $\mathsf{ref}\,0$ is "inhabited," while the operational equivalence between K and K' expresses exactly that $\mathsf{ref}\,0$ is not inhabited.

7. Related Work

Metric spaces. As already mentioned, the metric-space structure on uniform relations over universal domains is well-known (Abadi and Plotkin 1990; Amadio 1991; Amadio and Curien 1998; Cardone 1989; MacQueen et al. 1986). The inverse-limit method for solving recursive domain equations was first adapted to metric spaces by America and Rutten (1989).

Approximate locations. Approximate (or "semantic") locations were first introduced in our earlier work (Birkedal et al. 2009). That work contains an adequacy proof with respect to an operational semantics, and also an entirely different, quasi-syntactic interpretation of open types. Here we instead present an in some ways more natural interpretation that results from solving a recursive metric-space equation, thus obtaining a proper universe of semantic types. Open types are then interpreted in the expected way, i.e., as maps from environments of semantic types to semantic types.

Game semantics. Abramsky et al. (1998) present a game semantics for a language with general reference types. References are, informally speaking, modelled as pairs of "get" and "set" functions, following an idea by Reynolds. This model of references allows for a compositional interpretation of reference types, as in this paper. In other words, the meaning of the reference type $\mathsf{ref}\,\tau$ is completely determined by the meaning of the type τ. On the other hand, modelling references as pairs of functions introduces the so-called "bad variable" problem: there is no guarantee that a pair of functions inhabiting the reference type behaves like a reference cell, in the sense that, e.g., calling "get" returns the most recent value given to "set." Another consequence is that the model cannot be used to interpret equality testing of references.

Our model has a related, but milder variant of the bad variable problem. As illustrated by the example in Section 6.4, there is no guarantee that an inhabitant of a semantic reference type behaves like an actual location in terms of observations performed by lookup and assignment. It is, however, guaranteed that every inhabitant corresponds to an actual reference cell, and that lookup and assignment operations are domain-theoretic *approximations* of the "real" operations on that reference cell. The model can easily be used to interpret equality testing of references. On the other hand, the model of Abramsky et al. is fully abstract for a language that includes a "bad-variable constructor," while our model is far from fully abstract.

Laird (2007) presents a fully abstract trace semantics for a language with general references. This model, which has a strong game-semantics "flavor," does not exhibit the bad variable problem.

Game semantics have been used to model recursive types (McCusker 2000) and impredicative polymorphism (Abramsky and Jagadeesan 2005). The latter model features a very different approach to parametricity that allows for non-relational reasoning about terms of polymorphic type. We are not aware of any game semantics that models the combination of impredicative polymorphism, general reference types, and recursive types considered in this article.

Step-indexing and recursively defined worlds. The technique of solving a metric-space equation in order to build a Kripke-style model, as presented in this paper, has subsequently been used by Schwinghammer et al. in their model of separation logic for a language with higher-order store (Schwinghammer et al. 2009), and in a more recent extension (Schwinghammer et al. 2010) (with F. Pottier) that includes an "anti-frame rule" for local reasoning about state.

The fundamental circularity between worlds and types in realizability-style possible-worlds models of polymorphism and general references was observed by Ahmed (2004, p. 62) in the setting of operational semantics (and for unary relations). Rather than solve a recursive equation, her solution is to stratify worlds and types into different levels, represented by natural numbers. So-called step-indexing is used in the definition to ensure that a stratified variant of the fundamental theorem holds. These stratified worlds and types are somewhat analogous to the approximants of recursive-equation solutions that are employed in the inverse-limit method. The main advantage in "going to the limit" of the approximations and working with an actual solution (as we do here)

is that approximation information is then not ubiquitous in definitions and proofs; by analogy, the only "approximation information" in our model is in the interpretation of references and in the requirement that user-supplied relations are uniform.[§]

In recent work, the authors and others (Birkedal et al. 2010a) have shown that the metric-space approach presented here can used together with step-indexing, providing a high-level approach to giving operational semantics that involve recursively defined structures. In particular, we have presented a model of Charguéraud and Pottier's type-and-capability system for an ML-like higher-order language, and also an operational model of the logic of Schwinghammer et al. (2009).

Ahmed et al. (2009) have recently (and independently) proposed a step-indexed model of a language very similar to ours, but in which worlds are defined in a more complicated way: this allows for proofs of much more advanced equivalences involving local state. As described in the introduction, we have recently shown that our approach extends to this style of worlds (Birkedal et al. 2010b). In future work, we furthermore plan to investigate local-state parameters in the style of Bohr and Birkedal (2006). In the present paper, we instead hope to have presented the fundamental ideas behind Kripke logical relations over recursively defined sets of worlds as needed for semantic modeling of parametric polymorphism, recursive types, and general references.

Acknowledgements. We are grateful to the anonymous referees for comments and suggestions. We thank Nick Benton, Lennart Beringer, Stephen Brookes, Martin Hofmann, Andrew Kennedy, Paul Blain Levy, Rasmus Møgelberg, Bernhard Reus, John Reynolds, Jan Schwinghammer, and Hongseok Yang for comments and discussions regarding this work.

Appendix A. Concrete descriptions of some metric spaces

Recall that the set $\mathcal{T} = \mathcal{W} \to_{mon} \mathit{CURel}(V)$ is a subset of the underlying set of the exponential $\mathcal{T} = \mathcal{W} \to \mathit{CURel}(V)$ in CBUlt, and as such is given a natural metric by Proposition 4.18. Call that metric d_1 below. In addition, let d_2 be the metric on $\widehat{\mathcal{T}}$ associated with the isomorphism $i : \widehat{\mathcal{T}} \to 1/2 \cdot \mathcal{T}$ obtained from Theorem 4.20. The fact that i is an isomorphism then implies that

$$d_2(\widehat{\nu}, \widehat{\nu}') = 1/2 \cdot d_1(i(\widehat{\nu}),\, i(\widehat{\nu}')) \tag{A.1}$$

for all $\widehat{\nu}$ and $\widehat{\nu}'$ in $\widehat{\mathcal{T}}$.

Lemma A.1. For $\Delta, \Delta' \in \mathcal{W}$, we have that $\Delta \overset{n}{=} \Delta'$ if and only if either (1) $n = 0$, or (2) $n > 0$ and $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and

$$i\,(\Delta(l))\Delta_0 \overset{n-1}{=} i\,(\Delta'(l))\Delta_0$$

for all $l \in \mathrm{dom}(\Delta)$ and all $\Delta_0 \in \mathcal{W}$.

[§] In future work we plan to perform a more formal comparison.

(The "$n-1$" comes from the "1/2" on the right-hand side of the isomorphism (4.4).)

Proof. "Only if": Assume that $\Delta \stackrel{n}{=} \Delta'$ and that $n > 0$; we must show that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and that $i\,(\Delta(l))\Delta_0 \stackrel{n-1}{=} i\,(\Delta'(l))\Delta_0$ for all $l \in \mathrm{dom}(\Delta)$ and all $\Delta_0 \in \mathcal{W}$. Since $d(\Delta, \Delta') \leq 2^{-n} < 1$, the definition of the metric on $\mathcal{W}$ implies that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and that $d(\Delta, \Delta') = \max\{d_2(\Delta(l), \Delta'(l)) \mid l \in \mathrm{dom}(\Delta)\}$. Now let $l \in \mathrm{dom}(\Delta)$ and $\Delta_0 \in \mathcal{W}$. First, $d_2(\Delta(l), \Delta'(l)) \leq d(\Delta, \Delta') \leq 2^{-n}$, and (A.1) therefore gives that

$$d_1(i(\Delta(l)), i(\Delta'(l))) = 2 \cdot d_2(\Delta(l), \Delta'(l)) \leq 2^{-(n-1)}\,.$$

Then by definition of the metric d_1 on $\mathcal{T}$ (Proposition 4.18),

$$d(i(\Delta(l))\Delta_0,\, i(\Delta'(l))(\Delta_0)) \leq d_1(i(\Delta(l)), i(\Delta'(l))) \leq 2^{-(n-1)}\,,$$

i.e., $i(\Delta(l))\Delta_0 \stackrel{n-1}{=} i(\Delta'(l))(\Delta_0)$, and we are done.

"If": The relation $\Delta \stackrel{0}{=} \Delta'$ holds for any Δ and Δ' since $\mathcal{W}$ is 1-bounded. So assume that $n > 0$, that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$, and that $i\,(\Delta(l))\Delta_0 \stackrel{n-1}{=} i\,(\Delta'(l))\Delta_0$ for all $l \in \mathrm{dom}(\Delta)$ and all $\Delta_0 \in \mathcal{W}$. Then for every $l \in \mathrm{dom}(\Delta)$ we have $d_1(i\,(\Delta(l)), i(\Delta'(l))) \leq 2^{-(n-1)}$ by definition of d_1, and hence

$$d_2(\Delta(l), \Delta'(l)) = 1/2 \cdot d_1(i(\Delta(l)), i(\Delta'(l))) \leq 2^{-n}$$

by (A.1). Therefore, $d(\Delta, \Delta') \leq 2^{-n}$, i.e., $\Delta \stackrel{n}{=} \Delta'$. □

Lemma A.2. Let $(A, (\varpi_n)_{n\in\omega})$ be a uniform cpo. Abbreviate

$$\mathit{CURel}(A) = \mathit{CURel}(A, (\varpi_n)_{n\in\omega})$$

and consider the metrics on $\mathit{CURel}(A)$ and $\mathcal{W} \to_{mon} \mathit{CURel}(A)$ given by Proposition 4.18.

1. For $R, S \in \mathit{CURel}(A)$, we have that $R \stackrel{n}{=} S$ if and only if $\varpi_n \in R \to S_\perp$ and $\varpi_n \in S \to R_\perp$.
2. For $\nu, \nu' \in \mathcal{W} \to_{mon} \mathit{CURel}(A)$, we have that $\nu \stackrel{n}{=} \nu'$ if and only if $\nu(\Delta_0) \stackrel{n}{=} \nu'(\Delta_0)$ for all $\Delta_0 \in \mathcal{W}$.
3. A function ν from $\mathcal{W}$ to $\mathit{CRel}(A)$ belongs to $\mathcal{W} \to_{mon} \mathit{CURel}(A)$ if and only if it satisfies the following two conditions for all $\Delta, \Delta' \in \mathcal{W}$:
   (a) If $\Delta \leq \Delta'$, then $\nu(\Delta) \subseteq \nu(\Delta')$.
   (b) If $\Delta \stackrel{n}{=} \Delta'$, then $\varpi_n \in \nu(\Delta) \to \nu(\Delta')_\perp$.

Lemma A.3. Let $(D, (\varpi_n)_{n\in\omega})$ be a uniform cppo. Abbreviate

$$\mathit{AURel}(D) = \mathit{AURel}(D, (\varpi_n)_{n\in\omega})$$

and consider the metrics on $\mathit{AURel}(D)$ and $\mathcal{W} \to_{mon} \mathit{AURel}(D)$ given by Propositions 4.14 and 4.15.

1. For $R, S \in \mathit{AURel}(D)$, we have that $R \stackrel{n}{=} S$ if and only if $\varpi_n \in R \to S$ and $\varpi_n \in S \to R$.
2. For $\xi, \xi' \in \mathcal{W} \to_{mon} \mathit{AURel}(D)$, we have that $\xi \stackrel{n}{=} \xi'$ if and only if $\xi(\Delta_0) \stackrel{n}{=} \xi'(\Delta_0)$ for all $\Delta_0 \in \mathcal{W}$.

3. A function ξ from $\mathcal{W}$ to $ARel(D)$ belongs to $\mathcal{W} \to_{mon} AURel(D)$ if and only if it satisfies the following two conditions for all $\Delta, \Delta' \in \mathcal{W}$:
   (a) If $\Delta \leq \Delta'$, then $\xi(\Delta) \subseteq \xi(\Delta')$.
   (b) If $\Delta \stackrel{n}{=} \Delta'$, then $\varpi_n \in \xi(\Delta) \to \xi(\Delta')$.

Appendix. References

Abadi, M. (2000). Top-top-closed relations and admissibility. *Mathematical Structures in Computer Science*, 10(3):313–320.

Abadi, M. and Plotkin, G. D. (1990). A per model of polymorphism and recursive types. In *Proceedings of LICS*, pages 355–365.

Abramsky, S., Honda, K., and McCusker, G. (1998). A fully abstract game semantics for general references. In *Proceedings of LICS*, pages 334–344.

Abramsky, S. and Jagadeesan, R. (2005). A game semantics for generic polymorphism. *Annals of Pure and Applied Logic*, 133(1-3):3–37.

Ahmed, A. (2004). *Semantics of Types for Mutable State*. PhD thesis, Princeton University.

Ahmed, A., Dreyer, D., and Rossberg, A. (2009). State-dependent representation independence. In *Proceedings of POPL*, pages 340–353.

Amadio, R. M. (1991). Recursion over realizability structures. *Information and Computation*, 91(1):55–85.

Amadio, R. M. and Curien, P.-L. (1998). *Domains and Lambda-Calculi*. Cambridge University Press.

America, P. and Rutten, J. J. M. M. (1989). Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.*, 39(3):343–375.

Baier, C. and Majster-Cederbaum, M. E. (1997). The connection between initial and unique solutions of domain equations in the partial order and metric approach. *Formal Aspects of Computing*, 9(4):425–445.

Benton, N. and Hur, C.-K. (2009). Biorthogonality, step-indexing and compiler correctness. In *Proceedings of ICFP*, pages 97–108.

Benton, N. and Leperchey, B. (2005). Relational reasoning in a nominal semantics for storage. In *Proceedings of TLCA*, number 3461 in LNCS, pages 86–101.

Birkedal, L., Reus, B., Schwinghammer, J., Støvring, K., Thamsborg, J., and Yang, H. (2010a). Step-indexed kripke models over recursive worlds. Submitted for publication. Available at `http://www.itu.dk/people/birkedal/papers/step-recworld-conf.pdf`.

Birkedal, L., Støvring, K., and Thamsborg, J. (2009). Relational parametricity for references and recursive types. In *Proceedings of TLDI*, pages 91–104.

Birkedal, L., Støvring, K., and Thamsborg, J. (2010b). A relational realizability model for higher-order stateful ADTs. Submitted for publication. Available at `http://www.itu.dk/people/birkedal/papers/relrmhoadt.pdf`.

Bohr, N. and Birkedal, L. (2006). Relational reasoning for recursive types and references. In *Proceedings of APLAS*, number 4279 in LNCS, pages 79–96.

Cardone, F. (1989). Relational semantics for recursive types and bounded quantification. In *Proceedings of ICALP*, number 372 in LNCS, pages 164–178.

Crary, K. and Harper, R. (2007). Syntactic logical relations for polymorphic and recursive types. *Electronic Notes in Theoretical Computer Science*, 172:259–299.

Laird, J. (2007). A fully abstract trace semantics for general references. In *Proceedings of ICALP*, volume 4596 of *LNCS*, pages 667–679.

Levy, P. B. (2006). Call-by-push-value: Decomposing call-by-value and call-by-name. *Higher-Order and Symbolic Computation*, 19(4):377–414.

MacQueen, D. B., Plotkin, G. D., and Sethi, R. (1986). An ideal model for recursive polymorphic types. *Information and Control*, 71(1/2):95–130.

McCusker, G. (2000). Games and full abstraction for FPC. *Information and Computation*, 160(1-2):1–61.

Moggi, E. (1991). Notions of computation and monads. *Information and Computation*, 93:55–92.

Petersen, R. L., Birkedal, L., Nanevski, A., and Morrisett, G. (2008). A realizability model for impredicative Hoare type theory. In *Proceedings of ESOP*, number 4960 in LNCS, pages 337–352.

Pierce, B. C. (2002). *Types and Programming Languages*. The MIT Press.

Pitts, A. M. (1996). Relational properties of domains. *Information and Computation*, 127:66–90.

Pitts, A. M. (1998). Existential types: Logical relations and operational equivalence. In *Proceedings of ICALP*, volume 1443 of *LNCS*, pages 309–326.

Plotkin, G. D. and Power, J. (2004). Computational effects and operations: An overview. *Electronic Notes in Theoretical Computer Science*, 73:149–163.

Schwinghammer, J., Birkedal, L., Reus, B., and Yang, H. (2009). Nested Hoare triples and frame rules for higher-order store. In *Proceedings of CSL*, number 5771 in LNCS, pages 440–454.

Schwinghammer, J., Yang, H., Birkedal, L., Pottier, F., and Reus, B. (2010). A semantic foundation for hidden state. In *Proceedings of FOSSACS*, volume 6014 of *LNCS*, pages 2–17.

Smyth, M. B. (1992). Topology. In Abramsky, S., Gabbay, D., and Maibaum, T. S. E., editors, *Handbook of Logic in Computer Science*. Oxford University Press.

Smyth, M. B. and Plotkin, G. D. (1982). The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11(4):761–783.

Wagner, K. R. (1994). *Solving Recursive Domain Equations with Enriched Categories*. PhD thesis, Carnegie Mellon University.

Winskel, G. (1993). *The Formal Semantics of Programming Languages*. Foundation of Computing Series. The MIT Press.

Chapter 4

The Category-Theoretic Solution of Recursive Metric-Space Equations

The Category-Theoretic Solution of Recursive Metric-Space Equations

Lars Birkedal* Kristian Støvring†

Jacob Thamsborg‡

IT University of Copenhagen§

May 9, 2010

Abstract

It is well known that one can use an adaptation of the inverse-limit construction to solve recursive equations in the category of complete ultrametric spaces. We show that this construction generalizes to a large class of categories with metric-space structure on each set of morphisms: the exact nature of the objects is less important. In particular, the construction immediately applies to categories where the objects are ultrametric spaces with 'extra structure', and where the morphisms preserve this extra structure. The generalization is inspired by classical domain-theoretic work by Smyth and Plotkin.

For many of the categories we consider, there is a natural subcategory in which each set of morphisms is required to be a compact metric space. Our setting allows for a proof that such a subcategory always inherits solutions of recursive equations from the full category.

As another application, we present a construction that relates solutions of generalized domain equations in the sense of Smyth and Plotkin to solutions of equations in our class of categories.

Our primary motivation for solving generalized recursive metric-space equations comes from recent and ongoing work on Kripke-style models in which the sets of worlds must be recursively defined. We show a series of examples motivated by this line of work.

Keywords: Metric space, fixed point, recursive equation.

1 Introduction

Smyth and Plotkin [22] showed that in the classical inverse-limit construction of solutions to recursive domain equations, what matters is not that the *objects* of the category under consideration are domains, but that the sets of *morphisms* between objects are domains. In this article we show that, in the case of ultrametric spaces, the standard construction of solutions to recursive metric-space equations [6, 11] can be similarly generalized to a large class of categories with metric-space structure on each set of morphisms. The generalization in particular allows one to solve recursive

*Email: birkedal@itu.dk.

†Corresponding author. Email: kss@itu.dk. Phone: (+45)72185047. Fax: (+45)72185001.

‡Email: thamsborg@itu.dk

§Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark.

equations in categories where the objects are ultrametric spaces with some form of additional structure, and where the morphisms preserve this additional structure. Some applications from recent and ongoing work in semantics are shown in Section 7.

For many of the categories we consider, there is a natural variant, indeed a subcategory, in which each set of morphisms is required to be a compact metric space [3, 10]. Our setting allows for a general proof that such a subcategory inherits solutions of recursive equations from the full category. Otherwise put, the problem of solving recursive equations in such a 'locally compact' subcategory is, in a certain sense, reduced to the similar problem for the full category. The fact that one can solve recursive equations in a category of compact ultrametric spaces [10] arises as a particular instance. (For various applications of compact metric spaces in semantics, see the references in the introduction to van Breugel and Warmerdam [10].)

As another application, we present a construction that relates solutions of generalized domain equations in the sense of Smyth and Plotkin to solutions of equations in our class of categories. This construction generalizes and improves an earlier one due to Baier and Majster-Cederbaum [7].

The key to achieving the right level of generality in the results lies in inspiration from enriched category theory. We shall not refer to general enriched category theory below, but rather present the necessary definitions in terms of metric spaces. The basic idea is, however, that given a cartesian category V (or more generally, a monoidal category), one considers so-called V-categories, in which the 'hom-sets' are in fact objects of V instead of sets, and where the 'composition functions' are morphisms in V.

Other related work. The idea of considering categories with metric spaces as hom-sets has been used in earlier work [10, 19]. Rutten and Turi [19] show existence and uniqueness of fixed points in a particular category of (not necessarily ultrametric) spaces, but with a proof where parts are more general: some aspects of our Lemma 3.2 are covered. In other work, van Breugel and Warmerdam [10] show uniqueness for a more general notion of categories than ours, again not requiring ultrametricity. Neither of these articles contain a theorem about existence of fixed points for a general class of 'metric-enriched' categories (as in our Theorem 3.1), nor a general theorem about fixed points in locally compact subcategories (Theorem 4.1).

Alessi et al. [4] consider solutions to *non-functorial* recursive equations in certain categories of metric spaces, i.e., recursive equations whose solutions cannot necessarily be described as fixed-points of functors. In contrast, we only consider *functorial* recursive equations in this article.

Wagner [25] gives a comprehensive account of a generalized inverse limit construction that in particular works for categories of metric spaces and categories of domains. Our generalization is in a different direction, namely to categories where the hom-sets are metric spaces. We do not know whether there is a common generalization of our work and Wagner's work. In this article we do not aim for maximal generality, but rather for a level of generality that seems right for applications in the style of those in Section 7.

A more detailed discussion of the level of generality of our results, and of their relation to results in the literature, can be found in Section 9.

2 Ultrametric spaces

We first recall some basic definitions and properties about metric spaces [21].

A metric space (X, d) is 1-*bounded* if $d(x, y) \leq 1$ for all x and y in X. We shall only work with 1-bounded metric spaces. One advantage of doing so is that one

can define coproducts and general products of such spaces; alternatively, one could have allowed infinite distances.

An *ultrametric space* is a metric space (X, d) that satisfies the 'ultrametric inequality,'

$$d(x, z) \leq \max(d(x, y), d(y, z)),$$

and not just the weaker triangle inequality (where one has + instead of max on the right-hand side). It might be helpful to think of the function d of an ultrametric space (X, d) not as a measure of (euclidean) distance between elements, but rather as a measure of the degree of similarity between elements.

A function $f : X_1 \to X_2$ from a metric space (X_1, d_1) to a metric space (X_2, d_2) is *non-expansive* if $d_2(f(x), f(y)) \leq d_1(x, y)$ for all x and y in X_1. Stronger, such a function f is *contractive* if there exists $c < 1$ such that $d_2(f(x), f(y)) \leq c \cdot d_1(x, y)$ for all x and y in X_1. Notice that a non-expansive function is (uniformly) continuous in the metric-space sense and hence preserves limits of convergent sequences.

A metric space is *complete* if it is Cauchy-complete in the usual sense, i.e., if every Cauchy sequence in the metric space has a limit. By Banach's fixed-point theorem, every contractive function from a non-empty, complete metric space to itself has a unique fixed point.

In the following we only consider complete, 1-bounded ultrametric spaces. As a canonical example of such a metric space, consider the set $\mathbb{N}^\omega$ of infinite sequences of natural numbers, with distance function d given by:

$$d(x, y) = \begin{cases} 2^{-\max\{n \in \omega \mid \forall m \leq n.\, x(m) = y(m)\}} & \text{if } x \neq y \\ 0 & \text{if } x = y. \end{cases}$$

To avoid confusion, call the elements of $\mathbb{N}^\omega$ *strings* instead of sequences. Here the ultrametric inequality simply states that if x and y agree on the first n 'characters' and y and z also agree on the first n characters, then x and z agree on the first n characters. A Cauchy sequence in $\mathbb{N}^\omega$ is a sequence of strings $(x_n)_{n \in \omega}$ in which the individual characters 'stabilize': for every m there exists $N \in \omega$ such that $x_{n_1}(m) = x_{n_2}(m)$ for all $n_1, n_2 \geq N$.

Let CBUlt be the category with complete, 1-bounded ultrametric spaces as objects and non-expansive functions as morphisms [6]. This category is cartesian closed [21]; here one needs the ultrametric inequality. The terminal object is the one-point metric space. Binary products are defined in the natural way: $(X_1, d_1) \times (X_2, d_2) = (X_1 \times X_2,\, d_{X_1 \times X_2})$ where

$$d_{X_1 \times X_2}((x_1, x_2), (y_1, y_2)) = \max(d_1(x_1, y_1), d_2(x_2, y_2))\,.$$

The exponential $(X_1, d_1) \to (X_2, d_2)$, sometimes written $(X_2, d_2)^{(X_1, d_1)}$, has the set of non-expansive functions from (X_1, d_1) to (X_2, d_2) as the underlying set, and the 'sup'-metric $d_{X_1 \to X_2}$ as distance function: $d_{X_1 \to X_2}(f, g) = \sup\{d_2(f(x), g(x)) \mid x \in X_1\}$. For both products and exponentials, limits are pointwise. It follows from the cartesian closed structure that the function $(X_3, d_3)^{(X_2, d_2)} \times (X_2, d_2)^{(X_1, d_1)} \to (X_3, d_3)^{(X_1, d_1)}$ given by composition is non-expansive; this fact is needed in several places below.

Moreover, the category CBUlt is complete [18]: general products are defined in the same way as binary ones, except that the distance function on an infinite product space is in general given by a supremum instead of a maximum. An equalizer of two parallel arrows $f, g : X \longrightarrow Y$ is given by the subset $\{\, x \in X \mid f(x) = g(x) \,\}$ of X, with the metric inherited from X.

CBUlt is also cocomplete. The coproduct of a family $(X_j, d_j)_{j \in J}$ of CBUlt-objects is $(\coprod_{j \in J} X_j,\, d)$ where $\coprod_{j \in J} X_j$ is the disjoint union (coproduct in Set) of

the underlying sets X_j, and where the distance function d is given by

$$d(x,y) = \begin{cases} d_j(x,y), & \text{if } x \in X_j \text{ and } y \in X_j \text{ for some } j \in J, \\ 1, & \text{otherwise.} \end{cases}$$

Coequalizers are more complicated to describe, and we shall not need them in this article.

It is a trivial fact, but for our purposes a rather annoying one, that Banach's fixed-point theorem only holds for non-empty metric spaces. To avoid tedious special cases below, we shall therefore not work with the category CBUlt, but rather with the full subcategory $\mathsf{CBUlt}_{\mathrm{ne}}$ of *non-empty*, complete, 1-bounded ultrametric spaces. This category is also cartesian closed: since it is a full subcategory of CBUlt, it suffices to verify that CBUlt-products of non-empty metric spaces are non-empty, and similarly for exponentials. The category $\mathsf{CBUlt}_{\mathrm{ne}}$ is not complete, and in fact it does not even have all limits of ω^{op}-chains. We return to that point in Section 5.

In some settings it is useful to work with *compact* metric spaces [3, 10]. Recall that a metric space is compact in the usual topological sense if and only if is both complete and *totally bounded* [21]: for every $\epsilon > 0$, there exist finitely many points $x_1, \ldots, x_n$ in the space such that the open balls with centers x_i and radius ϵ cover the space. As a canonical example of a compact, 1-bounded ultrametric space, consider the set $\{0,1\}^\omega$ of infinite sequences of zeros or ones, with distance function given as in the example with sequences of natural numbers above. (Any finite set other than $\{0,1\}$ would also work.)

Let KBUlt be the full subcategory of CBUlt consisting of compact, 1-bounded ultrametric spaces, and let $\mathsf{KBUlt}_{\mathrm{ne}}$ be the full subcategory of *non-empty* such spaces. Both of these categories are cartesian closed [21] and have finite coproducts. KBUlt has all finite limits, but neither KBUlt nor $\mathsf{KBUlt}_{\mathrm{ne}}$ is complete. We return to that point in Section 4.

2.1 M-categories

Recall from the introduction that the basic idea of this article is to generalize a theorem about a particular category of metric spaces, here $\mathsf{CBUlt}_{\mathrm{ne}}$, to a theorem about all '$\mathsf{CBUlt}_{\mathrm{ne}}$-categories' where the hom-sets are in fact appropriate metric spaces. In analogy with the O-categories of Smyth and Plotkin (O for 'order' or 'ordered') we call such categories M-categories.

Definition 2.1. An *M-category* is a category $\mathcal{C}$ where each hom-set $\mathcal{C}(A,B)$ is equipped with a distance function turning it into a non-empty, complete, 1-bounded ultrametric space, and where each composition function $\circ : \mathcal{C}(B,C) \times \mathcal{C}(A,B) \to \mathcal{C}(A,C)$ is non-expansive with respect to these metrics. (Here the domain of such a composition function is given the product metric.)

In other words, an M-category is a category where each hom-set is equipped with a metric which turns it into an object in $\mathsf{CBUlt}_{\mathrm{ne}}$; furthermore, each composition function must be a morphism in $\mathsf{CBUlt}_{\mathrm{ne}}$.

A simple example of an M-category is $\mathsf{CBUlt}_{\mathrm{ne}}$ itself. The distance function on each hom-set $\mathsf{CBUlt}_{\mathrm{ne}}(A,B)$ is defined as for the exponential B^A in $\mathsf{CBUlt}_{\mathrm{ne}}$, i.e., $d(f,g) = \sup\{d_B(f(x),g(x)) \mid x \in A\}$. The fact that the composition functions are non-expansive, as observed in Section 2, depends on the ultrametric inequality. Since $\mathsf{CBUlt}_{\mathrm{ne}}$ is itself an M-category the results below can be used to solve standard recursive equations over ultrametric spaces.

Let $\mathcal{C}$ be an M-category. A functor $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ is *locally non-expansive* if $d(F(f,g),F(f',g')) \leq \max(d(f,f'),d(g,g'))$ for all f,f', g, and g' with appropriate

domains and codomains. In other words, such an F is locally non-expansive if each component

$$F_{A,A',B,B'} : \mathcal{C}(A', A) \times \mathcal{C}(B, B') \to \mathcal{C}(F(A, B), F(A', B'))$$

is a morphism in $\mathsf{CBUlt}_{\mathrm{ne}}$. Stronger, F is *locally contractive* if there exists some $c < 1$ such that $d(F(f, g), F(f', g')) \leq c \cdot \max(d(f, f'), d(g, g'))$ for all f,f', g, and g'. Notice that c is global in the sense that it is a common 'contractiveness factor' for all components of the functor: each component $F_{A,A',B,B'}$ is contractive with factor c.

In the particular categories we consider in the examples in Section 7, many 'natural' functors such as those given by binary products or coproducts are only locally non-expansive, not locally contractive. On each of these categories $\mathcal{C}$ there is, however, an appropriate functor $\frac{1}{2} : \mathcal{C} \to \mathcal{C}$ which multiplies all distances in hom-sets by the factor $1/2$. Composing a locally non-expansive functor with $\frac{1}{2}$ then yields a locally contractive functor.

3 Solving recursive equations

Let $\mathcal{C}$ be an M-category. We consider mixed-variance functors $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ on $\mathcal{C}$ and recursive equations of the form

$$X \cong F(X, X)\,.$$

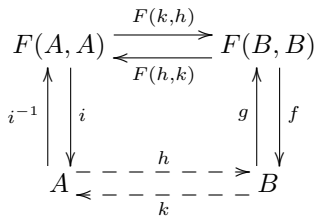
In other words, given such an F we seek a fixed point of F up to isomorphism.

Covariant endofunctors on $\mathcal{C}$ are a special case of mixed-variance functors. It would in some sense suffice to study covariant functors: if $\mathcal{C}$ is an M-category, then so are $\mathcal{C}^{\mathrm{op}}$ (with the same metric on each hom-set as in $\mathcal{C}$) and $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$ (with the product metric on each hom-set), and it is well-known how to construct a 'symmetric' endofunctor on $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$ from a functor such as F above. We explicitly study mixed-variance functors since the proof of the existence theorem below would in any case involve an M-category of the form $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$. As a benefit we directly obtain theorems of the form useful in applications. For example, for the existence theorem we are interested in completeness conditions on $\mathcal{C}$, not on $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$.

3.1 Uniqueness of solutions

The results below depend on the assumption that the given functor F is locally contractive. One easy consequence of this assumption is that, unlike in the domain-theoretic setting [22], there is at most one fixed point of F up to isomorphism.

Theorem 3.1. Let $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ be a locally contractive functor on an M-category $\mathcal{C}$, and assume that $i : F(A, A) \to A$ is an isomorphism. Then the pair (i, i^{-1}) is a *bifree algebra* for F in the following sense: for all objects B of $\mathcal{C}$ and all morphisms $f : F(B, B) \to B$ and $g : B \to F(B, B)$, there exists a unique pair of morphisms $(k : B \to A, h : A \to B)$ such that $h \circ i = f \circ F(k, h)$ and $i^{-1} \circ k = F(h, k) \circ g$:



In particular, A is the unique fixed point of F up to isomorphism.

Proof. First we observe that there is an obvious way to define a category of 'bialgebras' for F such that a bifree algebra, as defined above, is an initial object in this category. It follows that any two bifree algebras are isomorphic as bialgebras, hence that their underlying $\mathcal{C}$-objects are isomorphic. So once we have shown that every fixed point of F is a bifree algebra, it follows that there is at most one fixed point of F up to isomorphism.

Let now $i : F(A, A) \to A$ be an isomorphism and assume that F is locally contractive with factor $c < 1$; we show that (i, i^{-1}) is a bifree algebra. Let $f : F(B, B) \to B$ and $g : B \to F(B, B)$ be given. Recall that hom-sets in $\mathcal{C}$ are equipped with metrics that turn them into objects of $\mathsf{CBUlt}_{\mathrm{ne}}$, and let X be the $\mathsf{CBUlt}_{\mathrm{ne}}$-object $\mathcal{C}(B, A) \times \mathcal{C}(A, B)$. We obtain the desired pair of morphisms $(k : B \to A,\, h : A \to B)$ as the unique fixed point of the following contractive operator on X:

$$D(k, h) = (i \circ F(h, k) \circ g,\; f \circ F(k, h) \circ i^{-1})\,.$$

First we verify that this operator is indeed contractive. Given (k_1, h_1) and (k_2, h_2) in X,

$$\begin{aligned} d(D(k_1, h_1), D(k_2, h_2)) = \max(&d(i \circ F(h_1, k_1) \circ g,\; i \circ F(h_2, k_2) \circ g),\\ &d(f \circ F(k_1, h_1) \circ i^{-1},\; f \circ F(k_2, h_2) \circ i^{-1})\,, \end{aligned}$$

by the definition of the product metric. But the composition functions of an M-category are required to be non-expansive: therefore,

$$\begin{aligned} d(i \circ F(h_1, k_1) \circ g,\; i \circ F(h_2, k_2) \circ g) &\leq \max(d(i, i),\; d(F(h_1, k_1), F(h_2, k_2)),\; d(g, g))\\ &= d(F(h_1, k_1), F(h_2, k_2))\\ &\leq c \cdot \max(d(h_1, h_2),\; d(k_1, k_2))\\ &= c \cdot d((k_1, h_1),\; (k_2, h_2))\,, \end{aligned}$$

and similarly,

$$\begin{aligned} d(f \circ F(k_1, h_1) \circ i^{-1},\; f \circ F(k_2, h_2) \circ i^{-1}) &\leq d(F(k_1, h_1), F(k_2, h_2))\\ &\leq c \cdot \max(d(h_1, h_2),\; d(k_1, k_2))\\ &= c \cdot d((k_1, h_1),\; (k_2, h_2)). \end{aligned}$$

Therefore, $d(D(k_1, h_1), D(k_2, h_2)) \leq c \cdot d((k_1, h_1),\; (k_2, h_2))$, and D is locally contractive with factor c.

Since hom-sets of $\mathcal{C}$ are *non-empty* complete metric spaces, the operator D has a unique fixed point by Banach's theorem. It only remains to show that a pair of morphisms $(k : B \to A,\, h : A \to B)$ is a fixed point of D if and only if it makes the diagram in the statement of the theorem commute. But this is easy since i is an isomorphism: $k = i \circ F(h, k) \circ g$ holds if and only if $i^{-1} \circ k = F(h, k) \circ g$ holds, and similarly, $h = f \circ F(k, h) \circ i^{-1}$ holds if and only if $h \circ i = f \circ F(k, h)$ holds. We conclude that (i, i^{-1}) is a bifree algebra for F. □

In particular, if F is covariant and $i : FA \to A$ is an isomorphism, then i is an initial F-algebra and i^{-1} is a final F-coalgebra. As an example, consider the M-category $\mathsf{CBUlt}_{\mathrm{ne}}$ and take F to be the covariant functor $\frac{1}{2} : \mathsf{CBUlt}_{\mathrm{ne}} \to \mathsf{CBUlt}_{\mathrm{ne}}$ which given a metric space yields the same metric space but with all distances multiplied by $1/2$, and which is the identity on morphisms. Evidently, the one-point metric space is a fixed-point of F. By the theorem above it is also an initial algebra of F: this fact is essentially Banach's fixed-point theorem for functions that are contractive with coefficient $1/2$.

3.2 Existence of solutions

In the existence theorem for fixed points of contractive functors, the M-category $\mathcal{C}$ will be assumed to satisfy a certain completeness condition involving limits of ω^{op}-chains. Since there are different M-categories satisfying more or less general variants of this condition, it is convenient to present the existence theorem in a form that lists a number of successively weaker conditions.

One sufficient condition is that $\mathcal{C}$ has all limits of ω^{op}-chains, i.e., all limits of diagrams of the form

$$A_0 \xleftarrow{g_0} A_1 \xleftarrow{g_1} \cdots \xleftarrow{g_{n-1}} A_n \xleftarrow{g_n} \cdots \ .$$

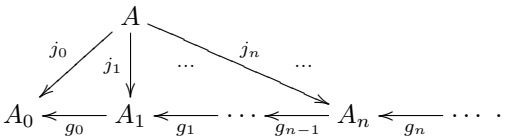
A weaker condition is that $\mathcal{C}$ has all limits of ω^{op}-chains of split epis, i.e., all limits of diagrams as above, but where each g_n has a right inverse. This perhaps rather odd-looking condition is the one that best matches the category $\mathsf{CBUlt}_{\mathrm{ne}}$ itself.

A still weaker condition is the following. An *increasing Cauchy tower* is a diagram

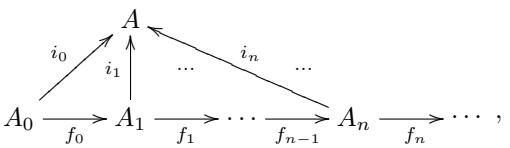
$$A_0 \underset{g_0}{\overset{f_0}{\rightleftarrows}} A_1 \underset{g_1}{\overset{f_1}{\rightleftarrows}} \cdots \underset{g_{n-1}}{\overset{f_{n-1}}{\rightleftarrows}} A_n \underset{g_n}{\overset{f_n}{\rightleftarrows}} \cdots$$

where $g_n \circ f_n = id_{A_n}$ for all n (so each g_n is split epi, as above), and where $\lim_{n\to\infty} d(f_n \circ g_n,\ id_{A_{n+1}}) = 0$. Notice that this definition only makes sense for M-categories. The M-category $\mathcal{C}$ has *inverse limits of increasing Cauchy towers* if for every such diagram, the sub-diagram containing only the arrows g_n has a limit. We return to a more detailed treatment of general Cauchy towers and their limits in Section 6.

Lemma 3.2. Let $(A_n, f_n, g_n)_{n\in\omega}$ be an increasing Cauchy tower as above, and let $(A, j_n)_{n\in\omega}$ be a cone from A to the ω^{op}-chain $(A_n, g_n)_{n\in\omega}$:

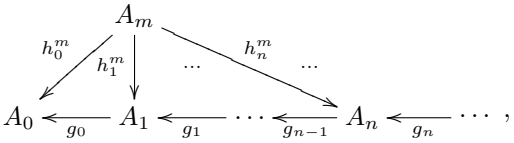


The following two conditions are equivalent: (1) The cone $(A_n, j_n)_{n\in\omega}$ is limiting. (2) There exist morphisms $i_n : A_n \to A$ such that $(A, i_n)_{n\in\omega}$ is a cocone from the ω-chain $(A_n, f_n)_{n\in\omega}$ to A,



and such that $j_n \circ i_n = id_{A_n}$ for all n and $\lim_{n\to\infty} d(i_n \circ j_n,\ id_A) = 0$.

Proof. (1) implies (2): Assume that the cone above is limiting. For each m we must define a morphism $i_m : A_m \to A$ into the object A of the limiting cone. We do so by defining a cone from A_m to $(A_n, g_n)_{n\in\omega}$,



where

$$h_n^m = \begin{cases} id_{A_m}, & \text{if } n = m, \\ g_n \circ g_{n+1} \circ \cdots \circ g_{m-1}, & \text{if } n < m, \\ f_{n-1} \circ f_{n-2} \circ \cdots \circ f_m, & \text{if } n > m. \end{cases}$$

It is easy to see that these morphisms indeed constitute a cone: in the case $n > m$ one uses that $g_{n-1} \circ f_{n-1} = id$. Hence there exists a unique morphism $i_m : A_m \to A$ such that $j_n \circ i_m = h_n^m$ for all n. In particular, $j_m \circ i_m = h_m^m = id_{A_m}$, as required in the statement of the lemma.

We must also show that $i_{m+1} \circ f_m = i_m$. By the defining property of i_m, it suffices to show that $i_{m+1} \circ f_m$ is also a cone morphism in the sense that $j_n \circ (i_{m+1} \circ f_m) = h_n^m$ for all n. And indeed, $j_n \circ i_{m+1} \circ f_m = h_n^{m+1} \circ f_m = h_n^m$ by the defining property of i_{m+1} and the definition of h_n^{m+1}.

It remains to show that $\lim_{n\to\infty} d(i_n \circ j_n,\, id_A) = 0$, or equivalently, that $\lim_{n\to\infty} i_n \circ j_n = id_A$ in the metric space $\mathcal{C}(A, A)$. To do so, we first show that $(i_n \circ j_n)_{n\in\omega}$ is a Cauchy sequence. Given $\epsilon > 0$, choose N large enough that $d(f_n \circ g_n,\, id_{A_{n+1}}) \leq \epsilon$ for all $n \geq N$. Then for all $n \geq N$,

$$\begin{aligned} d(i_n \circ j_n,\, i_{n+1} \circ j_{n+1}) &= d((i_{n+1} \circ f_n) \circ (g_n \circ j_{n+1}),\, i_{n+1} \circ j_{n+1}) \\ &= d(i_{n+1} \circ (f_n \circ g_n) \circ j_{n+1},\, i_{n+1} \circ id_{A_{n+1}} \circ j_{n+1}) \\ &\leq \max(d(i_{n+1}, i_{n+1}),\, d(f_n \circ g_n,\, id_{A_{n+1}}),\, d(j_{n+1}, j_{n+1})) \\ &= d(f_n \circ g_n,\, id_{A_{n+1}}) \\ &\leq \epsilon\,, \end{aligned}$$

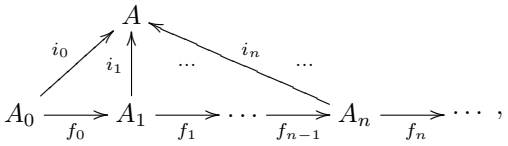
where we have used that the composition functions of an M-category are required to be non-expansive. From the ultrametric inequality one now easily obtains that $d(i_n \circ j_n,\, i_m \circ j_m) \leq \epsilon$ for all $n, m \geq N$. Hence $(i_n \circ j_n)_{n\in\omega}$ is a Cauchy sequence.

Since $C(A, A)$ is a complete metric space, the Cauchy sequence $(i_n \circ j_n)_{n\in\omega}$ has a limit $\lim_{n\to\infty} i_n \circ j_n$. It remains to show that this limit is in fact the identity morphism on A. To do so, we show that $\lim_{n\to\infty} i_n \circ j_n$ is a cone morphism from the limiting cone $(A, j_m)_{m\in\omega}$ to itself: for all m,

$$\begin{aligned} j_m \circ \Big(\lim_{n\to\infty} i_n \circ j_n\Big) &= j_m \circ \Big(\lim_{n\geq m} i_n \circ j_n\Big) \\ &= \lim_{n\geq m} (j_m \circ i_n \circ j_n) && (\text{`}\circ\text{' non-expansive}) \\ &= \lim_{n\geq m} (h_m^n \circ j_n) && (\text{defining property of } i_n) \\ &= \lim_{n\geq m} j_m && (\text{by definition of } h_m^n) \\ &= j_m\,. \end{aligned}$$

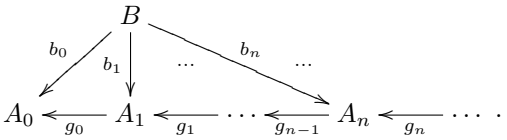
In the second line we have again used that the composition functions of an M-category are non-expansive, hence continuous, and the fact that continuous functions preserve (metric-space) limits. We conclude that $\lim_{n\to\infty} i_n \circ j_n$ is a cone morphism from a limiting cone to itself, and therefore that $\lim_{n\to\infty} i_n \circ j_n = id_A$.

(2) implies (1): Now assume that we have a commuting diagram



such that $j_n \circ i_n = id_{A_n}$ for all n and $\lim_{n\to\infty} d(i_n \circ j_n,\, id_A) = 0$. We must show that $(A, j_n)_{n\in\omega}$ is a limiting cone for the given ω^{op}-chain. So let $(B, b_n)_{n\in\omega}$ be

another cone:



We aim to define a mediating morphism $q : B \to A$ as the limit of the sequence $(i_n \circ b_n)_{n\in\omega}$. We first show that this is a Cauchy sequence. The argument is completely similar to the one for the sequence $(i_n \circ j_n)_{n\in\omega}$ above: given $\epsilon > 0$, choose N large enough that $d(f_n \circ g_n,\, id_{A_{n+1}}) \leq \epsilon$ for all $n \geq N$. Then for all $n \geq N$,

$$\begin{aligned} d(i_n \circ b_n,\, i_{n+1} \circ b_{n+1}) &= d((i_{n+1} \circ f_n) \circ (g_n \circ b_{n+1}),\, i_{n+1} \circ b_{n+1}) \\ &\leq d(f_n \circ g_n,\, id_{A_{n+1}}) \\ &\leq \epsilon\,, \end{aligned}$$

and it follows that $(i_n \circ j_n)_{n\in\omega}$ is a Cauchy sequence.

Since the metric space $\mathcal{C}(B, A)$ is complete, the Cauchy sequence above has a limit: define $q = \lim_{n\to\infty} i_n \circ b_n$. We must show that q is the unique mediating morphism from the cone $(B, b_m)_{m\in\omega}$ to the cone $(A, j_m)_{m\in\omega}$. Again, the argument is as above: first,

$$\begin{aligned} j_m \circ q = j_m \circ \Big(\lim_{n\to\infty} i_n \circ b_n\Big) &= j_m \circ \Big(\lim_{n\geq m} i_n \circ b_n\Big) \\ &= \lim_{n\geq m} (j_m \circ i_n \circ b_n) \\ &= \lim_{n\geq m} (h^n_m \circ b_n) \\ &= \lim_{n\geq m} b_m \\ &= b_m\,, \end{aligned}$$

so q is indeed a cone morphism. Second, given another such cone morphism $r : B \to A$,

$$r = id_A \circ r = \Big(\lim_{n\to\infty} i_n \circ j_n\Big) \circ r = \lim_{n\to\infty} (i_n \circ j_n \circ r) = \lim_{n\to\infty} (i_n \circ b_n) = q\,,$$

so q is unique. We conclude that $(A, j_n)_{n\in\omega}$ is a limiting cone for the ω^{op}-chain $(A_n, g_n)_{n\in\omega}$. □

Although not strictly necessary for our purposes, it is natural to ask whether the cocone described in Condition 2 of the lemma must be colimiting. We now show that this is the case by exploiting the generality of M-categories: the fact that $\mathcal{C}^{\mathrm{op}}$ is also an M-category allows for a simple proof of a limit-colimit coincidence (cf. Smyth and Plotkin [22]).

Proposition 3.3. Let $\mathcal{C}$ be an M-category, let $(A_n, f_n, g_n)_{n\in\omega}$ be an increasing Cauchy tower in $\mathcal{C}$ (as above), and let A be an object of $\mathcal{C}$. The following three conditions are equivalent:

1. A is a limit of the ω^{op}-chain $(A_n, g_n)_{n\in\omega}$.
2. A is a colimit of the ω-chain $(A_n, f_n)_{n\in\omega}$.
3. There exist a cone $(j_n)_{n\in\omega}$ from A to $(A_n, g_n)_{n\in\omega}$ and a cocone $(i_n)_{n\in\omega}$ from $(A_n, f_n)_{n\in\omega}$ to A satisfying that $j_n \circ i_n = id_{A_n}$ for all n and in addition that $\lim_{n\to\infty} d(i_n \circ j_n,\, id_A) = 0$.

Furthermore, in any pair consisting of a cone and a cocone that together satisfy the requirements in the third condition, the cone is limiting and the cocone is colimiting.

Proof. Lemma 3.2 shows that (1) and (3) are equivalent and that any cone $(j_n)_{n\in\omega}$ as in the third condition is limiting. The lemma also shows that these facts hold for the increasing Cauchy tower $(A_n, g_n, f_n)_{n\in\omega}$ in the M-category $\mathcal{C}^{\mathrm{op}}$. But by duality, this means exactly that (2) and (3) are equivalent and that any cocone $(i_n)_{n\in\omega}$ as in the third condition is colimiting. □

We now turn to the main result.

Theorem 3.4. Assume that the M-category $\mathcal{C}$ satisfies any of the following (successively weaker) conditions:

1. $\mathcal{C}$ is complete.
2. $\mathcal{C}$ has a terminal object and limits of ω^{op}-chains.
3. $\mathcal{C}$ has a terminal object and limits of ω^{op}-chains of split epis.
4. $\mathcal{C}$ has a terminal object and inverse limits of increasing Cauchy towers.

Then every locally contractive functor $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ on $\mathcal{C}$ has a unique fixed point up to isomorphism.

Proof. Uniqueness follows from the previous theorem, so it is enough to show that there exists some A such that $F(A, A) \cong A$. Assume that $\mathcal{C}$ satisfies Condition 4 above and let 1 be a terminal object of $\mathcal{C}$. By induction on n we construct a diagram

$$A_0 \underset{g_0}{\overset{f_0}{\rightleftarrows}} A_1 \underset{g_1}{\overset{f_1}{\rightleftarrows}} \cdots \underset{g_{n-1}}{\overset{f_{n-1}}{\rightleftarrows}} A_n \underset{g_n}{\overset{f_n}{\rightleftarrows}} \cdots$$

as follows: $A_0 = 1$ and $A_{n+1} = F(A_n, A_n)$ for $n > 0$. We take g_0 to be the unique morphism from A_1 to 1 and f_0 to be an arbitrary morphism in the other direction; recall that all hom-sets in an M-category are non-empty. Finally, $f_{n+1} = F(g_n, f_n)$ and $g_{n+1} = F(f_n, g_n)$ for $n > 0$.

We now show by induction on n that this diagram is an increasing Cauchy tower. More specifically, let $c < 1$ be a contractiveness factor of F. Then, for all n:

1. $g_n \circ f_n = id_{A_n}$
2. $d(f_n \circ g_n,\ id_{A_{n+1}}) \leq c^n$.

For $n = 0$, Part 1 follows from the fact that $g_0 \circ f_0$ must be the identity morphism on the terminal object A_0. Also, all distances in the spaces we consider are at most $c^0 = 1$, so Part 2 holds trivially.

As for the inductive case,

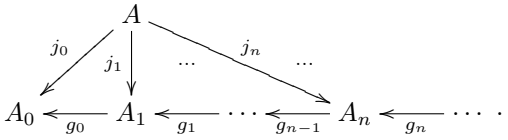
$$\begin{aligned} g_{n+1} \circ f_{n+1} &= F(f_n, g_n) \circ F(g_n, f_n) \\ &= F(g_n \circ f_n,\ g_n \circ f_n) \\ &= F(id_{A_n}, id_{A_n}) \qquad \text{(ind. hyp.)} \\ &= id_{A_{n+1}}, \end{aligned}$$

and furthermore,

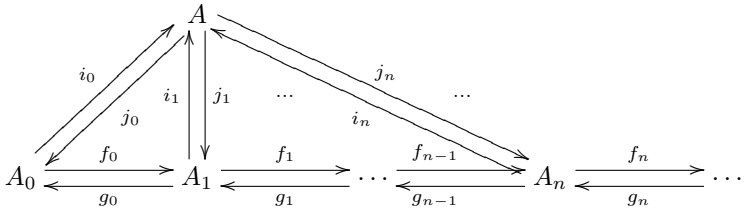
$$\begin{aligned} d(f_{n+1} \circ g_{n+1},\ id_{A_{n+2}}) &= d(F(g_n, f_n) \circ F(f_n, g_n),\ id_{A_{n+2}}) \\ &= d(F(f_n \circ g_n,\ f_n \circ g_n),\ id_{A_{n+2}}) \\ &= d(F(f_n \circ g_n,\ f_n \circ g_n),\ F(id_{A_{n+1}}, id_{A_{n+1}})) \\ &\leq c \cdot \max(d(f_n \circ g_n, id_{A_{n+1}}),\ d(f_n \circ g_n, id_{A_{n+1}},)) \\ &\leq c \cdot c^n \qquad \text{(ind. hyp.)} \\ &= c^{n+1}, \end{aligned}$$

so both parts hold. We conclude that the diagram above is indeed an increasing Cauchy tower.

By assumption on $\mathcal{C}$ there exists an inverse limit of this Cauchy tower, i.e., a limiting cone

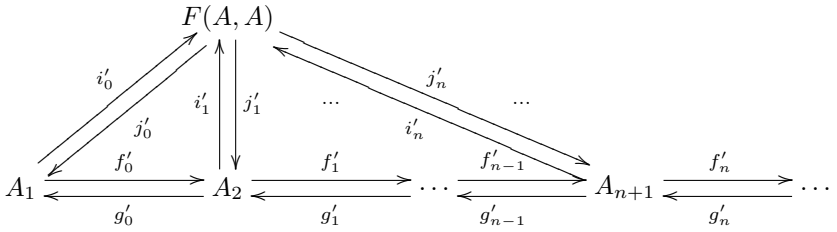


By Lemma 3.2 there exist morphisms $i_n : A_n \to A$ such that $(A, i_n)_{n\in\omega}$ is a cocone from the ω-chain $(A_n, f_n)_{n\in\omega}$ to A, and such that $j_n \circ i_n = id_{A_n}$ for all n and $\lim_{n\to\infty} d(i_n \circ j_n,\ id_A) = 0$. In particular we have a diagram



which commutes in the sense that $g_n \circ j_{n+1} = j_n$ and $i_{n+1} \circ f_n = i_n$ for all n. Removing the first object of the Cauchy tower $(A_n, f_n, g_n)_{n\in\omega}$ clearly gives a new Cauchy tower, and it is easy to see that the collection of arrows i_n and j_n with $n > 0$ satisfies Condition 2 of Lemma 3.2 with respect to that Cauchy tower. Hence by that lemma, A is also a limit of the ω^{op}-chain $(A_n, g_n)_{n>0}$ that starts from A_1.

We now show that $F(A, A)$ is also a limit of the ω^{op}-chain $(A_n, g_n)_{n>0}$. From that it follows that $F(A, A) \cong A$ and we are done. First we apply F to the diagram above, obtaining a diagram



that commutes in the same sense. Here $i'_n = F(j_n, i_n)$ and $j'_n = F(i_n, j_n)$, and similarly for the f'_n and g'_n. But by definition of the original Cauchy tower, the bottom line of the above diagram is exactly that Cauchy tower starting from A_1. Now, by functoriality we have $j'_n \circ i'_n = F(i_n, j_n) \circ F(j_n, i_n) = F(j_n \circ i_n,\ j_n \circ i_n) = F(id, id) = id$ for each n, and furthermore,

$$\begin{aligned}
\lim_{n\to\infty} d(i'_n \circ j'_n\, id_{F(A,A)}) &= \lim_{n\to\infty} d(F(j_n, i_n) \circ F(i_n, j_n), id_{F(A,A)}) \\
&= \lim_{n\to\infty} d(F(i_n \circ j_n,\ i_n \circ j_n), F(id_A, id_A)) \\
&\leq \lim_{n\to\infty} c \cdot d(i_n \circ j_n,\ id_A) \\
&= c \cdot \lim_{n\to\infty} d(i_n \circ j_n,\ id_A) \\
&= 0\,,
\end{aligned}$$

since F is contractive with factor c. Hence the morphisms in the diagram above satisfy Condition 2 of Lemma 3.2 with respect to the increasing Cauchy tower starting from A_1. By that lemma, $F(A, A)$ is therefore a limit of the ω^{op}-chain $(A_n, g_n)_{n>0}$. Since A is also such a limit we conclude that $F(A, A) \cong A$. □

By the fact that $\mathcal{C}^{\mathrm{op}}$ is also an M-category we additionally obtain a dual version of Theorem 3.4. For example, if $\mathcal{C}$ has an initial object and colimits of ω-chains of split monos ('embeddings'), then every locally contractive mixed-variance functor on $\mathcal{C}$ has a unique fixed point up to isomorphism. In the applications we have considered these dual conditions seem less useful since colimits in the categories involved are harder to describe than limits.

4 Locally compact subcategories of M-categories

The condition in Theorem 3.4 that involves Cauchy towers is included in order to accommodate categories where the hom-sets are *compact* ultrametric spaces. The simplest example is the full subcategory $\mathsf{KBUlt}_{\mathrm{ne}}$ of $\mathsf{CBUlt}_{\mathrm{ne}}$ consisting of compact, non-empty metric spaces. This category does not have all limits of ω^{op}-chains, not even of those chains where the morphisms are split epi. One can construct a counterexample as follows: for each $n \in \omega$, let A_n be the set $\{0, 1, \ldots, n-1\}$ equipped with the discrete metric. Let $f_n : A_n \to A_{n+1}$ be the inclusion and let $g_n : A_{n+1} \to A_n$ be the function that maps n to $n-1$ and every other number to itself. We claim that the ω^{op}-chain $(A_n, g_n)_{n\in\omega}$ in $\mathsf{KBUlt}_{\mathrm{ne}}$ does not have a limit. To see this, assume that $(A, j_n)_{n\in\omega}$ is a limiting cone with $j_n : A \to A_n$ for all n. By the argument in the beginning of the proof of Lemma 3.2 there exist morphisms $i_n : A_n \to A$ such that $j_n \circ i_n = id_{A_n}$ for all n. Since every A_n can in this way be embedded in A, we conclude that A contains arbitrarily large discrete subspaces. But then A cannot be totally bounded: for $\epsilon = 1/2$ there is no finite set of points such that the open balls with centers in those points and radius ϵ cover A. Hence A is not compact, a contradiction. This argument also works for KBUlt instead of $\mathsf{KBUlt}_{\mathrm{ne}}$.

The subcategory $\mathsf{KBUlt}_{\mathrm{ne}}$ is merely the simplest example of a full, 'locally compact' subcategory of an M-category. The setting of M-categories allows for a proof that such a subcategory always inherits fixed points of functors from the full category:

Theorem 4.1. Assume that $\mathcal{C}$ is an M-category with a terminal object and limits of ω^{op}-chains of split epis. Let I be an arbitrary object of $\mathcal{C}$, and let $\mathcal{D}$ be the full subcategory of $\mathcal{C}$ consisting of the objects A such that $\mathcal{C}(I, A)$ is a compact metric space. $\mathcal{D}$ is an M-category with limits of increasing Cauchy towers, and hence every locally contractive functor $F : \mathcal{D}^{\mathrm{op}} \times \mathcal{D} \to \mathcal{D}$ has a unique fixed point up to isomorphism.

Notice that the theorem refers to functors on $\mathcal{D}$, not on $\mathcal{C}$. There is in general no guarantee that a functor on $\mathcal{C}$ restricts to one on the subcategory $\mathcal{D}$, and hence formulating a recursive equation by means of a functor on $\mathcal{D}$ can require additional work [3]. In that sense, one might say that it is not exactly the problem of solving recursive equations which has been reduced to the case for the full category $\mathcal{C}$, but rather the problem of finding fixed-points of functors.

Proof. First, $\mathcal{D}$ is an M-category, being a subcategory of an M-category. Second, $\mathcal{D}$ contains each terminal object 1 of $\mathcal{C}$ since $\mathcal{C}(I, 1)$ is the one-point metric space which is clearly compact.

We next show that $\mathcal{D}$ has limits of increasing Cauchy towers; it then follows from Theorem 3.4 that $\mathcal{D}$ has fixed points of locally contractive functors. To that end, let $(A_n, f_n, g_n)_{n\in\omega}$ be an increasing Cauchy-tower in $\mathcal{D}$ (and hence also in $\mathcal{C}$). Each g_n is split epi, so by assumption the ω^{op}-chain $(A_n, g_n)_{n\in\omega}$ has a limiting cone $(A, (j_n)_{n\in\omega})$ in $\mathcal{C}$. Since $\mathcal{D}$ is a full subcategory, it now suffices to show that the limit object A belongs to $\mathcal{D}$, i.e., that $\mathcal{C}(I, A)$ is compact.

Here we use the characterization of compactness from Section 2: we already know that $\mathcal{C}(I, A)$ is complete, so it remains to show that $\mathcal{C}(I, A)$ is totally bounded. First, by Lemma 3.2 applied to $\mathcal{C}$ and the limiting cone $(A, (j_n)_{n\in\omega})$, there exists a family of morphisms $(i_n : A_n \to A)_{n\in\omega}$ satisfying certain conditions: in particular, $j_n \circ i_n = id_{A_n}$ for each n and $\lim_{n\to\infty} d(i_n \circ j_n, id_A) = 0$. Now we show that $\mathcal{C}(I, A)$ is totally bounded. Given $\epsilon > 0$, choose n large enough that $d(i_n \circ j_n, id_A) < \epsilon$. Since $\mathcal{C}(I, A_n)$ is compact, it is totally bounded. Hence there exists a finite set S of elements of $\mathcal{C}(I, A_n)$ such that for every $f \in \mathcal{C}(I, A_n)$ there is an $s \in S$ with $d(f, s) < \epsilon$. Let T be the finite subset $\{i_n \circ s \mid s \in S\}$ of $\mathcal{C}(I, A)$. Now let a be an arbitrary element of $\mathcal{C}(I, A)$. We show that a has distance less than ϵ to some element of T; hence $\mathcal{C}(I, A)$ is totally bounded. Indeed, choose $s \in S$ such that $d(j_n \circ a, s) < \epsilon$. Then by the ultrametric inequality and the assumption that composition is non-expansive,

$$\begin{aligned} d(a,\, i_n \circ s) &\leq \max(d(a,\, i_n \circ j_n \circ a),\, d(i_n \circ j_n \circ a,\, i_n \circ s)) \\ &\leq \max(d(id_A,\, i_n \circ j_n),\, d(j_n \circ a,\, s)) \\ &< \epsilon\,. \end{aligned}$$

□

Here one obtains $\mathsf{KBUlt}_{\mathrm{ne}}$ by taking $\mathcal{C} = \mathsf{CBUlt}_{\mathrm{ne}}$ and $I = 1$. In general, for a monoidal closed $\mathcal{C}$, the tensor unit is an appropriate choice of I. Since we show in the next section that $\mathsf{CBUlt}_{\mathrm{ne}}$ has limits of ω^{op}-chains of split epis, the theorem in particular gives:

Corollary 4.2 ([10])**.** Every locally contractive functor from $\mathsf{KBUlt}_{\mathrm{ne}}{}^{\mathrm{op}} \times \mathsf{KBUlt}_{\mathrm{ne}}$ to $\mathsf{KBUlt}_{\mathrm{ne}}$ has a unique fixed point up to isomorphism.

Moreover, the proof of the theorem above essentially works by using the hom-functor $\mathcal{C}(I, -) : \mathcal{D} \to \mathsf{KBUlt}_{\mathrm{ne}}$ to reduce the general case to the special case considered in the corollary.

5 Examples of categories admitting solutions

We now turn to some examples of categories that satisfy the different completeness requirements in Theorem 3.4. This section thereby illustrates which of the requirements in that theorem one might attempt to show given a particular M-category.

5.1 CBUlt_*

Consider first the category CBUlt_* of *pointed*, complete, 1-bounded ultrametric spaces. Objects are pairs (A, x) where A is a complete, 1-bounded ultrametric space and x is an element of A (a distinguished 'point'). Morphisms from (A_1, x_1) to (A_2, x_2) are non-expansive maps f from A_1 to A_2 which 'preserve the point', i.e., satisfy that $f(x_1) = x_2$. We equip the hom-sets of CBUlt_* with the 'sup'-metric, as given by the exponential in CBUlt:

$$d(f, g) = \sup\{d_{A_2}(f(x), g(x)) \mid x \in A_1\}\,.$$

Proposition 5.1. CBUlt_* is a complete M-category.

Proof. First, it is easy to see that CBUlt_* is an M-category. Each hom-set is non-empty since it contains the constant function whose value is the distinguished point of the codomain. The distance functions above clearly turn each hom-set into a 1-bounded ultrametric space; indeed, a sub-space of an exponential in CBUlt. Each such space is complete since the limit of a sequence of point-preserving functions is

also point-preserving. The composition functions are non-expansive since they are restrictions of composition functions from CBUlt.

To see that CBUlt_* is complete, it is easy to construct products and equalizers directly (as in CBUlt). More abstractly, CBUlt_* is the comma category $(1 \downarrow \mathsf{CBUlt})$, and the forgetful functor from this category to CBUlt creates limits [16, Exercise V.1.1]. □

5.2 $\mathsf{CBUlt}_{\mathrm{ne}}$

We have already observed that the category $\mathsf{CBUlt}_{\mathrm{ne}}$ of *non-empty*, complete, 1-bounded ultrametric spaces is an M-category with distance functions on hom-sets given as for exponentials. However, unlike CBUlt_*, it is not complete and does not even have all limits of ω^{op}-chains. To see this, let T be a rooted tree that contains nodes of arbitrarily large depth but contains no infinite path (by König's Lemma such a tree must be infinitely branching.) For each n, let A_n be the set of nodes of T of depth n, equipped with the discrete metric. Let $g_n : A_{n+1} \to A_n$ map each node to its parent. Then the ω^{op}-chain $(A_n, g_n)_{n\in\omega}$ in $\mathsf{CBUlt}_{\mathrm{ne}}$ does not have a limit. Indeed, a limit in $\mathsf{CBUlt}_{\mathrm{ne}}$ would also be a limit in CBUlt. But the limit of $(A_n, g_n)_{n\in\omega}$ in the complete category CBUlt is the set of tuples $\{\, (a_n)_{n\in\omega} \mid \forall n.\, g_n(a_{n+1}) = a_n \,\}$ with the product metric; this set is empty since T does not contain any infinite path, and hence the limit does not belong to $\mathsf{CBUlt}_{\mathrm{ne}}$.

Proposition 5.2. $\mathsf{CBUlt}_{\mathrm{ne}}$ is an M-category with limits of ω^{op}-chains of split epis.

Proof. Since $\mathsf{CBUlt}_{\mathrm{ne}}$ is a full subcategory of the complete category CBUlt, it suffices to show that CBUlt-limits of ω^{op}-chains of split epis in $\mathsf{CBUlt}_{\mathrm{ne}}$ are non-empty. Let $(A_n, g_n)_{n\in\omega}$ be such an ω^{op}-chain, and let for each n the function f_n be a right inverse of g_n. A concrete limit in CBUlt of such a chain is, as mentioned above, the set of tuples $\{\, (a_n)_{n\in\omega} \mid \forall n.\, g_n(a_{n+1}) = a_n \,\}$ with the product metric. Now let a_0 be an arbitrary element of A_0 (which is non-empty by assumption). It is easy to see that the limit above contains the tuple $((f_{n-1} \circ \ldots \circ f_0)(a_0))_{n\in\omega}$ and is therefore also non-empty. □

5.3 CBUlt

The category CBUlt is not an M-category since the set of morphisms from any non-empty metric space to the empty metric space is empty. Nevertheless, there is an obvious definition of 'locally contractive' for functors on this category, and given a locally contractive functor $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ that restricts to $\mathsf{CBUlt}_{\mathrm{ne}}$, one can use the main theorem with the category $\mathsf{CBUlt}_{\mathrm{ne}}$ to find a fixed point of F. It is not hard to see that F restricts to $\mathsf{CBUlt}_{\mathrm{ne}}$ if and only if $F(1,1)$ is non-empty (where 1 is the one-point metric space):

Theorem 5.3. Let $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ be a locally contractive functor satisfying that $F(1,1) \neq \emptyset$. There exists a unique (up to isomorphism) non-empty $A \in \mathsf{CBUlt}$ such that $F(A,A) \cong A$.

Proof. We show that F restricts to the full subcategory $\mathsf{CBUlt}_{\mathrm{ne}}$: given non-empty A and B, we must show that $F(A,B)$ is non-empty. Since B is non-empty there exist morphisms $f : A \to 1$ and $g : 1 \to B$ in CBUlt. Then $F(f,g)$ is a function from $F(1,1)$ to $F(A,B)$. Since $F(1,1)$ is non-empty by assumption, the existence of such a function implies that $F(A,B)$ is non-empty too. The theorem now immediately follows from Theorem 3.4 applied to the M-category $\mathsf{CBUlt}_{\mathrm{ne}}$. □

Note that uniqueness is only among non-empty metric spaces: a functor F as in the theorem might furthermore satisfy that $F(\emptyset,\emptyset) = \emptyset$.

5.4 $\mathsf{PreCBUlt}_{\mathrm{ne}}$

The examples in Section 7 use the category $\mathsf{PreCBUlt}_{\mathrm{ne}}$ of *pre-ordered*, non-empty, complete, 1-bounded ultrametric spaces. Objects of this category are pairs $(A, \leq)$ consisting of an object A of $\mathsf{CBUlt}_{\mathrm{ne}}$ and a preorder $\leq$ on the underlying set of A such that the following condition holds: if $(a_n)_{n\in\omega}$ and $(b_n)_{n\in\omega}$ are converging sequences in A with $a_n \leq b_n$ for all n, then also $\lim_{n\to\infty} a_n \leq \lim_{n\to\infty} b_n$. The morphisms of the category are the non-expansive and monotone functions between such objects. We equip the hom-sets of $\mathsf{PreCBUlt}_{\mathrm{ne}}$ with the usual 'sup'-metric.

Proposition 5.4. $\mathsf{PreCBUlt}_{\mathrm{ne}}$ is an M-category with limits of ω^{op}-chains of split epis.

Proof. To see that $\mathsf{PreCBUlt}_{\mathrm{ne}}$ is an M-category we proceed as in the proof of Proposition 5.2. The only new thing to show is that if $(f_n)_{n\in\omega}$ is a converging sequence of monotone and non-expansive functions between objects $(A, \leq_A)$ and $(B, \leq_B)$ of $\mathsf{PreCBUlt}_{\mathrm{ne}}$, then the limit f is a monotone (as well as non-expansive) function. But this follows immediately from the requirement above: if $a \leq_A a'$, then

$$f(a) = \lim_{n\to\infty} f_n(a) \leq_B \lim_{n\to\infty} f_n(a') = f(a')\,.$$

It remains to show that $\mathsf{PreCBUlt}_{\mathrm{ne}}$ has limits of ω^{op}-chains of split epis. Let $(A_n, g_n)_{n\in\omega}$ be such a chain. It is easy to verify that the limit is the set of tuples $\{\, (a_n)_{n\in\omega} \mid \forall n.\, g_n(a_{n+1}) = a_n \,\}$ with the product metric and the product preorder, and that this set is non-empty as in the proof of Proposition 5.2. □

5.5 Locally compact subcategories

We have already seen, using Theorem 4.1, that the full subcategory $\mathsf{KBUlt}_{\mathrm{ne}}$ of $\mathsf{CBUlt}_{\mathrm{ne}}$ has unique fixed points of locally contractive functors. Similarly, that theorem applied to the M-categories CBUlt_* and $\mathsf{PreCBUlt}_{\mathrm{ne}}$ of the previous examples gives unique fixed points of locally contractive functors on the 'compact' variants of these two categories. Notice that for CBUlt_*, the choice $I = 1$ in Theorem 4.1 does not work: one must instead choose I to be the metric space consisting of two points with distance 1. (CBUlt_* is not cartesian closed, but it is symmetric monoidal closed with this I as tensor unit.)

6 An alternative existence theorem

We next consider an alternative existence theorem for solutions of recursive equations in M-categories. Roughly put, the overall picture is as follows. In Section 3 above we generalized the results of America and Rutten [6] to M-categories; this was done in the style of Smyth and Plotkin [22]. In this section, we outline a similar generalization of the results of Alessi et al. [4]. The resulting existence theorem can, at least informally, be viewed as a closer categorical analogy to Banach's fixed-point theorem than the existence theorem in Section 3. In particular it will not be required that the M-category has a terminal object: any object will suffice to start the inductive construction of the solution. On the other hand, the M-category must satisfy a stronger completeness property. We do not know any applications that depend on these slightly different conditions on the category.

Let $\mathcal{C}$ be an arbitrary M-category. In the existence proof in Section 3 we worked extensively with pairs of morphisms (f, g) such that $f : A \to B$ and $g : B \to A$ for some objects A and B of $\mathcal{C}$ and such that $g \circ f = id_A$. Following Alessi et al. [3]

we call such pairs *embedding-projection* pairs.[1] The proof essentially takes place in a category that has such pairs as morphisms; this is made precise in, e.g., America and Rutten [6]. The alternative approach explored in this section does not depend on the embedding condition $g \circ f = id_A$ and so works with all pairs of morphisms with opposite domain and codomain. These pairs were introduced independently as ϵ-*adjoint* pairs in Rutten [18] and as ϵ-*isometries* in Alessi et al. [3]. In Alessi et al. [4] it was shown that the standard existence theorem on non-empty, complete, 1-bounded metric spaces from America and Rutten [6] could be obtained using ϵ-adjoint pairs instead of embedding-projection pairs. Here we outline a generalization of that result to M-categories.

Definition 6.1. The category $\mathcal{C}^{\approx}$ has the objects of $\mathcal{C}$ and morphisms $\iota : A \to B$ that are pairs of morphisms $\iota = \langle i, j\rangle$ of $\mathcal{C}$ such that $i : A \to B$ and $j : B \to A$. Composition of $\iota_1 = \langle i_1, j_1\rangle : A \to B$ and $\iota_2 = \langle i_2, j_2\rangle : B \to C$ is defined naturally by $\iota_2 \circ \iota_1 = \langle i_2 \circ i_1, j_1 \circ j_2\rangle : A \to C$. The identity morphism on the object A is $\langle id_A, id_A\rangle$.

We immediately remark that two objects are isomorphic in $\mathcal{C}$ if and only if they are isomorphic in $\mathcal{C}^{\approx}$ and hence we shall purposely blur the distinction. We may 'flip' any morphism $\iota = \langle i, j\rangle : A \to B$ by swapping the components to obtain a morphism $\bar{\iota} = \langle j, i\rangle : B \to A$.

Definition 6.2. The *noise* of a morphism $\iota = \langle i, j\rangle : A \to B$ in $\mathcal{C}^{\approx}$ is defined as

$$\delta(\iota) = \max(d_{\mathcal{C}(A,A)}(id_A, j \circ i),\ d_{\mathcal{C}(B,B)}(i \circ j, id_B)).$$

Note that we rely on the M-category structure on $\mathcal{C}$ to define the noise but make no attempt to make an M-category out of $\mathcal{C}^{\approx}$.

Intuitively, the noise measures 'how far' A and B are from each other by ι. Having $\delta(\iota) = 0$ obviously implies $j \circ i = id_A$ and $i \circ j = id_B$; in particular two objects are isomorphic if and only if there is a zero-noise morphism from one to the other. Also by definition $\delta(\iota) = \delta(\bar{\iota})$ for any morphism ι of $\mathcal{C}^{\approx}$. These two observations are somewhat analogous to the first and second of the defining axioms of an (ultra)metric space; the following lemma provides a cousin to the ultrametric inequality:

Lemma 6.3 (Noise Lemma)**.** For $\iota_1 : A \to B$ and $\iota_2 : B \to C$ we have $\delta(\iota_2 \circ \iota_1) \leq \max(\delta(\iota_2), \delta(\iota_1))$.

Proof. Write $\iota_1 = \langle i_1, j_1\rangle$ and $\iota_2 = \langle i_2, j_2\rangle$. Then:

$$\begin{aligned}
\delta(\iota_2 \circ \iota_1) &= \delta(\langle i_2 \circ i_1, j_1 \circ j_2\rangle) \\
&= \max(d(id_A,\ j_1 \circ j_2 \circ i_2 \circ i_1),\ d(i_2 \circ i_1 \circ j_1 \circ j_2,\ id_C)) \\
&\leq \max\big(\max(d(id_A,\ j_1 \circ i_1),\ d(j_1 \circ id_B \circ i_1,\ j_1 \circ j_2 \circ i_2 \circ i_1)), \\
&\qquad \max(d(i_2 \circ i_1 \circ j_1 \circ j_2,\ i_2 \circ id_B \circ j_2),\ d(i_2 \circ j_2,\ id_C))\big) \\
&\leq \max\big(\max(\delta(\iota_1), \delta(\iota_2)),\ \max(\delta(\iota_1), \delta(\iota_2))\big) \\
&= \max(\delta(\iota_1), \delta(\iota_2)).
\end{aligned}$$

Here we have used the ultrametric inequality as well as the ubiquitous fact that the composition functions of an M-category are non-expansive. □

In a metric space, to prove two elements equal it suffices to show that their distance is smaller than every $\epsilon > 0$. The corresponding technique in our metric-inspired setting is the following:

[1] We do not, however, use any analogue of the 'projection' condition $f \circ g \sqsubseteq id_B$ from the domain-theoretic case.

Lemma 6.4 (Proximity Lemma)**.** Two objects A and B of $\mathcal{C}$ are isomorphic if there is a sequence of $\mathcal{C}^{\approx}$-morphisms $(\langle i_n, j_n\rangle)_{n\in\omega}$ with $\langle i_n, j_n\rangle : A \to B$ such that $\lim_{n\to\infty} \delta(\langle i_n, j_n\rangle) = 0$ and such that $(i_n)_{n\in\omega}$ and $(j_n)_{n\in\omega}$ are Cauchy sequences in $\mathcal{C}(A, B)$ and $\mathcal{C}(B, A)$, respectively.

Proof. By completeness of $\mathcal{C}(A, B)$ and $\mathcal{C}(B, A)$ we know that $\lim_{n\to\infty} i_n : A \to B$ and $\lim_{n\to\infty} j_n : B \to A$ exist. We now have:

$$\begin{aligned} d_{\mathcal{C}(A,A)}\left(id_A, \lim_{n\to\infty} j_n \circ \lim_{n\to\infty} i_n\right) &= d_{\mathcal{C}(A,A)}\left(id_A, \lim_{n\to\infty} j_n \circ i_n\right) \\ &= \lim_{n\to\infty} d_{\mathcal{C}(A,A)}(id_A, j_n \circ i_n) \\ &\leq \lim_{n\to\infty} \delta(\iota_n) \\ &= 0 \end{aligned}$$

Here we have used non-expansiveness of composition and the fact that, for any ultrametric space (X, d), the distance function $d : X \times X \to \mathbb{R}$ is itself non-expansive and hence preserves limits. We conclude that $id_A = \lim_{n\to\infty} j_n \circ \lim_{n\to\infty} i_n$ and by symmetry we get the other way round. □

By analogy with the standard metric argument one might try to do away with the second demand that the component sequences be Cauchy. However, as observed in Remark 4.4 of Alessi et al. [4], this is not possible. A consequence is that a 'proper' distance between two objects defined as the infimum of the noises of morphisms from one to the other gives only a pseudo-metric; that is, the distance between two distinct objects can be zero. This problem is explored in Section 4 of Alessi et al. [4] and solved by restricting to compact metric spaces.

Definition 6.5. A *tower* in $\mathcal{C}^{\approx}$ is a sequence of pairs of objects and morphisms (A_n, ι_n) such that $\iota_n : A_n \to A_{n+1}$ for all $n \in \omega$. It is *Cauchy* if $\lim_{n\to\infty} \delta(\iota_n) = 0$, i.e., if

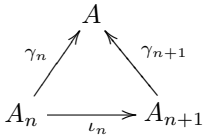
$$\forall \epsilon > 0.\ \exists N \in \mathbb{N}.\ \forall n \geq N.\ \delta(\iota_n) < \epsilon.$$

Notice that a Cauchy tower $(A_n, \iota_n)_{n\in\omega}$ where all the ι_n are embedding-projection pairs is exactly an 'increasing Cauchy tower' as defined in Section 3.

As in the case of standard Cauchy sequences, the objects of a Cauchy tower intuitively get arbitrarily close, measured here by the noises of the morphisms. By the Noise Lemma, i.e., due to our ultrametric setup, we immediately have that the above criterion is equivalent to one that may look more familiar:

$$\forall \epsilon > 0.\ \exists N \in \mathbb{N}.\ \forall m > n \geq N.\ \delta(\iota_{m-1} \circ \cdots \iota_n) < \epsilon.$$

Definition 6.6. A *limit* of a Cauchy tower $(A_n, \iota_n)_{n\in\omega}$ is a pair $(A, (\gamma_n)_{n\in\omega})$ of an object and a sequence of morphisms $\gamma_n : A_n \to A$ in $\mathcal{C}^{\approx}$ such that

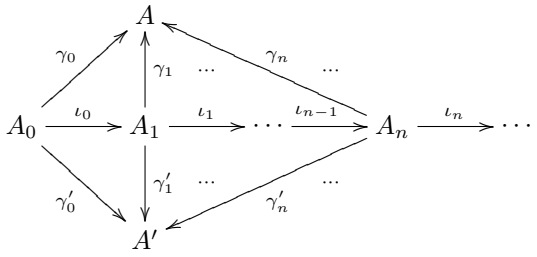


commutes for all $n \in \omega$ and such that $\lim_{n\to\infty} \delta(\gamma_n) = 0$.

(Proposition 6.8 below relates limits of Cauchy towers in the sense above to inverse limits of the kind considered in Section 3.)

Proposition 6.7. For any two limits $(A, (\gamma_n)_{n\in\omega})$ and $(A', (\gamma'_n)_{n\in\omega})$ of the same Cauchy tower $(A_n, \iota_n)_{n\in\omega}$ the objects A and A' are isomorphic.

Proof. This comes down to applying the Proximity Lemma. The setup is this:



We have $\gamma'_n \circ \overline{\gamma_n} : A \to A'$ for all $n \in \omega$, and since $\lim_n \delta(\gamma_n) = \lim_n \delta(\overline{\gamma_n}) = 0 = \lim_n \delta(\gamma'_n)$ we get $\lim_n \delta(\gamma'_n \circ \overline{\gamma_n}) = 0$ by the Noise Lemma. Now write $\gamma_n = \langle g_n, h_n \rangle$ and $\gamma'_n = \langle g'_n, h'_n \rangle$ for all $n \in \omega$. It remains to show that $(g'_n \circ h_n)_{n\in\omega}$ and $(g_n \circ h'_n)_{n\in\omega}$ are Cauchy sequences in the metric spaces $\mathcal{C}(A, A')$ and $\mathcal{C}(A', A)$, respectively. For any $n \in \omega$

$$\begin{aligned} d_{\mathcal{C}(A,A')}(g'_n \circ h_n, g'_{n+1} \circ h_{n+1}) &= d_{\mathcal{C}(A,A')}(g'_{n+1} \circ i_n \circ j_n \circ h_{n+1}, g'_{n+1} \circ h_{n+1}) \\ &= d_{\mathcal{C}(A_{n+1},A_{n+1})}(i_n \circ j_n, id_{A_{n+1}}) \\ &\leq \delta(\iota_n) \end{aligned}$$

where we write $\iota_n = \langle i_n, j_n \rangle$. But then $(g'_n \circ h_n)_{n\in\omega}$ is Cauchy because $(A_n, \iota_n)_{n\in\omega}$ is a Cauchy tower. By symmetry $(g_n \circ h'_n)_{n\in\omega}$ is also Cauchy. □

Notice how we use our ability to flip a morphism $\iota : A \to B$ to obtain $\bar{\iota} : B \to A$; in the category of embedding-projection pairs this is not possible in general.

We say that $\mathcal{C}^{\approx}$ is *tower-complete* if all Cauchy towers have limits. Verifying this condition directly may be an arduous task. The following criterion is sufficient:

Proposition 6.8. $\mathcal{C}^{\approx}$ is tower-complete if $\mathcal{C}$ has inverse limits of Cauchy towers.

We omit the proof. The arguments follow those in the first part of the proof of Lemma 3.2, but are more involved since we no longer restrict to embedding-projection pairs. More specifically, the cone $(h^m_n)_{n\in\omega}$ from A_m to $(A_n, g_n)_{n\in\omega}$ in that proof must now be defined as follows:

$$h^m_n = \begin{cases} k_n, & \text{if } n = m, \\ g_n \circ g_{n+1} \circ \cdots \circ g_{m-1} \circ k_m, & \text{if } n < m, \\ k_n \circ f_{n-1} \circ f_{n-2} \circ \cdots \circ f_m, & \text{if } n > m. \end{cases}$$

where each $k_n : A_n \to A_n$ is obtained as a limit of a Cauchy sequence:

$$k_n = \lim_{p\geq n} (g_n \circ g_{n+1} \circ \cdots \circ g_{p-1} \circ f_p \circ f_{p-1} \circ \cdots \circ f_n)\,.$$

For a domain-theoretic analogue of dropping the restriction to embedding-projection pairs, see Taylor [23].

6.1 Fixed points of Functors

We now move on to apply the theory to build fixed points of functors. We say that a functor $\Phi : \mathcal{C}^{\approx} \to \mathcal{C}^{\approx}$ is *contractive* if there is a $c < 1$ such that $\delta(\Phi(\iota)) \leq c \cdot \delta(\iota)$ holds for all morphisms ι of $\mathcal{C}^{\approx}$. Similarly it is called *non-expansive* if the noises do not increase, i.e., if $\delta(\Phi(\iota)) \leq \delta(\iota)$ holds for all ι. We may build functors on $\mathcal{C}^{\approx}$ from functors on $\mathcal{C}$:

Proposition 6.9. Let $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$. We define $F^{\approx} : \mathcal{C}^{\approx} \to \mathcal{C}^{\approx}$ by

$$F^{\approx}(A) = F(A, A), \quad F^{\approx}(\langle i, j \rangle) = \langle F(j, i), F(i, j) \rangle$$

for any object A and any morphism $\iota = \langle i, j \rangle : A \to B$ of $\mathcal{C}^{\approx}$. This constitutes a well defined functor. Moreover, if F is locally contractive then $F^{\approx}$ is contractive and if F is locally non-expansive then $F^{\approx}$ is non-expansive.

We saw in Theorem 3.1 that a locally contractive functor $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ has at most one fixed point up to isomorphism (for arbitrary $\mathcal{C}$.) One can give an alternative proof of that fact using the Proximity Lemma and the contractive functor $F^{\approx}$ derived from F. But even in the concrete case $\mathcal{C} = \mathsf{CBUlt}_{\mathrm{ne}}$ it is an open question whether every contractive endofunctor on $\mathcal{C}^{\approx}$ has at most one fixed point [3, p. 7].

Just as non-expansive maps between metric spaces are continuous and thus preserve limits of sequences, we have the following proposition as an immediate consequence of the above definitions:

Proposition 6.10. Non-expansive functors preserve limits of Cauchy towers. That is, for any non-expansive functor $\Phi : \mathcal{C}^{\approx} \to \mathcal{C}^{\approx}$ and any Cauchy tower $(A_n, \iota_n)_{n \in \omega}$ with limit $(A, (\gamma_n)_{n \in \omega})$ we have that $(\Phi(A_n), \Phi(\iota_n))_{n \in \omega}$ is a Cauchy tower with limit $(\Phi(A), (\Phi(\gamma_n))_{n \in \omega})$.

Theorem 6.11. If $\mathcal{C}^{\approx}$ is nonempty and tower-complete then any contractive functor $\Phi : \mathcal{C}^{\approx} \to \mathcal{C}^{\approx}$ has a fixed point, i.e., an object A of $\mathcal{C}^{\approx}$ with $A \cong \Phi(A)$.

Proof. Much of the theory above targets this proof; it is quite short and analogous to the proof of Banach's fixed-point theorem.

Let A_0 be any object of $\mathcal{C}^{\approx}$ and define $A_{n+1} = \Phi(A_n)$ for every $n \in \omega$. Let $\iota_0 : A_0 \to A_1$ be any morphism of $\mathcal{C}^{\approx}$ and define $\iota_{n+1} = \Phi(\iota_n) : A_{n+1} \to A_{n+2}$ for every $n \in \omega$. We can always initiate this process: $\mathcal{C}^{\approx}$ was assumed to have an object, and $\iota_0 : A_0 \to A_1$ always exists as the hom-sets of an M-category are non-empty.

It is immediate by the contractiveness of Φ that $(A_n, \iota_n)_{n \in \omega}$ is a Cauchy tower and hence has a limit $(A, (\gamma_n)_{n \in \omega})$ as $\mathcal{C}^{\approx}$ was assumed tower-complete. A contractive functor is in particular non-expansive and non-expansive functors preserves limits of Cauchy towers, so $(\Phi(A_n), \Phi(\iota_n))_{n \in \omega} = (A_{n+1}, \iota_{n+1})_{n \in \omega}$ is a Cauchy tower too with limit $(\Phi(A), (\Phi(\gamma_n))_{n \in \omega})$. But $(A, (\gamma_{n+1})_{n \in \omega})$ is a limit of $(A_{n+1}, \iota_{n+1})_{n \in \omega}$ too and uniqueness of limits (Proposition 6.7) gives $A \cong \Phi(A)$. □

Combining Proposition 6.8, Proposition 6.9, and Theorem 6.11 we have:

Theorem 6.12. If $\mathcal{C}$ has an object and has inverse limits of Cauchy towers then every locally contractive functor $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ has a unique fixed point up to isomorphism.

Notice that here we require *all* Cauchy towers to have inverse limits, not just the increasing ones. Therefore Theorem 6.12 does not immediately imply Theorem 3.4.

7 Applications

This section contains a series of examples of recursive equations motivated by recent and ongoing work in semantics. In all but the first of the examples we do not consider exactly those equations that arise from applications; for clarity we consider simplified variants that capture the essence of the circularity issues. We conclude the section by discussing in what sense the generality of M-categories is needed in applications.

7.1 Realizability semantics of dynamically allocated store

The first two examples of recursive equations come from realizability semantics of dynamically allocated store. In recent work [9] the authors presented a model that allows for simple parametricity-style reasoning about imperative abstract data types in an ML-like language with universal types, recursive types, and reference types. As in Standard ML, references are dynamically allocated during program execution.

Here is a brief outline of the model. First, the model is based on a realizability interpretation [5] over a certain recursively defined predomain V. In addition, we follow earlier work on modeling simple integer references [8] and use a Kripke-style possible worlds model. Here, however, the set of worlds needs to be recursively defined since we treat general references. Semantically, a world maps locations to semantic types, which, following the general realizability idea, are certain world-indexed families of relations on V: this introduces a circularity between semantic types and worlds that precludes a direct definition of either. Thus we are led to solving recursive (metric-space) equations of approximately the following form

$$\begin{aligned} \mathcal{W} &\cong \mathbb{N} \rightharpoonup_{fin} \mathcal{T} \\ \mathcal{T} &\cong \mathcal{W} \to_{mon} \mathit{CURel}(V) \end{aligned}$$

(see below) even in order to define the space in which types will be modeled.

We now describe these equations in more detail. $\mathit{CURel}(V)$ is the set of binary relations on V that satisfy certain technical requirements. The metric on $\mathit{CURel}(V)$ is defined essentially as in earlier work on realizability semantics [5], using the fact that V is a canonical solution to a predomain equation. The space $\mathbb{N} \rightharpoonup_{fin} \mathcal{T}$ consists of partial functions from $\mathbb{N}$ to $\mathcal{T}$ with finite domain: the distance between two functions with different domains is 1, while the distance between two functions with the same domain is given as a maximum of pointwise distances. The space $\mathbb{N} \rightharpoonup_{fin} \mathcal{T}$ (and hence also $\mathcal{W}$) is equipped with an extension order: for $\Delta, \Delta' \in \mathbb{N} \rightharpoonup_{fin} \mathcal{T}$ we take $\Delta \leq \Delta'$ to mean that $\mathrm{dom}(\Delta) \subseteq \mathrm{dom}(\Delta')$ and that $\Delta(n) = \Delta'(n)$ for all n in $\mathrm{dom}(\Delta)$. Finally, in order to ensure soundness of the interpretation, we require the usual 'Kripke monotonicity': the space $\mathcal{W} \to_{mon} \mathit{CURel}(V)$ should consist of functions that are both non-expansive and monotone with respect to the extension order on $\mathcal{W}$ and the inclusion order on $\mathit{CURel}(V)$.

In order to apply the main theorem to solve these equations, we have to express them in terms of a mixed-variance functor on an M-category. There are two approaches. First, one can 'solve for worlds' by defining a contravariant functor F on $\mathsf{PreCBUIt}_{\mathrm{ne}}$ such that

$$F(X, \leq) = (\mathbb{N} \rightharpoonup_{fin} \tfrac{1}{2}((X, \leq) \to_{mon} \mathit{CURel}(V))),\ \leq')$$

where $\leq'$ is the extension order on partial functions, as defined above. (Here the $\frac{1}{2}$ is needed in order to ensure that F is locally contractive.) Then $(\mathcal{W}, \leq)$ can be defined as the unique fixed point of F.

Alternatively, one can 'solve for types' [9] by defining a contravariant functor G on $\mathsf{CBUIt}_{\mathrm{ne}}$ (or on CBUIt as in Section 5.3) such that

$$G(X) = \tfrac{1}{2}((\mathbb{N} \rightharpoonup_{fin} X) \to_{mon} \mathit{CURel}(V))\,.$$

Then $\mathcal{T}$ can be defined as the unique fixed point of G. In this case we do not use the generality of M-categories: instead we exploit that the two mutually recursive equations above have a form that allows one to solve them in $\mathsf{CBUIt}_{\mathrm{ne}}$ by combining them into a single recursive equation in the right way. In the next example such an approach will not be possible; there, M-categories seem to be needed.

Remark. Even though metric spaces appear naturally in this example through earlier work on realizability, one might ask whether the equations above cannot instead be solved in a category of domains. Indeed they can, but the solution does not appear to be useful for our purpose. The main reason is that the quantitative information given by the metric-space approach appears to be needed in order to model reference types [9].

7.2 A more advanced model of store

In the previous example, semantic types were modeled as world-indexed families of binary relations on the predomain V of 'untyped values'. The intuitive idea is that worlds provide information about the currently allocated references, and that the interpretation of a type grows as more references are allocated.

The fact that relations are binary allows one to use the model to prove equivalences between programs that allocate references dynamically. However, the set of worlds $\mathcal{W} \cong \mathbb{N} \rightharpoonup_{fin} \mathcal{T}$ of the previous example only allows for fairly limited equivalence proofs. There, a world is no more than a single 'semantic store typing' that only allows one to describe situations where the two programs under consideration allocate references in lockstep.

Ongoing work suggests that the metric-space approach allows one to solve an equation involving more advanced Kripke worlds in the style of Ahmed et al. [2], and thereby allows for more advanced reasoning about local state.[2] Here we present a simplified equation that illustrates the main circularity issue. Let $S = \mathbb{N} \rightharpoonup_{fin} V$ be the set of *stores*, i.e., partial maps with finite domain from $\mathbb{N}$ to V. Whereas the simple worlds of the previous example induce a binary relation on stores that require two related stores to have the same domain, we now seek an alternative definition of worlds that induce a more liberal relation on stores.

The intuitive idea is that a world $\mathcal{W}'$ consists of a finite sequence of 'islands' I [2], each of which induces a local requirement on stores by describing how two specific parts of two given stores are required to be related. Consider the metric-space equations

$$\begin{aligned} \mathcal{W}' &\cong I^* \\ I &\cong \sum_{N_1, N_2 \in P_{\mathrm{fin}}(\mathbb{N})} \tfrac{1}{2} \left(\mathcal{W}' \to_{mon} \mathit{CURel}(S)_{N_1,N_2}\right) \end{aligned}$$

which are to be understood as follows. The space $\mathit{CURel}(S)$ [9] consists of binary relations on stores satisfying certain technical conditions; it is equipped with a metric in the same way as $\mathit{CURel}(V)$ above. Given finite subsets N_1 and N_2 of $\mathbb{N}$, the sub-space $\mathit{CURel}(S)_{N_1,N_2}$ of $\mathit{CURel}(S)$ only contains relations with *support* (N_1, N_2), i.e., $R \in \mathit{CURel}(S)_{N_1,N_2}$ and $(s_1, s_2) \in R$ implies $(s_1', s_2') \in R$ if $s_1(n) = s_1'(n)$ for all n in N_1 and $s_2(n) = s_2'(n)$ for all n in N_2. Intuitively, such relations are local in the sense that they only depend on locations from N_1 and N_2, respectively. The sum on the right-hand side of the second equation consists of triples (N_1, N_2, f); the distance between two such triples is 1 if either of the first two components differ, and the distance between the third components otherwise. Finally, assuming that the second equation holds, the space I^* consists of finite sequences of triples (N_1, N_2, f) such that the first components are pairwise disjoint, and similarly for the second components: the 'islands' must not overlap. The extension order on I^*, and hence on $\mathcal{W}'$, is sequence containment; the maps of the second equation are monotone with respect to this order and the inclusion order on $\mathit{CURel}(S)_{N_1,N_2}$.

[2] Ahmed et al. do not solve the recursive equation they consider, but instead work with a family of sets that are, intuitively, approximations to a solution.

Because of the different dependencies on finite subsets of integers, it does not seem possible to combine the two equations above into one equation in $\mathsf{CBUlt}_{\mathrm{ne}}$: some extra structure on metric spaces is needed, no matter what one tries to 'solve for'. Indeed, the equations can be 'solved for worlds' by defining a contravariant functor H on $\mathsf{PreCBUlt}_{\mathrm{ne}}$ directly from the equations,

$$H(X,\leq) = \left(\left(\sum_{N_1,N_2 \in P_{\mathrm{fin}}(\mathbb{N})} \tfrac{1}{2}\left((X,\leq) \to_{mon} \mathit{CURel}_{N_1,N_2}(S)\right) \right)^*, \leq' \right)$$

where $\leq'$ is the extension order on sequences, and then letting $\mathcal{W}'$ be the unique fixed point of H.

7.3 Storable locks

In recent work by Gotsman et al. separation logic has been extended to reason about storable locks and threads [14]. As observed in *loc. cit.* the natural model of predicates involves a circular definition because locks protecting invariants (predicates) can be stored in the heap. However, Gotsman et al. side-step this issue by restricting the storable locks to protect only a statically determined finite set of kinds of invariants. In ongoing work, Birkedal and Buisse are generalizing the work by Gotsman et al. by solving a suitable recursive equation. The equation is

$$\mathit{UPred} \cong \tfrac{1}{2}\Big(\big(\mathbb{N} \rightharpoonup_{\mathit{fin}} (\mathbb{N} + (\mathbb{N} \times \mathit{UPred}))\big) \to_{mon} P{\downarrow}(\mathbb{N}) \Big).$$

Here $P{\downarrow}(\mathbb{N})$ is the complete, bounded ultrametric space consisting of downwards-closed subsets of $\mathbb{N}$; this set forms a complete Heyting algebra. The idea is that a semantic predicate is a $P{\downarrow}(\mathbb{N})$-valued predicate on heaps, which are maps from locations (numbers) to either numbers or pairs (k, I) consisting of a thread id k and a semantic predicate I. The latter is used if the location is a lock, held by thread k and protecting the invariant I.

This equation can be solved in $\mathsf{CBUlt}_{\mathrm{ne}}$ by solving for *UPred* (much as in the example in Section 7.1), or by solving for heaps by defining a contravariant functor on $\mathsf{PreCBUlt}_{\mathrm{ne}}$.

7.4 Semantics of nested Hoare triples

In recent work, Schwinghammer et al. [20] investigate the semantics of separation logic for higher-order store. There uniform admissible subsets of heaps form the basic building block when interpreting the assertions of the logic. Since assertions in general depend on invariants for stored code (because of higher-order store), the space of semantic predicates consists of functions $\mathcal{W} \to \mathit{UAdm}$ from a set of 'worlds,' describing the invariants, to the collection of uniform admissible subsets of heaps. The set *UAdm* is an ultrametric space with metric given as for $\mathit{CURel}(V)$ above. But, the invariants for stored code are themselves semantic predicates, and hence the space of worlds $\mathcal{W}$ should be 'the same' as $\mathcal{W} \to \mathit{UAdm}$. Thus the following equation is solved in $\mathsf{CBUlt}_{\mathrm{ne}}$:

$$\mathcal{W} \cong \tfrac{1}{2}\left(\mathcal{W} \to \mathit{UAdm}\right).$$

7.5 Discussion

As we have seen, three of the four examples above could be treated by solving recursive equations in $\mathsf{CBUlt}_{\mathrm{ne}}$, i.e., without using the generality of M-categories. The fourth example, the advanced model of store in Section 7.2, does seem to require

M-categories; however, a recent simplification [13] leads to an equation that can be solved in $\mathsf{CBUlt}_{\mathrm{ne}}$.

In a little more detail, the general situation is the following. Instead of directly finding a fixed point of an endofunctor $F : \mathcal{C} \to \mathcal{C}$ on an M-category, one can attempt to express the functor as $F = G \circ H$ for functors $H : \mathcal{C} \to \mathsf{CBUlt}_{\mathrm{ne}}$ and $G : \mathsf{CBUlt}_{\mathrm{ne}} \to \mathcal{C}$, and then find a fixed point of $H \circ G : \mathsf{CBUlt}_{\mathrm{ne}} \to \mathsf{CBUlt}_{\mathrm{ne}}$ instead of F. In this way, only the classical existence theorem is needed. (It could of course also be that one came up with $H \circ G$ instead of F in the first place.) In the first example above, this is the approach that could be used to 'solve for types' instead of 'solving for worlds.'

Despite of this possibility of rewriting a recursive equation to an equation in $\mathsf{CBUlt}_{\mathrm{ne}}$, we find M-categories genuinely useful in applications for the following reasons. First, there is no guarantee that one can rewrite a given recursive equation; as already noted, we cannot see how to treat the example in Section 7.2 by solving an equation in $\mathsf{CBUlt}_{\mathrm{ne}}$. Second, and more subtly: in the first and third examples above, the 'rewriting' approach still depends on the fact that $\mathsf{PreCBUlt}_{\mathrm{ne}}$ is an M-category. More specifically, to obtain a well-defined endofunctor on $\mathsf{CBUlt}_{\mathrm{ne}}$ in these examples, one implicitly uses the defining property of objects of $\mathsf{PreCBUlt}_{\mathrm{ne}}$ that guarantees that hom-sets of $\mathsf{PreCBUlt}_{\mathrm{ne}}$ are *complete* metric spaces. Without having identified $\mathsf{PreCBUlt}_{\mathrm{ne}}$ as an M-category, this leads to some ad-hoc calculations and results about monotone function spaces, as can be seen in Birkedal et al. [9].

8 Domain equations: from O-categories to M-categories

As another illustration of M-categories, we present a general construction that gives for every O-category $\mathcal{C}$ (see below) a derived M-category $\mathcal{D}$. In addition, the construction gives for every locally continuous mixed-variance functor F on $\mathcal{C}$ a locally contractive mixed-variance functor G on $\mathcal{D}$ such that a fixed point of G, necessarily unique by Theorem 3.1, is the same as a fixed point of F that furthermore satisfies the 'minimal invariance' condition of Pitts [17].[3] Thus, generalized domain equations can be solved in M-categories.

The construction generalizes and improves an earlier one due to Baier and Majster-Cederbaum (BM) [7] which is for the particular category $\mathsf{Cppo}_{\perp}$ of pointed cpos and strict, continuous functions (or full subcategories thereof.) More precisely, taking $\mathcal{C}$ to be a full subcategory of $\mathsf{Cppo}_{\perp}$ in our Proposition 8.2 below gives a result that strengthens Lemma 4.18 of BM. In general, the goal of that earlier work is to relate recursive domain equations over full subcategories of $\mathsf{Cppo}_{\perp}$ to recursive equations over full subcategories of a particular category CMS of complete metric spaces. Working with those particular categories instead of arbitrary O-categories and M-categories complicates the relations one can obtain: for example, Theorem 3 of BM only applies to a restricted class of domain equations that does not include general function spaces. The reason is that the construction of BM, which must be applied to a full subcategory of $\mathsf{Cppo}_{\perp}$, does not yield a category that is (in any obvious way) a full subcategory of CMS. It is, however, an M-category in which every locally contractive functor has a unique fixed point. We hence believe to have at least partially answered the question left open in the conclusion of BM whether a suitable notion of correspondence exists for general domain equations.

Rank-ordered cpos [7], recently re-discovered under the name 'uniform cpos' [9], arise from a particular instance of an M-category obtained from the construction

[3] The latter is in turn the same as a bifree algebra for F in the same sense as in Theorem 3.1. See the argument in Pitts [17].

here, namely by taking $\mathcal{C} = \mathsf{Cppo}_\perp$. The extra metric information in that category, as compared with the underlying O-category, is useful in realizability models [1, 5]; in particular it is used to define the metric on the set of binary relations $CURel(V)$ in the work described in Section 7.1 [9]. In earlier work, Abadi and Plotkin [1, Section 8] give a metric-space formulation of a realizability model of polymorphism and recursive types. They note that the extra metric information can be used to model subtypes and bounded quantification.

We now turn to the results. An *O-category* [22] is a category $\mathcal{C}$ where each hom-set $\mathcal{C}(A, B)$ is equipped with an ω-complete partial order, usually written $\sqsubseteq$, and where each composition function is continuous with respect to these orders. A functor $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ is *locally continuous* if each function on hom-sets that it induces is continuous.

Assume now that $\mathcal{C}$ is an O-category such that each hom-set $\mathcal{C}(A, B)$ contains a least element $\perp_{A,B}$ and such that the composition functions of $\mathcal{C}$ are strict: $f \circ \perp_{A,B} = \perp_{A,C} = \perp_{B,C} \circ g$ for all f and g. We construct an M-category $\mathcal{D}$ of 'rank-ordered $\mathcal{C}$-objects' as follows. An object $(A, (\pi_n)_{n\in\omega})$ of $\mathcal{D}$ is a pair consisting of an object A of $\mathcal{C}$ and a family of endomorphisms $\pi_n : A \to A$ in $\mathcal{C}$ that satisfies the following four requirements:

(1) $\pi_0 = \perp_{A,A}$.

(2) $\pi_m \sqsubseteq \pi_n$ for all $m \leq n$.

(3) $\pi_m \circ \pi_n = \pi_n \circ \pi_m = \pi_{\min(m,n)}$ for all m and n.

(4) $\bigsqcup_{n\in\omega} \pi_n = id_A$.

(See also the rank-ordered sets of Baier and Majster-Cederbaum [7] and the projection spaces of de Vries [12].) Then, a morphism from $(A, (\pi_n)_{n\in\omega})$ to $(A', (\pi'_n)_{n\in\omega})$ in $\mathcal{D}$ is a morphism f from A to A' in $\mathcal{C}$ that is *uniform* [1] in the sense that $\pi'_n \circ f = f \circ \pi_n$ for all n. Composition and identities in $\mathcal{D}$ are the same as in $\mathcal{C}$. Finally, the distance function on a hom-set $\mathcal{D}((A, (\pi_n)_{n\in\omega}), (A', (\pi'_n)_{n\in\omega}))$ is defined as follows:

$$d(f,g) = \begin{cases} 2^{-\max\{\, n\in\omega \mid \pi'_n \circ f = \pi'_n \circ g \,\}} & \text{if } f \neq g \\ 0 & \text{if } f = g. \end{cases}$$

To see that d is well-defined, suppose that $f \neq g$. Then there must exist a greatest number n such that $\pi'_n \circ f = \pi'_n \circ g$. Indeed, $n = 0$ is such a number by (1) above and strictness of the composition functions of $\mathcal{C}$. If the equation holds for arbitrarily large n, then by (3) above it holds for all n. But then by (4) above and the fact that the composition functions of $\mathcal{C}$ are continuous,

$$f = id_{A'} \circ f = \Big(\bigsqcup_{n\in\omega} \pi'_n \Big) \circ f = \bigsqcup_{n\in\omega} (\pi'_n \circ f) = \bigsqcup_{n\in\omega} (\pi'_n \circ g) = \cdots = g\,,$$

a contradiction. Hence d is well-defined.

Proposition 8.1. $\mathcal{D}$ is an M-category.

Proof. First, each hom-set $\mathcal{D}((A, (\pi_n)_{n\in\omega}), (A', (\pi'_n)_{n\in\omega}))$ is non-empty: it contains the element $\perp_{A,A'}$ since $\pi'_n \circ \perp_{A,A'} = \perp_{A,A'} \circ \pi_n = \perp_{A,A'}$ by strictness. Second, it is easy to see that the distance function on such a hom-set gives rise to a 1-bounded ultrametric space. Third, the composition functions of $\mathcal{D}$ are non-expansive: it suffices to see that if $\pi''_n \circ f_1 = \pi''_n \circ f_2$ and $\pi'_n \circ g_1 = \pi'_n \circ g_2$, then $\pi''_n \circ (f_1 \circ g_1) = \pi''_n \circ f_2 \circ g_1 = f_2 \circ \pi'_n \circ g_1 = f_2 \circ \pi'_n \circ g_2 = \pi''_n \circ (f_2 \circ g_2)$.

It remains to show that each hom-set is a complete metric space. Let $(f_m)_{m\in\omega}$ be a Cauchy sequence. It follows from the definition of d that for each $n \in \omega$ there

exists g_n such that $\pi'_n \circ f_m = g_n$ for all sufficiently large m. Then by (2) above the sequence $(g_n)_{n\in\omega}$ is increasing: given n, we have $g_n = \pi'_n \circ f_m \sqsubseteq \pi'_{n+1} \circ f_m = g_{n+1}$ for all sufficiently large m.

The supremum $g = \sqcup_{n\in\omega} g_n$ is the limit of the sequence $(f_m)_{m\in\omega}$. Indeed, given an arbitrary number k, by (3) we have $\pi'_k \circ g_n = \pi'_k \circ \pi'_n \circ f_m = \pi'_k \circ f_m = g_k$ for all $n \geq k$ (and sufficiently large m), and therefore, by continuity of composition, $\pi'_k \circ (\sqcup_{n\in\omega} g_n) = \sqcup_{n\geq k}(\pi'_k \circ g_n) = g_k = \pi'_k \circ f_m$ for all sufficiently large m. This shows that $d(f_m, g) \leq 2^{-k}$ for all sufficiently large m. Hence g is indeed the limit of the sequence $(f_m)_{m\in\omega}$.

We must show that g is uniform. First, each g_n is uniform since for all k and all sufficiently large m we have $\pi'_k \circ g_n = \pi'_k \circ \pi'_n \circ f_m = \pi'_n \circ \pi'_k \circ f_m = \pi'_n \circ f_m \circ \pi_k = g_n \circ \pi_k$. Second, g is uniform since each g_n is: $\pi'_k \circ (\sqcup_{n\in\omega} g_n) = \sqcup_{n\in\omega}(\pi'_k \circ g_n) = \sqcup_{n\in\omega}(g_n \circ \pi_k) = (\sqcup_{n\in\omega} g_n) \circ \pi_k$. In conclusion, each hom-set is complete, and $\mathcal{D}$ is an M-category. □

Now let $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ be a locally continuous functor. We construct a locally contractive functor $G : \mathcal{D}^{\mathrm{op}} \times \mathcal{D} \to \mathcal{D}$ from F:

- On objects, G is given by

$$G((A, (\pi_n^A)_{n\in\omega}), (B, (\pi_n^B)_{n\in\omega})) = (F(A, B), (\pi_n^{A,B})_{n\in\omega})$$

  where $\pi_0^{A,B} = \bot$ and $\pi_{n+1}^{A,B} = F(\pi_n^A, \pi_n^B)$ for all n.

- On morphisms, G is the same as F, i.e., $G(f, g) = F(f, g)$.

To see that G is well-defined on objects, we must verify conditions (1)-(4) in the definition of objects of $\mathcal{D}$. Here (1) is immediate, (3) follows from strictness of composition and functoriality of F, and (2) and (4) follow from local continuity of F. In addition, given morphisms $f : (A', (\pi_n^{A'})_{n\in\omega}) \to (A, (\pi_n^A)_{n\in\omega})$ and $g : (B, (\pi_n^B)_{n\in\omega}) \to (B', (\pi_n^{B'})_{n\in\omega})$, we must show that $G(f, g) = F(f, g)$ is a well-defined morphism in $\mathcal{D}$. Clearly, $\pi_0^{A',B'} \circ F(f, g) = F(f, g) \circ \pi_0^{A,B}$, and for all n,

$$\begin{aligned}
\pi_{n+1}^{A',B'} \circ F(f, g) &= F(\pi_n^{A'}, \pi_n^{B'}) \circ F(f, g) \\
&= F(f \circ \pi_n^{A'}, \pi_n^{B'} \circ g) \\
&= F(\pi_n^A \circ f,\ g \circ \pi_n^B) \\
&= F(f, g) \circ F(\pi_n^A, \pi_n^B) \\
&= F(f, g) \circ \pi_{n+1}^{A,B}\,.
\end{aligned}$$

Finally, G is locally contractive with factor $1/2$: it suffices to see that if $\pi_n^A \circ f_1 = \pi_n^A \circ f_2$ and $\pi_n^{B'} \circ g_1 = \pi_n^{B'} \circ g_2$, then

$$\begin{aligned}
\pi_{n+1}^{A',B'} \circ F(f_1, g_1) &= F(\pi_n^{A'}, \pi_n^{B'}) \circ F(f_1, g_1) \\
&= F(f_1 \circ \pi_n^{A'}, \pi_n^{B'} \circ g_1) \\
&= F(\pi_n^A \circ f_1,\ \pi_n^{B'} \circ g_1) \\
&= F(\pi_n^A \circ f_2,\ \pi_n^{B'} \circ g_2) \\
&= \pi_{n+1}^{A',B'} \circ F(f_2, g_2)\,.
\end{aligned}$$

Proposition 8.2. Let G be constructed from F as above, and let A be an object of $\mathcal{C}$. The following two conditions are equivalent.

(1) There exists an isomorphism $i : F(A, A) \to A$ such that

$$id_A = \mathit{fix}(\lambda e^{\mathcal{C}(A,A)}.\, i \circ F(e, e) \circ i^{-1})\,.$$

(Here *fix* is the least-fixed-point operator.)

(2) There exists a family of morphisms $(\pi_n)_{n\in\omega}$ such that $\overline{A} = (A, (\pi_n)_{n\in\omega})$ is the unique fixed-point of G up to isomorphism.

Proof. (1) implies (2): Define the $\mathcal{C}$-morphisms $\pi_n : A \to A$ by induction on n: $\pi_0 = \bot_{A,A}$ and $\pi_{n+1} = i \circ F(\pi_n, \pi_n) \circ i^{-1}$. We must show that $(A, (\pi_n)_{n\in\omega})$ is an object of $\mathcal{D}$ by verifying the four requirements in the definition of $\mathcal{D}$. The first requirement is immediate by definition, the second and third requirements are easy to show by induction, and the fourth requirement is exactly the assumption that $id_A = \mathit{fix}(\lambda e^{\mathcal{C}(A,A)}.\, i \circ F(e,e) \circ i^{-1})$. Now let $\overline{A} = (A, (\pi_n)_{n\in\omega})$. It remains to show that $G(\overline{A}, \overline{A}) \cong \overline{A}$; uniqueness follows from Theorem 3.1. We claim that the isomorphism $i : F(A,A) \to A$ in $\mathcal{C}$ is also an isomorphism from $G(\overline{A}, \overline{A})$ to $\overline{A}$ in $\mathcal{D}$, i.e., that both i and its inverse i^{-1} in $\mathcal{C}$ are uniform with respect to the families of morphisms $(\pi_n^{A,A})_{n\in\omega}$ and $(\pi_n)_{n\in\omega}$ on $F(A,A)$ and A, respectively. Clearly $\pi_0 \circ i = i \circ \pi_0^{A,A}$ by strictness. Also,

$$\pi_{n+1} \circ i = (i \circ F(\pi_n, \pi_n) \circ i^{-1}) \circ i = i \circ F(\pi_n, \pi_n) = i \circ \pi_{n+1}^{A,A}$$

by the definitions of π_{n+1} and $\pi_{n+1}^{A,A}$. So i is uniform. The proof that i^{-1} is uniform is completely similar. In conclusion, $i : G(\overline{A}, \overline{A}) \to \overline{A}$ is an isomorphism in $\mathcal{D}$.

(2) implies (1): Assume that $i : G(\overline{A}, \overline{A}) \to \overline{A}$ is an isomorphism in $\mathcal{D}$. Then $i : F(A,A) \to A$ is clearly also an isomorphism in $\mathcal{C}$; formally one applies the forgetful functor $\mathcal{D} \to \mathcal{C}$ to i. Since i is uniform, as are all morphisms in $\mathcal{D}$, we have that $\pi_{n+1} \circ i = i \circ \pi_{n+1}^{A,A} = i \circ F(\pi_n, \pi_n)$, and hence that $\pi_{n+1} = i \circ F(\pi_n, \pi_n) \circ i^{-1}$. By the definition of objects of $\mathcal{D}$ we furthermore have that $\pi_0 = \bot_{A,A}$ and that $\sqcup_{n\in\omega} \pi_n = id_A$. But then

$$\mathit{fix}(\lambda e^{\mathcal{C}(A,A)}.\, i \circ F(e,e) \circ i^{-1}) = \bigsqcup_{n\in\omega} \pi_n = id_A\,. \qquad \square$$

It remains to discuss how completeness properties of $\mathcal{C}$ transfer to $\mathcal{D}$. One can show, using the O-category variant of Lemma 3.2 [22], that the forgetful functor from $\mathcal{D}$ to $\mathcal{C}$ creates terminal objects and limits of ω^{op}-chains of split epis. Alternatively, by imposing an additional requirement on $\mathcal{C}$ one can show that the forgetful functor creates *all* limits: for a given limit in $\mathcal{C}$, the induced bijection between cones and mediating morphisms must be an isomorphism in the category of cpos (where cones are ordered pointwise, using the order on each hom-set). That requirement is in particular satisfied by the usual concrete categories of cpos.

9 Discussion and related work

We now discuss the level of generality of our results and clarify what the contribution is compared to other results in the literature.

Metric spaces vs. ultrametric spaces. In the definition of an M-category we require that hom-sets are *ultrametric* spaces and not merely general metric spaces. In most of the related work on metric-space equations that we cite there is no such restriction; only Rutten [18] restricts to ultrametric spaces.

There are two reasons that we restrict to ultrametric spaces. The first is perhaps mostly a matter of taste: the category of complete, general metric spaces is not cartesian closed, only symmetric monoidal closed [15]. This complicates the inspiration one can draw from enriched category theory. For example, van Breugel and Warmerdam [10] consider a product of *CMS*-categories which is given by the cartesian structure, and not the canonical monoidal structure, of the category *CMS*

of complete metric spaces; one has to show 'by hand' that this gives a well-defined *CMS*-category.

The main reason, however, that we restrict to ultrametric spaces is that doing so seems to be sufficient for applications in semantics. Turi and Rutten [24, p.529] observe:

> In semantics, one usually works with these more specific structures [ultrametric spaces], but it is convenient to state the general theorems in the full generality of metric spaces.

We agree that ultrametric spaces suffice, but unlike Turi and Rutten we found it convenient *not* to state the results for general metric spaces. The simple reason is that some definitions and proofs would become more complicated. More specifically, the definition of an increasing Cauchy tower below would need to be modified in the style of the definition of a 'converging tower' in America and Rutten [6]. With that change, we expect that at least the results of Sections 3 and 4 would carry through also for general metric spaces.

Actually, one could specialize further. All the ultrametric spaces we consider in the applications in Section 7 have an even more special form: all non-zero distances in those spaces are of the form 2^{-n} for some natural number $n \geq 0$. In practice, this means that when working with those applications, and in particular in calculations, one considers the metric-space structure as given by a family of equivalence relations $=_n$ where $x =_n y$ holds if and only if $d(x, y) \leq 2^{-n}$. One could perhaps specialize even further, by requiring that these relations $=_n$ are obtained from 'projection' functions [7, 12]. We did not restrict to any such structures, but found ultrametric spaces to give simple proofs at a reasonable level of generality.

Generalized ultrametric spaces. Apart from the distinction between general metric spaces and ultrametric spaces, there are other choices one could consider in the definition of M-categories. First, it is likely that the restriction to *non-empty* metric spaces could be removed, but that this would require unpleasant special cases in the uniqueness and existence theorems.

Generalizing in a different direction than metric spaces, one could consider categories where the hom-sets are *generalized* ultrametric spaces [18]. In these spaces, only the ultrametric inequality and the axiom $d(x, x) = 0$ are required: the distance function need not be symmetric, and distinct elements can have distance 0. These spaces are particularly nice from an (enriched) category theoretic viewpoint, since they are exactly categories enriched over the preorder category $[0, 1]^{\mathrm{op}}$. On the other hand, limits in such spaces are somewhat more complicated than in ordinary metric spaces, and some basic intuitions need to be revisited: for example, non-expansive functions are not necessarily continuous.

A remarkable aspect of generalized ultrametric spaces is that they generalize both metric spaces and preorders. We find this particularly interesting in light of the comparison between M-categories and O-categories in Section 8: one could imagine that categories enriched over (complete) generalized ultrametric spaces would allow for a result about solving recursive equations that generalizes both our work and the classical result of Smyth and Plotkin. We do not know whether there exists such a common generalization and leave it as a direction for future work.

Finally, one could perhaps go even further and generalize from the preorder category $[0, 1]^{\mathrm{op}}$ to arbitrary quantales. Here the goal would be to prove results that relate to the work of Wagner [25] in the same way that Smyth and Plotkin's work on O-categories relates to Scott's classical inverse limit construction. All of this is merely speculation, however, and quite far from the more application-oriented goals of this paper.

Contributions compared to related work. As mentioned in the introduction, the idea of considering categories with metric spaces as hom-sets has been used in earlier work [10, 19]. We now turn to a more detailed description of what is new in our results.

First of all, while the results in Sections 4 (on locally compact subcategories) and 8 (on relating O-categories to M-categories) are generalizations of earlier work, we find these generalizations sufficiently far-reaching and non-trivial to be considered new in their own right.

As for the main results in Section 3 about solutions to recursive equations, the contribution with respect to previous work consists in Lemma 3.2, Proposition 3.3, and Theorem 3.4. First, Lemma 3.2 (and its immediate consequence, Proposition 3.3) shows a limit-colimit coincidence result for general M-categories. Theorem 4.15 of Rutten and Turi [19] gives some, but not all of this result. That theorem only refers to cones in a category $\mathcal{C}^E$ of embedding-projection pairs[4], and does not relate these cones to limiting cones or colimiting cocones in $\mathcal{C}$. In particular, the theorem does not show that an inverse limit in $\mathcal{C}$ can be turned into a cone in a category of embedding-projection pairs. This argument, which is an essential part of the construction of fixed points of functors, is later given in the concrete case of a category of complete metric spaces only (Theorem 4.23).

Second, Theorem 3.4 is new: we do not know of any other existence theorem about fixed points of functors for a general class of 'metric-enriched' categories. A small contribution here is the identification of suitable completeness conditions on M-categories, together with examples that illustrate their use. In particular, the condition of having all inverse limits of split epis is needed for the prime example of an M-category, namely $\mathsf{CBUlt}_{\mathrm{ne}}$ itself. This condition seems to be of general use given that the hom-sets of an M-category are required to be non-empty.

Still, it is fair to say that (as one reviewer put it) the ingredients of the existence theorem already occur in various places in the literature. Perhaps a more proper way to evaluate our main result is to consider how the applications in Section 7 would be treated using these previous results, notably Theorem 4.25 of Rutten and Turi [19]. Although the recursive equations in the applications can be solved in this way, we think that the route requires sufficient extra work that it useful to state a general existence theorem that directly applies to mixed-variance functors on arbitrary M-categories. Most importantly, in the absence of our Lemma 3.2 it is, as noted above, necessary to argue 'by hand' that inverse limits of ω^{op}-chains in $\mathsf{PreCBUlt}_{\mathrm{ne}}$ can be turned into cones in a category of embedding-projection pairs.

10 Conclusion

We have generalized the standard solution of recursive equations over complete ultrametric spaces [6] to the abstract setting of M-categories where, in the style of Smyth and Plotkin [22], the focus is on the metric structure on the morphisms rather than the objects. We have furthermore outlined an alternative existence theorem which is, at least informally, a closer categorical analogy to Banach's fixed-point theorem.

We have given a general account of 'compact' variants of such categories, showing that these subcategories always inherit solutions of recursive equations from the full categories. As another application we have presented a construction that provides a correspondence between solutions of generalized domain equations in O-categories with solutions of equations in M-categories.

In addition, we have sketched a number of applications from denotational semantics. In particular, the application in Section 7.2 requires a solution to a recursive

[4] We assume that the second $\mathcal{C}$ in the statement of that theorem should have been $\mathcal{C}^E$.

equation over metric spaces with additional structure; our results provide such a solution.

Acknowledgments We thank the anonymous referees for helpful comments and suggestions. This research was partially funded by the grant 'Modular Reasoning about Software' from The Danish Council for Independent Research, Natural Sciences.

References

[1] M. Abadi and G. D. Plotkin. A per model of polymorphism and recursive types. In J. Mitchell, editor, *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 355–365, Philadelphia, Pennsylvania, June 1990. IEEE Computer Society Press.

[2] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In Z. Shao and B. C. Pierce, editors, *Proceedings of the Thirty-Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 340–353, Savannah, Georgia, Jan. 2009. ACM Press.

[3] F. Alessi, P. Baldan, and G. Bellè. A fixed-point theorem in a category of compact metric spaces. *Theoretical Computer Science*, 146(1&2):311–320, 1995.

[4] F. Alessi, P. Baldan, G. Bellè, and J. J. M. M. Rutten. Solutions of functorial and non-functorial metric domain equations. In S. Brookes, M. Main, A. Melton, and M. Mislove, editors, *Proceedings of the 11th Conference on the Mathematical Foundations of Programming Semantics*, volume 1 of *Electronic Notes in Theoretical Computer Science*, pages 1–12. Elsevier, 1995.

[5] R. M. Amadio. Recursion over realizability structures. *Information and Computation*, 91(1):55–85, 1991.

[6] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39(3):343–375, 1989.

[7] C. Baier and M. E. Majster-Cederbaum. The connection between initial and unique solutions of domain equations in the partial order and metric approach. *Formal Aspects of Computing*, 9(4):425–445, 1997.

[8] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In P. Urzyczyn, editor, *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005*, number 3461 in Lecture Notes in Computer Science, pages 86–101, Nara, Japan, Apr. 2005. Springer.

[9] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In L. de Alfaro, editor, *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009*, number 5504 in Lecture Notes in Computer Science, pages 456–470, York, United Kingdom, Mar. 2009. Springer.

[10] F. van Breugel and J. Warmerdam. Solving domain equations in a category of compact metric spaces. Report CS-R9424, CWI, Amsterdam, 1994.

[11] J. W. de Bakker and J. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[12] F.-J. de Vries. Projection spaces and recursive domain equations. *Information Processing Society of Japan SIG Notes*, 95(114):37–38, 1995. Extended abstract. Retrieved from `http://ci.nii.ac.jp/Detail/detail.do?LOCALID=ART0003278737&lang=en`.

[13] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In M. V. Hermenegildo and J. Palsberg, editors, *Proceedings of the Thirty-Seventh Annual ACM Symposium on Principles of Programming Languages*, pages 185–198, Madrid, Spain, 2010. ACM Press.

[14] A. Gotsman, J. Berdine, B. Cook, N. Rinetzky, and M. Sagiv. Local reasoning for storable locks and threads. In Z. Shao, editor, *Programming Languages and Systems, 5th Asian Symposium, APLAS 2007*, number 4807 in Lecture Notes in Computer Science, pages 19–37, Singapore, Nov. 2007. Springer.

[15] F. W. Lawvere. Metric spaces, generalized logic, and closed categories. In *Rendiconti del Seminario Matematico e Fisico di Milano, XLIII*. Tipografia Fusi, Pavia, 1973.

[16] S. Mac Lane. *Categories for the Working Mathematician, Second Edition*, volume 5 of *Graduate Texts in Mathematics*. Springer, 1998.

[17] A. M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.

[18] J. J. M. M. Rutten. Elements of generalized ultrametric domain theory. *Theoretical Computer Science*, 170(1-2):349–381, 1996.

[19] J. J. M. M. Rutten and D. Turi. On the foundation of final semantics: Non-standard sets, metric spaces, partial orders. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop on Sematics: Foundations and Applications*, number 666 in Lecture Notes in Computer Science, pages 477–530, Beekbergen, The Netherlands, June 1992. Springer.

[20] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In E. Grädel and R. Kahle, editors, *Proceedings of the 18th EACSL Annual Conference on Computer Science Logic*, volume 5771 of *Lecture Notes in Computer Science*, pages 440–454, Coimbra, Portugal, Sept. 2009. Springer.

[21] M. B. Smyth. Topology. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[22] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11(4):761–783, 1982.

[23] P. Taylor. Homomorphisms, bilimits and saturated domains, 1987. Manuscript. Available from `http://www.paultaylor.eu/domains/`.

[24] D. Turi and J. J. M. M. Rutten. On the foundations of final coalgebra semantics. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998.

[25] K. R. Wagner. *Solving Recursive Domain Equations with Enriched Categories*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1994.

Chapter 5

A Relational Realizability Model for Higher-Order Stateful ADTs

A Relational Realizability Model for Higher-Order Stateful ADTs

Lars Birkedal, Kristian Støvring, Jacob Thamsborg*

IT University of Copenhagen

Abstract

We present a realizability model for reasoning about contextual equivalence of higher-order programs with impredicative polymorphism, recursive types, and higher-order mutable state.

The model combines the virtues of two recent earlier models: (1) Ahmed, Dreyer, and Rossberg's step-indexed logical relations model, which was designed to facilitate proofs of representation independence for "state-dependent" ADTs and (2) Birkedal, Støvring, and Thamsborg's realizability logical relations model, which was designed to facilitate abstract proofs without tedious step-index arithmetic. The resulting model can be used to give abstract proofs of representation independence for "state-dependent" ADTs.

Keywords: Abstract Data Types, Logical Relations, Local State, Parametricity

1. Introduction

The method of *logical relations* has proved to be a useful technique for reasoning about the equivalence of higher-order programs that use different internal data representations to implement the same functionality. Since Reynold's seminal work on using logical relations for reasoning about relational parametricity for pure System F, there has been a long series of work

*Corresponding Author

Email addresses: birkedal@itu.dk (Lars Birkedal), kss@itu.dk (Kristian Støvring), thamsborg@itu.dk (Jacob Thamsborg)

URL: www.itu.dk/~birkedal (Lars Birkedal), www.itu.dk/~kss (Kristian Støvring), www.itu.dk/~thamsborg (Jacob Thamsborg)

[1]Rued Langgaards Vej 7, DK-2300 Copenhagen S

on generalizing and extending the technique to reason about increasingly realistic programming languages [26, 28, 23, 10, 4, 17].

For programming languages involving recursive types and general references there are two main technical challenges:

Well-definedness Show that the logical relation is well-defined (that is exists); traditionally logical relations have been defined by induction on the structure of types but that is not possible in the presence of recursive types (and/or references).

Mutable Abstract Data Types Define the logical relation in such a way that one can use it to show equivalences of programs using local state for implementing mutable abstract data types in different ways

Recently (in 2009) two logical relation models, developed in parallel, were proposed for reasoning about a call-by-value language with impredicative polymorphism, recursive types, and general references: one was developed by Ahmed, Dreyer, and Rossberg (hereafter ADR) [3] and one was developed by the current authors (hereafter BST) [12]. Both models use *Kripke* logical relations to capture that the meaning of types depends on how many references have been allocated

We now highlight some features of the ADR and BST models to situate the present paper.

The ADR model is a step-indexed model over the operational semantics in which the logical relation is indexed by natural numbers, following ideas of Appel and McAllester [8]. Step-indexing is used to address the challenge of showing well-definedness of the logical relation. The main technical innovation in the ADR model is an advanced definition of worlds, which makes it possible to show contextual equivalences of many examples involving local state. In particular, it is possible to reason about programs using local state invariants that *evolve* over time.

However, because of the use of step-indexing it is quite painful to reason directly using the model because one is forced to engage in tedious step-index arithmetic to derive even simple results. For example, to show that a function f_1 contextually approximates another function f_2 it does not suffice to show that they map related arguments to related results. Instead, one must show the stronger condition that, for any $n \in \mathbb{N}$, if v_1 and v_2 are logically related for n steps, then $f_1(v_1)$ and $f_2(v_2)$ are logically related for n steps as well. The step-stratified possible worlds that arise in a model like

ADR only make matters worse by requiring additional quantification over future worlds throughout the proof.

This led Dreyer et al. to develop logics for reasoning more abstractly about step-indexed logical relation models, first for a language without references [20] and then, most recently, for reasoning about the ADR model [21]. The latter logic, called LADR, is a modal relational logic in which one can reason about ADR style contextual equivalences at a higher level of abstraction avoiding low-level details about steps and worlds.

The focus of the BST model was to obtain a relatively abstract logical relations model, without any step-indexing, by constructing the logical relations over a simple adequate domain-theoretic model of the programming language. Thence the well-definedness of the model was more complicated to establish and the main technical innovations in the BST model were (i) the observation that one can solve the naturally occurring recursive world equation in a category of ultrametric spaces and (ii) a novel modeling of locations with a domain-theoretic codification of approximation information, crucially used for establishing the well-definedness of the model. The model is indeed more abstract than the ADR model in the sense that, e.g., two functions f_1 and f_2 are related if they map related arguments to related results and there is no reasoning about steps. On the other hand, the BST model used a simple form of world, which only allowed to prove equivalences of programs that used local state in simple ways.

In this paper we extend the BST model with more refined worlds similar to those from the ADR model (specifically, we use the world description of LADR, which is a slight simplification of the one in ADR). Thus we show that the semantic techniques used in the BST model scale to state-of-the-art world descriptions and the resulting model can be used to show equivalences like those that can be shown using the ADR model, but with more abstract reasoning without any step-indexing. We compare reasoning in the resulting model to reasoning using the ADR model and the LADR logic.

2. Overview of the Technical Development

The present paper is a lengthy and somewhat technical one. To navigate safely the many details, we provide a quick, informal overview of the development and give extended textual explanations of some high points.

The language in question, including typing rules, is introduced in Section 3. It is a quite standard call-by-value language with universal, recursive and reference types.

An untyped denotational semantics is given in Section 4. The semantics is adequate and is given in monadic style by means of a universal predomain; this again, is obtained as the solution to a recursive domain equation. The semantics is quite standard with the exception of approximate locations, see Subsection 2.1 below. This section also defines the crucial domain-theoretic notions of uniform predomains and domains.

Some basic metric space theory is recalled in Section 5, in particular we discuss the notion of ultrametrics. Also we introduce a category of certain ordered metric spaces with an associated fixed-point theorem to be used in Section 7.

Bohr relations on uniform predomains and domains are defined and also equipped with a metric in Section 6. These are the kind of relations on states and values we will work with; the definition is motivated in Subsection 2.2 below.

The possible worlds of our Kripke logical relations are built in Section 7. These mimic the worlds of ADR. They are obtained as the fixed point of a functor on a certain category of ordered metric spaces; we laboriously build this functor and verify that it meets the requirements of the fixed point theorem. See also Subsection 2.3 below for a short, informal description of the worlds and some considerations on the choice of categories and fixed-point theorem.

The world-indexed logical relation is finally built in Section 8. The relation on states induced by a world corresponds to the approach taken in ADR, the interpretation of reference types does not, rather we take a more extensional approach. The remaining types are interpreted much as in BST, in particular we rely on our metric setup and Banach's fixed-point theorem in the case of recursive types. Also we rely on the approximate locations discussed in Subsection 2.1 below to ensure that the interpretation of reference types is well-defined.

The fundamental theorem of logical relations and proof resides in Section 9 after a definition of semantic relatedness; that the latter implies contextual approximation is an immediate corollary. The proof is lengthy, but it is a simple matter of verification in light of the definitions of the previous sections, and we only include some of the proof cases.

A worked-out example is the last Section 10 of the paper. We introduce some necessary syntactic sugar and prove the equivalence of Example 5.1 in ADR. This particular example is spelled out in ADR too, and so one can compare reasoning in the two models. Indeed, we conclude this section with some general considerations on this, also taking into account the recent LADR logic [21]. This serves as conclusion to the entire paper as well and has directions for future work.

2.1. Approximate Locations

As mentioned in the introduction, it is not, in general, trivial to prove the existence of logical relations in the presence of recursive types; a simple definition by induction on the types will not do. Minimal invariance as proposed by Pitts [27] and others is, arguably, the method of choice to tackle this issue, but it is not readily applicable because of the general reference types. In some sense, the standard, flat modelling of locations as integers does not provide enough foothold to get the iterative machinery of minimal invariance going.

Faced with this issue, the authors coined the idea of *approximate* locations in earlier work [13]. A location, say l, is modelled by an element λ_l that is the least upper bound of an ascending chain $\lambda_l^1 \sqsubseteq \lambda_l^2 \sqsubseteq \cdots \sqsubseteq \lambda_l$ of so-called approximate locations. The interpretation of references to a type ν then has 'proper' semantic locations such as λ_l as well as approximate semantic locations such as λ_k^{n+1}; the latter intuitively signifies that ν and the type of values stored at location k might agree only up to the nth approximation.

This idea was rewrapped and reused in BST; we copy that usage here, apart from a minor technical change to the interpretation of lookup and assignment due to the more refined worlds. See Section 4 for details.

The approximated locations are required for technical reasons; the interpretation of reference types simply would not be non-expansive without them. On the other hand, they do not mirror anything in the language and are, as such, junk. Some implications of their presence in the model is discussed at the end of Section 10.

2.2. Bohr Relations

One novelty of this paper is the particular choice of conditions we impose on our relations.

We carve our relations out of a universal predomain V that loosely corresponds to the set of closed, syntactic values. V is essentially obtained as the solution to a recursive domain equation as prescribed by Smyth and Plotkin [32]. But we must impose some restrictions – it will not do to allow all relations on V. The presence of recursive terms requires that relations respect the denotational construction of fixed points. And recursive types renders the existence of the logical relation non-trivial and the relations must accommodate that.

In BST we worked with *complete*, *uniform* relations. Complete means chain-complete, i.e., if we have an ascending chain of pairs in a relation, then the pair of the least upper bounds also must be in the relation. Uniform loosely means closed under the projections that come with solutions to recursive domain equations. For each $n \in \omega$ we have a projection $\pi_n : V \to V_\perp$; a relation $R \subseteq V \times V$ is uniform if for all $(v_1, v_2) \in R$ and all $n \in \omega$ we have that

$$(\pi_n(v_1), \pi_n(v_2)) \in \{\perp, \perp\} \cup \{\lfloor w_1 \rfloor, \lfloor w_2 \rfloor \mid (w_1, w_2) \in R\},$$

where $\lfloor - \rfloor : V \to V_\perp$ is the standard inclusion. Completeness and uniformity deal with the issues that arise from recursive terms and types respectively. Indeed, they are both well-known approaches, completeness is present, e.g., in work by Reynolds [30] and uniformity is found, e.g., in work by Abadi and Plotkin [2] and by Amadio [5].

Restricting to uniform and complete relations comes at a price, however: we are, e.g., unable to relate an integer to a pair of integers since the latter but not the former 'bottom out' under application of $\pi_1 : V \to V_\perp$. This is a shortcoming because the conceptual relations that one 'plugs into' universal types must be complete and uniform too, which limits the use of relational parametricity. Note that the restriction to uniform and complete relations does have some intuitive merit; we do, after all, approximate contextual equivalence with our relations and thus relating bottom to non-bottom seems inappropriate.

It is this shortcoming we address with *Bohr relations*, which we formally introduce in Section 6. Conceptually, we aim for relations that approximate contextual approximation rather than contextual equivalence. Technically, Bohr relations only restrict the left hand side: Bohr relations are chain-

complete and downwards closed in the left coordinate. The former means, that if we have a sequence of pairs in the relation such that the left coordinates form an ascending chain and the right coordinates are identical, then the pair of the least upper bound of the left coordinates and the right coordinate must be in the relation too. The latter means, that if we have a pair in the relation, then any pair with a smaller left coordinate and identical right coordinate must be in the relation too. Being uniform instead of downwards closed in the left coordinate would work as well, but we stick to the latter for simplicity.

While not all relations on V are Bohr relations, they all have a least Bohr relation that contain them; this closure can be 'plugged into' universal types. Thus the overall idea is to remove the artificial 'synchronization' restriction imposed by (two-sided) uniformity and so be free to apply relational parametric reasoning at will.

Going for contextual approximation instead of contextual equivalence seems standard in recent step-indexed models of recursive types. There is an analogy to Bohr relations here: step-indexed models, e.g. [4], do not require expressions to terminate in the same number of steps in order for them to be related. Rather they allot a number of steps for the left hand side to terminate, and if this happens then the right hand side is required to terminate in any number of steps. Requiring the expressions to march in step would, most likely, not invalidate the soundness of the reasoning but rather prove fewer (albeit stronger) equivalences.

For expository reasons, we have focused on relations between values and reasoning by relational parametricity in the explanations above. It is worthwhile, however, to note, that the restrictions on relations apply to relations between states too. In particular, we would have been unable to relate, say, the empty state to any non-empty state with the (two-sided) uniformity requirement of BST. This posed no problem in BST because of the simple notion of worlds, but it would have been a severe limitation here.

We finally remark that the idea of approximating contextual approximation rather than contextual equivalence is present in the 4-tuples of Bohr and Birkedal [17], hence the nomenclature. Their setup handled any kind of relation, whether complete or not, uniform or not. We think we have distilled this ability: 4-tuples are – roughly and in retrospect – just two Bohr relations grouped together to be able to argue both ways of contextual approximation in one go.

2.3. Solving Recursive World Equations

Definitional circularities arise when modelling higher order store phenomena; the main accomplishment of BST is the use of metric space theory to solve one such circularity. That particular circularity involves both the space of types and the space of worlds and so one has the choice of solving for either. This is not, however, an immaterial choice. Types come with no particular order and we can make do with a classic fixed-point result for functors on the category of ultrametric spaces by America and Rutten [6]; this was the approach taken in BST. Worlds, on the other hand, come with an extension ordering that corresponds to further allocation. Hence we arrive at a functor on certain ordered metric spaces and the cited result no longer suffices.

In ADR, the notion of world is far more refined than in BST. A world is a series of islands, each managing separate parts of the store. Islands themselves are dynamic, they have a population that may grow according to a population law. Also each island has a heap law that regulates the part of the store managed by the island; the heap law is indexed by the population and hence may vary over time. We refer the reader to Section 7 and in particular to ADR for further motivation and explanation; here it shall suffice to state that the heap laws are indexed also over worlds themselves and so the definition of worlds is circular. But unlike the circularity solved in BST, there seems to be no way of 'cycling' this circularity to arrive at point where the fixed-point result of America and Rutten is applicable.

Faced with this challenge, the authors proved a generalized fixed-point theorem [15] that allows for additional structure on the metric spaces, in particular certain orderings. And it is a special case of this theorem that we shall apply in Section 7 to build our space of worlds.

Recently, Dreyer et al. have developed the logic LADR [21] to facilitate reasoning in the ADR model. In the process, they simplified the ADR model somewhat and it is the notion of worlds from this, simpler model, that we have chosen to settle on in this paper. We believe that this simplification has removed the obstacles that prevented the use of the fixed-point result of America and Rutten in our adaptation of the original ADR model. In other words, we probably could do without the aforementioned generalized fixed-point result. It would, however, take some amount of 'hacking' to do so and the development would be more complicated.

3. Programming Language

We consider the same programming language as the one used in the BST model [12]. It is a standard call-by-value language with universal types, iso-recursive types, ML-style reference types, and a ground type of integers.

The language is sketched in Figures 1 and 2. The typing rules are standard [25]. In the figure, Ξ and Γ range over contexts of type variables and term variables, respectively. As we do not consider operational semantics in this article, there is no need for location constants, and hence no need for store typings.

4. Untyped semantics

The terms of the language above are not intrinsically typed. In other words, the language consists of an untyped term language and a set of rules for assigning types to untyped terms. We now take advantage of this distinction and give a semantics of the untyped term language. This "untyped semantics" is almost identical to the one used in the BST model [12, 16] (we point out some minor differences below), but we include a description here in order to keep the article self-contained.

As usual for models of untyped languages, the semantics is given by means of a "universal" complete partial order (cpo) in which one can inject integers, pairs, functions, etc. This universal cpo is obtained by solving a recursive domain equation.

The only non-standard aspect of the semantics is the treatment of store locations. As explained in Section 2.1, the model includes *approximate* locations. This means that locations are modeled as elements of the cpo $Loc = \mathbb{N} \times \overline{\omega}$ where $\overline{\omega}$ is the "vertical natural numbers" cpo: $1 \sqsubset 2 \sqsubset \cdots \sqsubset n \sqsubset \cdots \sqsubset \infty$. (For notational reasons it is convenient to call the least element 1 rather than 0.) The intuitive idea is that locations can be approximated: the element $(l, \infty) \in Loc$ is the "ideal" location numbered l, while the elements of the form (l, n) for $n < \infty$ are its approximations. As already mentioned, these approximate locations are included in order to ensure that the logical relation we construct is well-defined.

4.1. Domain-theoretic preliminaries

We assume that the reader is familiar with basic denotational semantics, as presented for example in Winskel [34], and with semantics in monadic style [24].

Types: $\tau ::= \mathsf{int} \mid 1 \mid \tau_1 \times \tau_2 \mid 0 \mid \tau_1 + \tau_2 \mid \mu\alpha.\tau \mid \forall\alpha.\tau \mid \alpha \mid \tau_1 \to \tau_2 \mid \mathsf{ref}\,\tau$

Terms: $t ::= x \mid \overline{m} \mid \mathsf{ifz}\,t_0\,t_1\,t_2 \mid t_1 + t_2 \mid t_1 - t_2 \mid () \mid (t_1,\, t_2) \mid \mathsf{fst}\,t \mid \mathsf{snd}\,t$
$\mid \mathsf{void}\,t \mid \mathsf{inl}\,t \mid \mathsf{inr}\,t \mid \mathsf{case}\,t_0\,x_1.t_1\,x_2.t_2 \mid \mathsf{fold}\,t \mid \mathsf{unfold}\,t$
$\mid \Lambda\alpha.t \mid t\,[\tau] \mid \lambda x.t \mid t_1\,t_2 \mid \mathsf{fix}\,f.\lambda x.t \mid \mathsf{ref}\,t \mid !t \mid t_1 := t_2$

Typing rules:

$$\frac{}{\Xi \mid \Gamma \vdash x : \tau}\ (\Xi \vdash \Gamma,\, \Gamma(x) = \tau) \qquad \frac{}{\Xi \mid \Gamma \vdash \overline{m} : \mathsf{int}}\ (\Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Gamma \vdash t_0 : \mathsf{int} \qquad \Xi \mid \Gamma \vdash t_1 : \tau \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash \mathsf{ifz}\,t_0\,t_1\,t_2 : \tau}$$

$$\frac{\Xi \mid \Gamma \vdash t_1 : \mathsf{int} \qquad \Xi \mid \Gamma \vdash t_2 : \mathsf{int}}{\Xi \mid \Gamma \vdash t_1 \pm t_2 : \mathsf{int}} \qquad \frac{}{\Xi \mid \Gamma \vdash () : 1}\ (\Xi \vdash \Gamma)$$

$$\frac{\Xi \mid \Gamma \vdash t_1 : \tau_1 \qquad \Xi \mid \Gamma \vdash t_2 : \tau_2}{\Xi \mid \Gamma \vdash (t_1,\, t_2) : \tau_1 \times \tau_2} \qquad \frac{\Xi \mid \Gamma \vdash t : 0}{\Xi \mid \Gamma \vdash \mathsf{void}\,t : \tau}\ (\Xi \vdash \tau)$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\,t : \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{snd}\,t : \tau_2}$$

$$\frac{\Xi \mid \Gamma \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{inl}\,t : \tau_1 + \tau_2}\ (\Xi \vdash \tau_2) \qquad \frac{\Xi \mid \Gamma \vdash t : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{inr}\,t : \tau_1 + \tau_2}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Gamma \vdash t_0 : \tau_1 + \tau_2 \qquad \Xi \mid \Gamma, x_i : \tau_i \vdash t_i : \tau \quad (i = 1, 2)}{\Xi \mid \Gamma \vdash \mathsf{case}\,t_0\,x_1.t_1\,x_2.t_2 : \tau}$$

(Continued in Figure 2.)

Figure 1: Programming language

$$\frac{\Xi \mid \Gamma \vdash t : \tau[\mu\alpha.\tau/\alpha]}{\Xi \mid \Gamma \vdash \mathsf{fold}\, t : \mu\alpha.\tau} \qquad \frac{\Xi \mid \Gamma \vdash t : \mu\alpha.\tau}{\Xi \mid \Gamma \vdash \mathsf{unfold}\, t : \tau[\mu\alpha.\tau/\alpha]}$$

$$\frac{\Xi, \alpha \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \Lambda\alpha.t : \forall\alpha.\tau}\ (\Xi \vdash \Gamma) \qquad \frac{\Xi \mid \Gamma \vdash t : \forall\alpha.\tau_0}{\Xi \mid \Gamma \vdash t\,[\tau_1] : \tau_0[\tau_1/\alpha]}\ (\Xi \vdash \tau_1)$$

$$\frac{\Xi \mid \Gamma, x : \tau_0 \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \lambda x.t : \tau_0 \to \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t_1 : \tau \to \tau' \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash t_1\, t_2 : \tau'}$$

$$\frac{\Xi \mid \Gamma, f : \tau_0 \to \tau_1, x : \tau_0 \vdash t : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{fix}\, f.\lambda x.t : \tau_0 \to \tau_1} \qquad \frac{\Xi \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \mathsf{ref}\, t : \mathsf{ref}\, \tau}$$

$$\frac{\Xi \mid \Gamma \vdash t : \mathsf{ref}\, \tau}{\Xi \mid \Gamma \vdash !t : \tau} \qquad \frac{\Xi \mid \Gamma \vdash t_1 : \mathsf{ref}\, \tau \qquad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash t_1 := t_2 : 1}$$

Figure 2: Programming language (ctd.)

Let Cpo be the category of ω-cpos and ω-continuous functions. We use the standard notation for products, sums, and function spaces in Cpo. Injections into binary sums are written ι_1 and ι_2. For any set M and any cpo A, the cpo $M \rightharpoonup_{\mathit{fin}} A$ has maps from finite subsets of M to A as elements, and is ordered as follows: $f \sqsubseteq f'$ if and only if f and f' has the same domain M_0 and $f(m) \sqsubseteq f'(m)$ for all $m \in M_0$.

A complete, pointed partial order (cppo) is a cpo containing a least element. We use the notation $A_\perp = \{\lfloor a \rfloor \mid a \in A\} \cup \{\perp\}$ for the cppo obtained by "lifting" a cpo A. The least fixed-point of a continuous function $f : D \to D$ from a cppo D to itself is written $\mathit{fix}\, f$. The cppo of strict, continuous functions from a cpo A to a cppo D is written $A \multimap D$. For continuous functions $f : A \to B_\perp$ and $g : B \to C_\perp$ we define $g \mathbin{\overline{\circ}} f : A \to C_\perp$ as follows:

$$f \mathbin{\overline{\circ}} g = \lambda a. \begin{cases} f\, b, & \text{if } g\, a = \lfloor b \rfloor, \\ \perp, & \text{otherwise.} \end{cases}$$

Having now specified the kinds of partial orders we use, we follow common practice and introduce some more abstract terminology: in this article, a *predomain* simply means a cpo, and a *domain* means a cppo.

The semantics below is presented in monadic style [24], i.e., structured using a monad that models the effects of the language. It is most convenient to define this monad by means of a Kleisli triple: for every predomain S and

every domain Ans, the continuation-and-state monad $T_{S,Ans} : \mathsf{Cpo} \to \mathsf{Cpo}$ over S and Ans is given by

$$\begin{aligned} T_{S,Ans}\, A &= (A \to S \to Ans) \to S \to Ans \\ \eta_A\, a &= \lambda k.\lambda s.\, k\, a\, s \\ c \;\star_{A,B}\; f &= \lambda k.\lambda s.\, c\, (\lambda a.\lambda s'.\, f\, a\, k\, s')\, s \end{aligned}$$

where $\eta_A : A \to T_{S,Ans}A$ and $\star_{A,B} : T_{S,Ans}A \to (A \to T_{S,Ans}B) \to T_{S,Ans}B$. In the following we omit the type subscripts on η and $\star$. (Continuations are included for a technical reason, namely to ensure chain-completeness of the relations that will be used to model computations.)

4.2. A universal uniform predomain

The standard methods for solving recursive domain equations give solutions that satisfy certain induction principles [32, 27]. One way of formulating this property is that one obtains as a solution not only a domain D, but also a sequence of "projection" functions ϖ_n on D such that each element d of D is the limit of its projections $\varpi_0(d)$, $\varpi_1(d)$, etc. These functions therefore provide a handle for proving properties about D by induction on n.

Definition 4.1.

1. *A* uniform predomain $(A, (\varpi_n)_{n\in\omega})$ *is a predomain A together with a family $(\varpi_n)_{n\in\omega}$ of continuous functions from A to $A_\perp$, satisfying*

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \cdots \sqsubseteq \varpi_n \sqsubseteq \ldots \tag{1}$$

$$\bigsqcup_{n\in\omega} \varpi_n = \lambda a.\lfloor a \rfloor \tag{2}$$

$$\varpi_m \mathbin{\overline{\circ}} \varpi_n = \varpi_n \mathbin{\overline{\circ}} \varpi_m = \varpi_{\min(m,n)} \tag{3}$$

$$\varpi_0 = \lambda e.\, \bot\,. \tag{4}$$

2. *A* uniform domain $(D, (\varpi_n)_{n\in\omega})$ *is a domain D together with a family $(\varpi_n)_{n\in\omega}$ of strict, continuous functions from D to itself, satisfying*

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \cdots \sqsubseteq \varpi_n \sqsubseteq \ldots \tag{5}$$

$$\bigsqcup_{n\in\omega} \varpi_n = id_D \tag{6}$$

$$\varpi_m \circ \varpi_n = \varpi_n \circ \varpi_m = \varpi_{\min(m,n)} \tag{7}$$

$$\varpi_0 = \lambda e.\, \bot\,. \tag{8}$$

Uniform domains are called *rank-ordered cpos* in earlier work by Baier and Majster-Cederbaum [9].

Proposition 4.2. *There exists a uniform predomain $(V, (\pi_n)_{n\in\omega})$ satisfying the following two properties:*

1. *The following isomorphism holds in* Cpo*:*

$$V \;\cong\; \mathbb{Z} + Loc + 1 + (V \times V) + (V + V) + V \\ + T_{S,Ans}V + (V \to T_{S,Ans}V) \quad (9)$$

   where

$$T_{S,Ans}V = (V \to S \to Ans) \to S \to Ans \\ S = \mathbb{N} \rightharpoonup_{fin} V \\ Ans = (\mathbb{Z} + Err)_\perp$$

   and

$$Loc = \mathbb{N}_0 \times \overline{\omega} \\ Err = 1\,.$$

2. *Abbreviate $TV = T_{S,Ans}V$ and $K = V \to S \to Ans$. Define the following injection functions corresponding to the summands on the right-hand side of the isomorphism* (9)*:*

$$in_{\mathbb{Z}} : \mathbb{Z} \to V \qquad in_{+} : V + V \to V \\ in_{Loc} : Loc \to V \qquad in_{\to} : (V \to TV) \to V \\ in_{1} : 1 \to V \qquad in_{\mu} : V \to V \\ in_{\times} : V \times V \to V \qquad in_{\forall} : TV \to V$$

   With that notation, the functions $\pi_n : V \to V_\perp$ satisfy (and are determined by) the equations shown in Figure 3.

These two properties determine V uniquely, up to isomorphism in Cpo*.*

Proof (sketch). Proposition 3.2 of Birkedal et al. [14] gives a uniform predomain $(V, (\varpi_n)_{n\in\omega})$ where V satisfies (9). The proposition furthermore gives a uniform predomain $(S, (\varpi^S_n)_{n\in\omega})$ as well as uniform domains $(K, (\varpi^K_n)_{n\in\omega})$

$$\pi_0 = \lambda v.\bot \tag{10}$$

$$\pi_{n+1}(in_{\mathbb{Z}}(m)) = \lfloor in_{\mathbb{Z}}(m) \rfloor \tag{11}$$

$$\pi_{n+1}(in_1(*)) = \lfloor in_1(*) \rfloor \tag{12}$$

$$\pi_{n+1}(in_{Loc}(l,\infty)) = \lfloor in_{Loc}(l, n+1) \rfloor \tag{13}$$

$$\pi_{n+1}(in_{Loc}(l,m)) = \lfloor in_{Loc}(l, \min(n+1, m)) \rfloor \tag{14}$$

$$\pi_{n+1}(in_{\times}(v_1, v_2)) = \begin{cases} \lfloor in_{\times}(v_1', v_2') \rfloor & \text{if } \pi_n\, v_1 = \lfloor v_1' \rfloor \text{ and } \pi_n\, v_2 = \lfloor v_2' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{15}$$

$$\pi_{n+1}(in_{+}(\iota_i\, v)) = \begin{cases} \lfloor in_{+}(\iota_i\, v') \rfloor & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \qquad (i = 1, 2) \tag{16}$$

$$\pi_{n+1}(in_{\mu}\, v) = \begin{cases} \lfloor in_{\mu}\, v' \rfloor & \text{if } \pi_n v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{17}$$

$$\pi_{n+1}(in_{\forall}\, c) = \lfloor in_{\forall}(\pi_n^T\, c) \rfloor \tag{18}$$

$$\pi_{n+1}(in_{\to}\, f) = \left\lfloor in_{\to}\left(\lambda v.\, \begin{cases} \pi_n^T\,(f\, v') & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases}\right)\right\rfloor \tag{19}$$

Here the functions $\pi_n^S : S \to S_\bot$ and $\pi_n^K : K \to K$ and $\pi_n^T : TV \to TV$ are defined as follows:

$$\pi_0^S = \lambda s.\bot \qquad \pi_0^K = \lambda k.\bot \qquad \pi_0^T = \lambda c.\bot \tag{20}$$

$$\pi_{n+1}^S(s) = \begin{cases} \lfloor s' \rfloor & \text{if } \pi_{n+1} \circ s = \lambda l.\lfloor s'(l) \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{21}$$

$$\pi_{n+1}^K(k) = \lambda v.\lambda s.\begin{cases} k\, v'\, s' & \text{if } \pi_{n+1}\, v = \lfloor v' \rfloor \text{ and } \pi_{n+1}^S\, s = \lfloor s' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{22}$$

$$\pi_{n+1}^T(c) = \lambda k.\lambda s.\begin{cases} c\,(\pi_{n+1}^K\, k)\, s' & \text{if } \pi_{n+1}^S\, s = \lfloor s' \rfloor \\ \bot & \text{otherwise}\,. \end{cases} \tag{23}$$

Figure 3: Characterization of the projection functions $\pi_n : V \to V_\bot$.

and $(TV, (\varpi_n^T)_{n\in\omega})$ where S, K, and TV are as above. Now define the functions π_n as shown in Figure 3, by induction on n. We must show that $(V, (\pi_n)_{n\in\omega})$ is a uniform predomain.

One can show the following inequalities by mutual induction:

$$\begin{aligned}
\varpi_n &\sqsubseteq \pi_{n+1} \sqsubseteq \varpi_{n+1} \\
\varpi_n^S &\sqsubseteq \pi_n^S \sqsubseteq \varpi_{n+1}^S \\
\varpi_n^K &\sqsubseteq \pi_n^K \sqsubseteq \varpi_{n+1}^K \\
\varpi_n^T &\sqsubseteq \pi_n^T \sqsubseteq \varpi_{n+1}^T .
\end{aligned}$$

It follows from the first inequality that $(\pi_n)_{n\in\omega}$ is increasing. Furthermore, the same inequality gives that $\bigsqcup_{n\in\omega} \pi_n = \lambda v.\lfloor v \rfloor$ since $\bigsqcup_{n\in\omega} \varpi_n = \bigsqcup_{n\in\omega} \varpi_{n+1} = \lambda v.\lfloor v \rfloor$. The remaining requirements in the definition of a uniform predomain are easy to check. □

From here on, let V and $(\pi_n)_{n\in\omega}$ be as in the proposition above. We furthermore use the abbreviations, notation for injections, etc. introduced in the proposition; in particular, $TV = (V \to S \to Ans) \to S \to Ans$. Additionally, abbreviate $\lambda_l = in_{Loc}(l, \infty)$ and $\lambda_l^n = in_{Loc}(l, n)$. Let $error_{Ans} \in Ans$ be the "error answer" and let $error \in TV$ be the "error computation":

$$\begin{aligned}
error_{Ans} &= \lfloor \iota_2 * \rfloor \\
error &= \lambda k.\lambda s.\, error_{Ans} .
\end{aligned}$$

The proof of the proposition above gives:

Proposition 4.3.

1. $(S, (\pi_n^S)_{n\in\omega})$ *is a uniform predomain.*
2. $(K, (\pi_n^K)_{n\in\omega})$ *and* $(TV, (\pi_n^T)_{n\in\omega})$ *are uniform domains.*

In order to model the three operations of the untyped language that involve references, we define the three functions *alloc*, *lookup*, and *assign* in Figure 4.

Lemma 4.4. *The functions alloc, lookup, and assign are continuous.*

Notice that the definitions of *lookup* and *assign* depend on the projection functions π_n^S. Intuitively, if one for example looks up the approximate location $(l, n+1)$ in a store s, one only obtains the approximate element

$$alloc : V \to TV, \quad lookup : V \to TV, \quad assign : V \to V \to TV.$$

$$alloc\, v = \lambda k\, \lambda s.\, k\, (\lambda_{free(s)})\, (s[free(s) \mapsto v])$$
$$(\text{where } free(s) = \min\{n \in \mathbb{N} \mid n \notin \mathrm{dom}(s)\})$$

$$lookup\, v = \lambda k\, \lambda s. \begin{cases} k\, s(l)\, s & \text{if } v = \lambda_l \text{ and } l \in \mathrm{dom}(s) \\ k\, s'(l)\, s & \text{if } v = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n^S(s) = \lfloor s' \rfloor \\ \bot_{Ans} & \text{if } v = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n^S(s) = \bot \\ error_{Ans} & \text{otherwise} \end{cases}$$

$$assign\, v_1\, v_2 = \lambda k\, \lambda s. \begin{cases} k\, (in_1 *)\, (s[l \mapsto v_2]) & \text{if } v_1 = \lambda_l \text{ and } l \in \mathrm{dom}(s) \\ k\, (in_1 *)\, (s'[l \mapsto v_2']) & \text{if } v_1 = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n^S(s) = \lfloor s' \rfloor \\ & \text{and } \pi_n(v_2) = \lfloor v_2' \rfloor \\ \bot_{Ans} & \text{if } v_1 = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } (\pi_n^S(s) = \bot \text{ or } \pi_n(v_2) = \bot) \\ error_{Ans} & \text{otherwise} \end{cases}$$

Figure 4: Functions used for interpreting reference operations.

$\pi^S_{n+1}(s)(l)$ as result. It would not suffice to define, e.g., $\mathit{lookup}(\lambda^{n+1}_l)(k)(s) = \bot$ for $l \in \mathrm{dom}(s)$, and hence avoid mentioning the projection functions: *lookup* would then not be continuous.

We are now ready to define the untyped semantics.

Definition 4.5. *Let t be a term and let X be a set of variables such that* $\mathrm{FV}(t) \subseteq X$. *The* untyped semantics of t with respect to X *is the continuous function $[\![t]\!]_X : V^X \to TV$ defined by induction on t in Figures 5 and 6.*

Definition 4.6. *Let t be a term with no free term variables or type variables. The* program semantics of t *is the element $[\![t]\!]^{\mathrm{P}}$ of Ans defined by*

$$[\![t]\!]^{\mathrm{P}} = [\![t]\!]_\emptyset \emptyset\, k_{\mathit{init}}\, s_{\mathit{init}}$$

where

$$k_{\mathit{init}} = \lambda v.\lambda s. \begin{cases} \lfloor \iota_1\, m \rfloor & \text{if } v = \mathit{in}_{\mathbb{Z}}(m) \\ \mathit{error}_{\mathit{Ans}} & \text{otherwise} \end{cases}$$

and where $s_{\mathit{init}} \in S$ is the empty store.

Remark. The model in this section differs slightly from the BST model. First, the projection functions have been modified in order to ease calculations. Second, the semantic functions *lookup* and *assign* depend on projections of entire stores, not just projections of the individual values to be looked up or stored. This latter modification seems necessary when relations on stores must be described by the more refined "worlds" in this article. Intuitively, the refined worlds allow binary relations on stores that are not simply composed from binary relations on the individual values in the stores.

5. Ultrametric spaces

We recall some basic definitions and properties about metric spaces. For more details, see for example de Bakker and de Vink [18] or the long version of the article about the BST model [16].

A metric space (X, d) is 1-*bounded* if $d(x, y) \leq 1$ for all x and y in X. An *ultrametric space* is a metric space that satisfies the 'ultrametric inequality,'

$$d(x, z) \leq \max(d(x, y), d(y, z)),$$

and not just the weaker triangle inequality (where one has + instead of max on the right-hand side). It might be helpful to think of the function d of an

For every t with $\mathrm{FV}(t) \subseteq X$, define the continuous $[\![t]\!]_X : V^X \to TV$ by induction on t:

$$
\begin{aligned}
[\![x]\!]_X\rho &= \eta(\rho(x)) \\
[\![\overline{m}]\!]_X\rho &= \eta(in_{\mathbb{Z}}\, m) \\
[\![\mathsf{ifz}\, t_0\, t_1\, t_2]\!]_X\rho &= [\![t_0]\!]_X\rho \star \lambda v_0. \begin{cases} [\![t_1]\!]_X\rho & \text{if } v_0 = in_{\mathbb{Z}}\, 0 \\ [\![t_2]\!]_X\rho & \text{if } v_0 = in_{\mathbb{Z}}\, m \text{ where } m \neq 0 \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![t_1 \pm t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2. \begin{cases} \eta(in_{\mathbb{Z}}(m_1 \pm m_2)) \\ \qquad \text{if } v_1 = in_{\mathbb{Z}}\, m_1 \\ \qquad \text{and } v_2 = in_{\mathbb{Z}}\, m_2 \\ \mathit{error} \quad \text{otherwise} \end{cases} \\
[\![()]\!]_X\rho &= \eta(in_1\, *) \\
[\![(t_1,\, t_2)]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2.\, \eta(in_{\times}(v_1, v_2)) \\
[\![\mathsf{fst}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_1) & \text{if } v = in_{\times}(v_1, v_2) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{snd}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_2) & \text{if } v = in_{\times}(v_1, v_2) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{void}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \mathit{error} \\
[\![\mathsf{inl}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{+}(\iota_1\, v)) \\
[\![\mathsf{inr}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{+}(\iota_2\, v))
\end{aligned}
$$

(Continued in Figure 6.)

Figure 5: Untyped semantics of terms.

$$
\begin{aligned}
[\![\mathsf{case}\, t_0\ x_1.t_1\ x_2.t_2]\!]_X\rho &= [\![t_0]\!]_X\rho \star \lambda v_0.\ \begin{cases} [\![t_1]\!]_{X,x_1}(\rho[x_1 \mapsto v]) & \text{if } v_0 = in_+(\iota_1\, v) \\ [\![t_2]\!]_{X,x_2}(\rho[x_2 \mapsto v]) & \text{if } v_0 = in_+(\iota_2\, v) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\lambda x.t]\!]_X\rho &= \eta(in_\to(\lambda v.\ [\![t]\!]_{X,x}(\rho[x \mapsto v]))) \\
[\![t_1\, t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\ [\![t_2]\!]_X\rho \star \lambda v_2.\ \begin{cases} g\, v_2 & \text{if } v_1 = in_\to\, g \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{fix}\, f.\lambda x.t]\!]_X\rho &= \eta(in_\to(\mathit{fix}(\lambda g^{V\to TV}.\ \lambda v.\ [\![t]\!]_{X,f,x}(\rho[f \mapsto in_\to\, g,\ x \mapsto v])))) \\
[\![\mathsf{fold}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \eta(in_\mu\, v) \\
[\![\mathsf{unfold}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \begin{cases} \eta(v_0) & \text{if } v = in_\mu\, v_0 \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\Lambda\alpha.t]\!]_X\rho &= \eta(in_\forall\, ([\![t]\!]_X\rho)) \\
[\![t\, [\tau]]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \begin{cases} c & \text{if } v = in_\forall\, c \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{ref}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \mathit{alloc}\, v \\
[\![!t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \mathit{lookup}\, v \\
[\![t_1 := t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\ [\![t_2]\!]_X\rho \star \lambda v_2.\ \mathit{assign}\, v_1\, v_2
\end{aligned}
$$

Figure 6: Untyped semantics of terms (ctd.)

ultrametric space (X, d) not as a measure of (euclidean) distance between elements, but rather as a measure of the degree of similarity between elements.

A function $f : X_1 \to X_2$ from a metric space (X_1, d_1) to a metric space (X_2, d_2) is *non-expansive* if $d_2(f(x), f(y)) \leq d_1(x, y)$ for all x and y in X_1. Stronger, such a function f is *contractive* if there exists $c < 1$ such that $d_2(f(x), f(y)) \leq c \cdot d_1(x, y)$ for all x and y in X_1.

A metric space is *complete* if every Cauchy sequence has a limit. By Banach's fixed-point theorem, every contractive function from a non-empty, complete metric space to itself has a unique fixed point.

For a given complete metric space, consider the function *fix* that maps every contractive operator to its unique fixed-point. On complete ultrametric spaces, *fix* is non-expansive in the following sense [5]:

Proposition 5.1. *Let (X, d) be a non-empty, complete ultrametric space. For all contractive functions f and g from (X, d) to itself, $d(\mathit{fix}\, f, \mathit{fix}\, g) \leq d(f, g)$.*

All the metric spaces we consider satisfy the following property:

Definition 5.2. *A metric space is* bisected *if all non-zero distances are of the form 2^{-n} for some natural number $n \geq 0$.*

The following notation is convenient when working with bisected metric spaces: in such a space, $x =_n y$ means that $d(x, y) \leq 2^{-n}$. Notice that each relation $=_n$ is an equivalence relation. Here transitivity follows from the ultrametric inequality. Also, notice that a bisected metric space is one-bounded. In other words, the relation $x =_0 y$ always holds.

Proposition 5.3. *Let (X_1, d_1) and (X_2, d_2) be bisected metric spaces. A function $f : X_1 \to X_2$ is non-expansive if and only if*

$$x_1 =_n x_1' \implies f(x_1) =_n f(x_1')$$

holds for all $x_1, x_1' \in X_1$ and all natural numbers $n > 0$.

5.1. Categories of ultrametric spaces

Let $\mathsf{CBUlt}_{\mathsf{ne}}$ be the category with non-empty, complete, 1-bounded ultrametric spaces as objects and non-expansive functions as morphisms. This

category is cartesian closed [31, 16]; here one needs the ultrametric inequality. The terminal object is the one-point metric space. Binary products are defined in the natural way: $(X_1, d_1) \times (X_2, d_2) = (X_1 \times X_2,\ d_{X_1 \times X_2})$ where

$$d_{X_1 \times X_2}((x_1, x_2), (y_1, y_2)) = \max(d_1(x_1, y_1), d_2(x_2, y_2))\,.$$

The exponential $(X_1, d_1) \to (X_2, d_2)$ has the set of non-expansive functions from (X_1, d_1) to (X_2, d_2) as the underlying set, and the 'sup'-metric $d_{X_1 \to X_2}$ as distance function: $d_{X_1 \to X_2}(f, g) = \sup\{d_2(f(x), g(x)) \mid x \in X_1\}$. For both products and exponentials, limits are pointwise.

Let $\mathsf{PreCBUlt_{ne}}$ be the category of *pre-ordered*, non-empty, complete, 1-bounded ultrametric spaces. Objects of this category are pairs $(A, \leq)$ consisting of an object A of $\mathsf{CBUlt_{ne}}$ and a preorder $\leq$ on the underlying set of A such that the following condition holds: if $(a_n)_{n\in\omega}$ and $(b_n)_{n\in\omega}$ are converging sequences in A with $a_n \leq b_n$ for all n, then also $\lim_{n\to\infty} a_n \leq \lim_{n\to\infty} b_n$. The morphisms of the category are the non-expansive and monotone functions between such objects. We refer to the objects of this category as 'continuous preorders'.

Birkedal et al. [14] generalize the standard construction of solutions to recursive metric-space equations [6, 19] to a large class of categories with metric-space structure on each set of morphisms. In particular, one can solve recursive equations in the category $\mathsf{PreCBUlt_{ne}}$:

Definition 5.4. *A functor* $F : \mathsf{PreCBUlt_{ne}}^{\mathrm{op}} \times \mathsf{PreCBUlt_{ne}} \to \mathsf{PreCBUlt_{ne}}$ *is* locally non-expansive *if* $d(F(f, g), F(f', g')) \leq \max(d(f, f'), d(g, g'))$ *for all* f,f', g, *and* g' *with appropriate domains and codomains. Stronger,* F *is* locally contractive *if there exists some* $c < 1$ *such that* $d(F(f, g), F(f', g')) \leq c \cdot \max(d(f, f'), d(g, g'))$ *for all* f,f', g, *and* g'.

Theorem 5.5 ([14])**.** *Every locally contractive functor* $F : \mathsf{PreCBUlt_{ne}}^{\mathrm{op}} \times \mathsf{PreCBUlt_{ne}} \to \mathsf{PreCBUlt_{ne}}$ *has a unique fixed point: there exists an object* Z *of* $\mathsf{PreCBUlt_{ne}}$ *such that* $Z \cong F(Z, Z)$*, and if* Z' *is another such object then* $Z \cong Z'$.

6. Bohr Relations on Uniform Domains and Predomains

We introduce the notion of Bohr relations on domains and predomains. And we equip spaces of such with complete bisected ultrametrics. To do this, we need additional structure, we require uniform domains and predomains.

First up, we introduce a Hausdorff metric on the admissible, downwards closed subsets of a uniform domain. This buys us the metric on Bohr relations on a uniform domain. Then we show that there is a simple bijective correspondence between chain-complete, downwards closed subsets of a uniform predomain and the admissible, downwards closed subsets of the uniform domain obtained by lifting. We define a metric on the former by means of this bijection and this gives us the metric on Bohr relations on a uniform predomain.

We apply standard metric space constructions such as Hausdorff distance and carving closed subsets out of complete metric spaces. As such, we save some mileage by appeal to standard (mostly completeness) results. But the route is sufficiently indirect that going directly for the Theorems 6.9 and 6.16 by brute force is a viable alternative; indeed, this was the approach of the authors in [12].

6.1. Distance on $ADSub(D)$

In the following subsection $(D, (\pi_n)_{n\in\omega})$ denotes an arbitrary uniform domain.

Based on the additional structure on the domain D given by the projections, we build a metric on D:

Proposition 6.1. *There is a (unique) complete, bisected, ultrametric d_π on D such that for any $n \in \omega$ and any two $d, e \in D$ we have*

$$d =_n e \iff \pi_n(d) = \pi_n(e).$$

Proof. We define the map $d_\pi : D \times D \to \mathbb{R}$ by mapping any two $d, e \in D$ to

$$d_\pi(d,e) = \begin{cases} 0 & \text{if } d = e \\ 2^{-\max\{n\in\omega \mid \pi_n(d)=\pi_n(e)\}} & \text{if } d \neq e. \end{cases}$$

Let us initially verify that this is well-defined, we need to show that for $d \neq e$ we have that the set $\{n \in \omega \mid \pi_n(d) = \pi_n(e)\}$ is non-empty and finite. The former is a consequence of having $\pi_0(d) = \bot = \pi_0(e)$. Note now that for $m \leq n$ we have that $\pi_n(d) = \pi_n(e)$ implies $\pi_m(d) = \pi_m(e)$ since we have $\pi_m(d) = \pi_{\min(m,n)}(d) = \pi_m(\pi_n(d)) = \pi_m(\pi_n(e)) = \pi_{\min(m,n)}(e) = \pi_m(e)$. If now the set in question was infinite, then all projections of d and e would agree and they would be equal, contradicting our assumption.

By similar reasoning we easily show that for any $n \in \omega$ and any two $d, e \in D$ we have $d =_n e$ iff $\pi_n(d) = \pi_n(e)$. The map d_π is bisected by construction and for any two $d, e \in A$ we easily have that $d = e$ iff $d_\pi(d, e) = 0$ and that $d_\pi(d, e) = d_\pi(e, d)$. It remains to prove the strong triangle inequality and completeness. In search for the former, we pick $d, e, f \in D$ and aim to prove

$$d_\pi(d, f) \leq \max(d_\pi(d, e), d_\pi(e, f)).$$

Without loss of generality we may assume $d \neq f$, $d \neq e$ and $e \neq f$. There are $n, m \in \omega$ such that $d_\pi(d, e) = 2^{-n}$ and $d_\pi(d, e) = 2^{-m}$, let $l = \min(n, m)$. But then $d =_l e$ and $e =_l f$ and so $\pi_l(d) = \pi_l(e) = \pi_l(f)$ and we have

$$d_\pi(d, f) \leq 2^{-l} = 2^{-\min(n,m)} = \max(2^{-n}, 2^{-m}) = \max(d_\pi(d, e), d_\pi(e, f)).$$

To prove completeness we take an arbitrary Cauchy sequence $(d_n)_{n\in\omega}$ in D, we must build an $d \in D$ such that $\lim_n d_n = d$. For each $m \in \omega$ we pick an $M_m \in \omega$ such that we for any $n \geq M_m$ have that $d_n =_m d_{M_m}$. We may without loss of generality assume that $M_m \leq M_{m+1}$ for all $m \in \omega$. Our candidate for the limit now is

$$d = \bigsqcup_{m\in\omega} \pi_m(d_{M_m}).$$

To verify that this least upper bound actually exists, we remark that for any $m \in \omega$ we have

$$\pi_m(d_{M_m}) = \pi_m(d_{M_{m+1}}) \sqsubseteq \pi_{m+1}(d_{M_{m+1}}).$$

To finally prove that d is the limit we take any $m \in \omega$ and note that for any $n \geq M_m$ we have that

$$\begin{aligned}
\pi_m(d) &= \pi_m\left(\bigsqcup_{o\in\omega} \pi_o(d_{M_o})\right) \\
&= \bigsqcup_{o\in\omega} \pi_{\min(m,o)}(d_{M_o}) \\
&= \bigsqcup_{o\geq m} \pi_m(d_{M_o}) \\
&= \bigsqcup_{o\geq m} \pi_m(d_{M_m}) \\
&= \pi_m(d_{M_m}) \\
&= \pi_m(d_n)
\end{aligned}$$

which as noted implies that $d =_m d_n$ and we are done. □

We recollect the notion of Hausdorff distance:

Definition 6.2. *The* Hausdorff *distance d_H between two non-empty subsets $X, Y \subseteq M$ of a 1-bounded metric space (M, d) is defined as follows:*

$$d_H(X,Y) = \max\big(\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\big)$$

The notion of Hausdorff distance is standard, cf. Definitions 2.2 and 2.6 and Lemma 2.7 of [18] for an expository presentation. We restrict to 1-bounded metric spaces to avoid dealing with unbounded suprema and also focus on non-empty sets to simplify the presentation.

The Hausdorff distance is not a metric on the entire set of non-empty subsets of M as a distance of zero may fail to imply equality. But if we restrict ourselves to the *closed*, non-empty subsets we get a proper metric and completeness carries over:

Proposition 6.3. *The Hausdorff distance is a 1-bounded metric on the set $\mathcal{P}_{ncl}(X)$ of non-empty and closed subsets of a 1-bounded metric space (X, d). $(\mathcal{P}_{ncl}(X), d_H)$ is ultrametric and complete and if (X, d) is ultrametric and complete, respectively.*

These are textbook result, cf. Theorems 2.3 and 2.10 of [18].

Intuitively, the Hausdorff distance between X and Y is the least distance r such that for any $x \in X$ we can find $y \in Y$ with mutual distance no greater than r and vice versa. This intuition is captured in the following proposition under the assumption that the the underlying metric space is bisected:

Proposition 6.4. *Let (M, d) be a bisected metric space. Then for any non-empty $X, Y \subseteq M$ we have that $d_H(X, Y)$ is zero or of the form 2^{-n} for some $n \in \omega$ and for any $n \in \omega$ we get*

$$X =_n Y \iff \forall x \in X \exists y \in Y.\ x =_n y \wedge \forall y \in Y \exists x \in X.\ x =_n y.$$

Proof. To prove that for two non-empty $X, Y \subseteq M$ we have $d_H(X, Y) \in \{0\} \cup \{\, 2^{-m} \mid m \in \omega \,\}$ we simply observe that this set is closed under non-empty suprema and infima.

We now proceed to prove the biimplication. Pick non-empty $X, Y \subseteq M$ and $n \in \omega$ arbitrarily. To prove that the left hand side implies the right hand

side we assume that $X =_n Y$, take arbitrary $x \in X$ and need to find $y \in Y$ with $x =_n y$. For the sake of deriving a contradiction we assume that this cannot be done, i.e., that for every $y \in Y$ we have $d(x, y) > 2^{-n}$. By the assumption of the proposition this would mean that $d(x, y) \geq 2^{-n+1}$ for all $y \in Y$ which would imply that

$$d_H(X, Y) \geq \sup_{z \in X} \inf_{y \in Y} d(z, y) \geq \inf_{y \in Y} d(x, y) \geq 2^{-n+1}$$

which contradicts our assumption that $X =_n Y$. Proving the other conjunct proceeds similarly and the reverse implication is standard and does not rely on the special form of the metric. □

We are now in a position to define a distance on the set $ADSub(D)$ of admissible and downwards closed subsets of D as the Hausdorff distance on top of the distance d_π on D.

Proposition 6.5. *Any downwards closed and chain-complete subset of D is a closed subset of the metric space (D, d_π).*

Proof. Let $X \subseteq D$ be a downwards closed and chain-complete subset of D. Let $(x_m)_{m \in \omega}$ be a sequence in X with $\lim_m x_m = x$ for some $x \in D$. We must prove that $x \in X$ too. We know that

$$x = \bigsqcup_m \pi_m(x)$$

so by chain-completeness of D it suffices to show that $\pi_m(x) \in D$ holds for any $m \in \omega$. But this is a consequence of D being downwards closed since for any $m \in \omega$ there is $M_m \in \omega$ with $x =_m x_{M_m}$ which implies that $\pi_m(x) = \pi_m(x_{M_m}) \sqsubseteq x_{M_m} \in D$. □

Any admissible subset of D is non-empty as it contains the least element and so $ADSub(D) \subseteq \mathcal{P}_{ncl}(D)$. By Propositions 6.3 and Proposition 6.4 we know that d_H is a complete, bisected ultrametric on $\mathcal{P}_{ncl}(D)$. Hence d_H is a bisected ultrametric on $ADSub(D)$ too, where we overload d_H to mean both the Hausdorff distance on $\mathcal{P}_{ncl}(D)$ and its restriction to $ADSub(D)$. To obtain completeness we need the following:

Proposition 6.6. *The set $ADSub(D)$ is a closed subset of the metric space $(\mathcal{P}_{ncl}(D), d_H)$.*

Proof. Take some sequence $(X_m)_{m\in\omega}$ in $ADSub(D)$ and assume that $\lim_m X_m = X$ for some $X \in \mathcal{P}_{ncl}(D)$, we must prove that $X \in ADSub(D)$ too.

Let us initially prove that the least element $\bot \in D$ is in X. For any $m \in \omega$ there is M_m such that $X_{M_m} =_m X$. And as $\bot \in X_{M_m}$ we know that there is is a member, x_m say, of X with $\bot =_m x_m$ by Proposition 6.4. But then clearly $\lim_m x_m = \bot$ and since X was closed we have $\bot \in X$.

We now prove X chain-complete. We take an increasing chain $(x_m)_{m\in\omega}$ in X and aim to show that $x = \sqcup_m x_m \in X$. Take any $n \in \omega$, there is M_n such that $X_{M_n} =_n X$ and so the increasing chain $(\pi_n(x_m))_{m\in\omega}$ is in X_{M_n} as X_{M_n} was downwards closed. But X_{M_n} was chain-complete too and hence $\pi_n(x) \in X_{M_n}$ and so we may find $y_n \in X$ with $y_n =_n \pi_n(x) =_n x$. Clearly $\lim_n y_n = x$ and since X was closed we have $x \in X$.

Finally take $x, y \in D$ with $x \sqsubseteq y$ and $y \in X$, we need to show $x \in X$. For any any $m \in \omega$, there is M_m such that $X_{M_m} =_m X$ and hence $\pi_m(y) \in X_{M_m}$ as X_{M_m} was downwards closed. But then $\pi_m(x) \in X_{M_m}$ too as X_{M_m} was downwards closed and we proceed as above. □

In summa, we have the following:

Corollary 6.7. *There is a (unique) complete, bisected ultrametric d_H on $ADSub(D)$ such that for any two $X, Y \in ADSub(D)$ and any $n \in \omega$ we have*

$$X =_n Y \iff \pi_n(X) \subseteq Y \wedge \pi_n(Y) \subseteq X.$$

6.2. Bohr Relations on Uniform Domains

Definition 6.8 (Bohr Relation)**.** *A relation $R \subseteq D \times D$ on a domain D is called a* Bohr relation *if for any $e \in D$ we have that*

$$R(-, e) = \{d \mid (d, e) \in R\}$$

is admissible and downwards closed.

Theorem 6.9. *Let $(D, (\pi_n)_{n\in\omega})$ be a uniform domain. There is a (unique) complete, bisected ultrametric d_B on $BohrRel(D)$ such that for any two $R, S \in BohrRel(D)$ and any $n \in \omega$ we have*

$$R =_n S \iff \forall e \in D.\ \pi_n(R(-, e)) \subseteq S(-, e) \wedge \pi_n(S(-, e)) \subseteq R(-, e)$$

The proof proceeds along the lines of the proof of Theorem 6.16 only we appeal to Corollary 6.7 instead of Proposition 6.13.

It is not hard to prove the following:

Proposition 6.10. *Let $(D, (\pi_n)_{n\in\omega})$ be a uniform domain. Let $(R_n)_{n\in\omega}$ and $(S_n)_{n\in\omega}$ be sequences in $BohrRel(D)$ such that $\lim_n R_n = R$ and $\lim_n S_n = S$ for R and S also in $BohrRel(D)$. We then have that*

$$(\forall n \in \omega.\, R_n \subseteq S_n) \implies R \subseteq S.$$

Summing up, we have that the Bohr relations on a uniform domain equipped with the metric from Theorem 6.9 above and ordered by set-theoretic inclusion is an object of $\mathsf{PreCBUlt}_{\mathsf{ne}}$; see also Subsection 5.1.

6.3. Distance on $CDSub(A)$

In the following subsection, $(A, (\pi_n)_{n\in\omega})$ denotes an arbitrary uniform predomain.

Now let us return to uniform predomains. Recall our goal of obtaining a metric on the set of chain-complete and downwards closed subsets of a uniform predomain. We employ lifting to build a a uniform domain from a given uniform predomain and then apply the above theory.

It is well known that we may lift a predomain A to a domain $A_\perp$ by introducing a least element. This idea extends naturally to build uniform domains from uniform predomains:

Proposition 6.11. *Define, for $m \in \omega$, a new projection $\pi'_m : A_\perp \to A_\perp$ by*

$$\pi'_m(d) = \begin{cases} \pi_m(a) & \text{if } d = \lfloor a \rfloor \\ \perp & \text{if } d = \perp \end{cases}$$

for each $d \in A_\perp$. Then $(A_\perp, (\pi'_m)_{m\in\omega})$ is a uniform domain.

Proof. That $A_\perp$ is a domain and the new projections continuous are basic results of domain theory, see, e.g., section 8.3.4 of [34]. As the projections are strict by definition, it remains to verify the four defining axioms of uniform domains under the assumption of the axioms of uniform predomains:

For any $m \in \omega$ we need initially to show $\pi'_m \leq \pi'_{m+1}$. We prove this pointwise so we take $d \in A_\perp$ arbitrary. We may without loss of generality assume $d = \lfloor a \rfloor$ for some $a \in A$ and we have $\pi'_m(d) = \pi_m(a) \leq \pi_{m+1}(a) = \pi'_{m+1}(d)$.

We need to show $\sqcup_m \pi'_m = id_{A_\perp}$. As above, we take $d \in A_\perp$ arbitrary and discharge the case $d = \perp$ easily. So assume $d = \lfloor a \rfloor$ for some $a \in A$ we get that

$$\left(\bigsqcup_m \pi'_m\right)(d) = \bigsqcup_m \pi_m(a) = \left(\bigsqcup_m \pi_m\right)(a) = \lfloor a \rfloor = d.$$

For the third axiom we pick $m, n \in \omega$ and must show that $\pi'_m \circ \pi'_n = \pi'_n \circ \pi'_m = \pi'_{\min(m,n)}$. We prove this pointwise, so we take take $d \in A_\perp$ and may without loss of generality assume that $d = \lfloor a \rfloor$ for some $a \in A$. But then we need to show that

$$\pi'_m(\pi_n(a)) = \pi'_n(\pi_m(a)) = \pi_{\min(m,n)}(a)$$

which coincides with the third axiom of uniform predomains.

The fourth and final axiom requires π'_0 to be constant bottom which is obviously true as it holds for π_0 by assumption. □

We now give a bijective correspondence between the set $CDSub(A)$ of chain-complete, downwards closed subsets of A and the set $ADSub(A_\perp)$. For $X \subseteq A$ we let $X_\perp$ denote $\{\lfloor x \rfloor \mid x \in X\} \cup \{\perp\}$; this provides the bijection:

Proposition 6.12. *The map $(-)_\perp : \mathcal{P}(A) \to \mathcal{P}(A_\perp)$ establishes a bijective correspondence between $CDSub(A)$ and $ADSub(A_\perp)$.*

Proof. Take $X \in CDSub(A)$, we must prove that $X_\perp \in ADSub(A_\perp)$. To prove chain-completeness we take an increasing chain $(d_n)_{n\in\omega}$ in $X_\perp$ and we must show $\bigsqcup_n d_n \in X_\perp$ too. We may without loss of generality assume that no elements of the chain are bottom and hence we can choose an $x_n \in X$ with $d_n = \lfloor x_n \rfloor$ for all $n \in \omega$. But then $(x_n)_{n\in\omega}$ is an increasing chain too and we have $\bigsqcup_n x_n \in X$ by assumption. By continuity we get

$$\bigsqcup_n d_n = \bigsqcup_n \lfloor x_n \rfloor = \left\lfloor \bigsqcup_n x_n \right\rfloor \in X_\perp$$

and since $\perp \in X_\perp$ by definition we have proved admissibility. Downwards closure is simple, take $d, e \in A_\perp$ with $d \sqsubseteq e$ and $e \in X_\perp$, we must show $d \in X_\perp$ too. If $d = \perp$ we are done, otherwise there is $x \in A$ and $y \in X$ with $d = \lfloor x \rfloor$ and $e = \lfloor y \rfloor$ and hence $x \leq y$ which means that $x \in X$ too.

For any two $X, Y \in \mathcal{P}(A)$ we have that $X_\perp = Y_\perp$ readily implies $X = Y$. It remains to show that for any $X \in ADSub(A_\perp)$ there is an $Y \in CDSub(A)$ with $X = Y_\perp$. Unsurprisingly, we aim for

$$Y = \{a \in A \mid \lfloor a \rfloor \in X\}$$

which obviously has $Y_\perp = X$ since we must have $\perp \in X$. Continuity of $\lfloor - \rfloor$ immediately yields that $Y \in CDSub(A)$ and we are done. □

Proposition 6.13. *There is a (unique) complete, bisected ultrametric $d_\perp$ on $CDSub(A)$ such that for any two $X, Y \in CDSub(A)$ and any $n \in \omega$ we have*

$$X =_n Y \iff \pi_n(X) \subseteq Y_\perp \wedge \pi_n(Y) \subseteq X_\perp.$$

Proof. Given the preceding development, it should come as no surprise that we lift the uniform predomain to obtain the uniform domain $(A_\perp, (\pi'_m)_{m\in\omega})$ by Proposition 6.11. $A_\perp$ is endowed with the complete, bisected ultrametric d_π of Proposition 6.1 and $ADSub(A_\perp)$ with the complete, bisected ultrametric d_H of Corollary 6.7. For any two $X, Y \in CDSub(A)$ we now define

$$d_\perp(X, Y) = d_H(X_\perp, Y_\perp),$$

which yields a complete, bisected ultrametric on $CDSub(A)$ by Proposition 6.12. Now take any two $X, Y \in CDSub(A)$ and any $n \in \omega$, we must prove that

$$X =_n Y \iff \pi_n(X) \subseteq Y_\perp \wedge \pi_n(Y) \subseteq X_\perp.$$

Assume that we have $X =_n Y$, i.e., that $X_\perp =_n Y_\perp$. We take $x \in X$ and must prove that $\pi_n(x) \in Y_\perp$. We have $\lfloor x \rfloor \in X_\perp$ and hence there is $y \in Y_\perp$ such that $\lfloor x \rfloor =_n y$. As $Y_\perp$ is downwards closed we have $\pi'_n(y) \in Y_\perp$ and so

$$\pi_n(x) = \pi'_n(\lfloor x \rfloor) = \pi'_n(y)$$

and we have proved the desired; proving the other conjunct proceeds similarly.

Going for the other implication, we assume that $\pi_n(X) \subseteq Y_\perp \wedge \pi_n(Y) \subseteq X_\perp$ and must prove $X =_n Y$, i.e., that $X_\perp =_n Y_\perp$. So take $x \in X_\perp$, we must produce $y \in Y_\perp$ with $x =_n y$. We may without loss of generality assume $x \neq \perp$. So there is $x' \in X$ with $x = \lfloor x' \rfloor$ and our assumption buys us that $\pi'_n(x) = \pi_n(x') \in Y_\perp$. But we obviously have $\pi'_n(x) =_n x$ and are done; the symmetric property is proved similarly. □

6.4. Bohr Relations on Uniform Predomains

Definition 6.14 (Bohr Relation)**.** *A relation $R \subseteq A \times A$ on a predomain A is called a* Bohr relation *if for any $b \in A$ we have that*

$$R(-, b) = \{a \mid (a, b) \in R\}$$

is chain-complete and downwards closed.

As the defining property of Bohr relations is preserved by set-theoretic intersection, we easily get the following closure operator:

Proposition 6.15. *For any relation $R \subseteq A \times A$ we have that*

$$\overline{R} = \bigcap_{R \subseteq S \subseteq A \times A,\ S \text{ Bohr}} S$$

is a Bohr relation, furthermore it is least such that contain R.

Theorem 6.16. *Let $(A, (\pi_n)_{n \in \omega})$ be a uniform predomain. There is a (unique) complete, bisected ultrametric d_B on $BohrRel(A)$ such that for any two $R, S \in BohrRel(A)$ and any $n \in \omega$*

$$\begin{aligned} R =_n S \iff & \\ & \big[\forall (a,b) \in R.\ \pi_n(a) = \bot \vee (\exists a' \in A.\ \pi_n(a) = \lfloor a' \rfloor \wedge (a', b) \in S)\big] \wedge \\ & \big[\forall (a,b) \in S.\ \pi_n(a) = \bot \vee (\exists a' \in A.\ \pi_n(a) = \lfloor a' \rfloor \wedge (a', b) \in R)\big]. \end{aligned}$$

Proof. Consider the space of all functions $A \to CDSub(A)$. We may define a distance d_F between any two members $f, g \in A \to CDSub(A)$ of this set by setting

$$d_F(f, g) = \sup_{b \in A} d_\bot(f(b), g(b))$$

and it is a textbook result that this constitutes a complete ultrametric as this is the case for $CDSub(D)$ by Proposition 6.13. See, e.g., Lemmas 1.24 and 1.28 of [18] for details. As $d_\bot$ is bisected and the set $\{0\} \cup \{2^{-n} \mid n \in \omega\}$ is closed under non-empty suprema we have that d_F is bisected as well, and we may replace the supremum by the maximum in the above definition. We now define the map $\Phi : BohrRel(A) \to (A \to CDSub(A))$ by setting

$$\Phi(R)(b) = R(-, b)$$

for any $R \in BohrRel(A)$ and any $b \in A$. This is well-defined by the definition of Bohr relations and furthermore a bijection. We define the distance d_B between two $R, S \in BohrRel(A)$ by setting

$$d_B(R, S) = d_F(\Phi(R), \Phi(S))$$

and by a bijection argument we have that d_B is a complete, bisected ultrametric on $BohrRel(A)$.

Take now two $R, S \in BohrRel(A)$ and any $n \in \omega$ and assume that we have $R =_n S$. Take $(a, b) \in R$, assume that $\pi_n(a) = \lfloor a' \rfloor$ for some $a' \in A$, we must prove that $(a', b) \in S$. By definition we have $\Phi(R) =_n \Phi(S)$ which means that

$$R(-, b) = \Phi(R)(b) =_n \Phi(S)(b) = S(-, b)$$

and since $a \in R(-, b)$ we have $\pi_n(a) \in (S(-, b))_\perp$ by Proposition 6.13. But since $\pi_n(a) = \lfloor a' \rfloor$ we must have $a' \in S(-, b)$, i.e., $S(a', b)$. Proving the second conjunct of the right hand side of the biimplication proceeds similarly.

So assume now that the right hand side of the desired biimplication holds, we must prove that $R =_n S$. This means proving $\Phi(R) =_n \Phi(S)$ which again comes down to proving that for any $b \in A$ we have

$$R(-, b) = \Phi(R)(b) =_n \Phi(S)(b) = S(-, b)$$

So take $a \in R(-, b)$, i.e., $R(a, b)$ holds. We must by Proposition 6.13 prove that $\pi_n(a) \in (S(-, b))_\perp$ but this is exactly what the first disjunct of the right hand side gives us. And the second disjunct similarly buys us the converse implication. □

As was the case for uniform domains, we can prove the following:

Proposition 6.17. *Let $(A, (\pi_n)_{n\in\omega})$ be a uniform predomain. Let $(R_n)_{n\in\omega}$ and $(S_n)_{n\in\omega}$ be sequences in $BohrRel(A)$ such that $\lim_n R_n = R$ and $\lim_n S_n = S$ for R and S also in $BohrRel(A)$. We then have that*

$$(\forall n \in \omega.\, R_n \subseteq S_n) \implies R \subseteq S.$$

As concluded for Bohr Relations on uniform domains, we also have that the Bohr relations on a uniform predomain equipped with the metric from Theorem 6.16 above and ordered by set-theoretic inclusion is an object of $\mathsf{PreCBUlt}_{\mathsf{ne}}$; we refer to Subsection 5.1 for a definition of this category.

7. Building Worlds

In this section we build the space of worlds to be used in our Kripke logical relation. The space of worlds is obtained using Theorem 5.5, i.e., as the fixed point of a functor on certain pre-ordered metric spaces.

7.1. M-categories

Since we aim to apply Theorem 5.5 we need to keep track of whether the functors we build are locally contractive. To that end, it is most convenient to introduce the general M-categories of Birkedal et al. [14]; these are categories such as $\mathsf{CBUlt}_{\mathsf{ne}}$ or $\mathsf{PreCBUlt}_{\mathsf{ne}}$ that have a metric-space structure on each hom-set. This subsection can be skipped on a first reading: one can then read the definitions of the functors in the following sections while taking for granted that they do satisfy the required technical conditions.

Definition 7.1. *An* M-category *is a category* $\mathcal{C}$ *where each hom-set* $\mathcal{C}(A, B)$ *is equipped with a distance function turning it into a non-empty, complete,* 1*-bounded ultrametric space, and where each composition function*

$$\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$$

is non-expansive with respect to these metrics. (Here the domain of such a composition function is given the product metric.)

In other words, an M-category is a category where each hom-set is equipped with a metric which turns it into an object in $\mathsf{CBUlt}_{\mathsf{ne}}$; furthermore, each composition function must be a morphism in $\mathsf{CBUlt}_{\mathsf{ne}}$. We observe that if $\mathcal{C}$ is an M-category, then so are $\mathcal{C}^{\mathrm{op}}$ (with the same metric on each hom-set as in $\mathcal{C}$) and $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$ (with the product metric on each hom-set)

Proposition 7.2 ([14])**.** $\mathsf{CBUlt}_{\mathsf{ne}}$ *and* $\mathsf{PreCBUlt}_{\mathsf{ne}}$ *are* M*-categories when each hom-set is given the 'sup'-metric:*

$$d_{\mathcal{C}(X_1,X_2)}(f, g) = \sup\{d_{X_2}(f(x), g(x)) \mid x \in X_1\}$$

Definition 7.3. *A functor* $F : \mathcal{C} \to \mathcal{D}$ *between* M*-categories* $\mathcal{C}$ *and* $\mathcal{D}$ *is called* locally ε-Lipschitz *for some* $\varepsilon \geq 0$ *if, for all morphisms* $f, g : A \to B$ *of* $\mathcal{C}$*, we have*

$$d(F(f), F(g)) \leq \varepsilon \cdot d(f, g),$$

where the leftmost distance is in the hom-set $\mathcal{D}(F(A), F(B))$ *and the rightmost is in the hom-set* $\mathcal{C}(A, B)$*.*

We also say that the functor has the *local Lipschitz constant* ε. Notice that being locally contractive and locally non-expansive comes down to having a local Lipschitz constant strictly less than one and less than or equal to one, respectively.

The following are compositional rules for computing the local Lipschitz constant. They are stated in their most general form, notice in particular that the shrinking functor of Proposition 7.9 cannot readily be generalized to arbitrary M-categories. We omit all proofs as they are quite simple.

Proposition 7.4 (Identity Functor)**.** *Let $\mathcal{C}$ be an M-category. The identity functor on $\mathcal{C}$ is locally 1-Lipschitz.*

Proposition 7.5 (Constant Functor)**.** *Let $\mathcal{C}$ and $\mathcal{D}$ be M-categories and let D be a fixed object of $\mathcal{D}$. The constant functor that maps objects and morphisms of $\mathcal{C}$ to D and 1_D respectively is locally 0-Lipschitz.*

Proposition 7.6 (Functor Pairing)**.** *Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ be M-categories and let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{E}$ be locally ε-Lipschitz and locally δ-Lipschitz respectively. Then $\langle F, G\rangle : \mathcal{C} \to \mathcal{D} \times \mathcal{E}$ is locally $\max\{\delta, \varepsilon\}$-Lipschitz.*

Proposition 7.7 (Hom Functor)**.** *Let $\mathcal{C}$ be an M-category. The hom functor $(-) \to (-) : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathsf{CBUlt}_{\mathsf{ne}}$ defined in the standard way is locally 1-Lipschitz.*

Note that this is not, in general, an exponential. Rather, we just return the set of morphisms equipped with the metric structure it has according to the definition of M-categories.

Proposition 7.8 (Functor Composition)**.** *Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ be M-categories and let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{E}$ be locally ε-Lipschitz and locally δ-Lipschitz respectively. Then $G \circ F : \mathcal{C} \to \mathcal{E}$ is locally $\delta\varepsilon$-Lipschitz.*

Proposition 7.9 (Shrinking Functor)**.** *For any $0 < \varepsilon \leq 1$ we have that the functor $\varepsilon \cdot (-) : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ that multiplies all distances by ε is locally ε-Lipschitz.*

Proposition 7.10 (Product Functor)**.** *The standard metric product functor $(-) \times (-) : \mathsf{CBUlt}_{\mathsf{ne}} \times \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is locally 1-Lipschitz.*

Proposition 7.11 (Finite Maps Functor)**.** *Let X be an arbitrary set. The functor $X \rightharpoonup_{fin} (-) : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is locally 1-Lipschitz. It assigns the distance 1 to maps with different domains and the pointwise maximum otherwise, the action on morphisms is the obvious.*

7.2. The Space $\mathcal{W}$ of Worlds

We now turn to constructing the space of worlds. First, for any set X we let $\mathcal{L}(X)$ denote the set $\{(x, L) \in X \times \mathcal{P}(X) \mid x \in L\}$ of pairs of elements of X and subsets of X such that the former belongs to the latter. It is obviously non-empty provided that X is.

Proposition 7.12. *The functor* $I : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ *defined by*

$$\mathcal{L}(\mathcal{P}(V)) \times \left(\mathcal{P}(V) \to \frac{1}{2}\left[- \to \mathit{BohrRel}(S)\right]\right)$$

is locally $\frac{1}{2}$*-Lipschitz. And so is the functor* $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ *defined by*

$$\mathcal{P}(\mathit{Loc}) \times \mathcal{P}(\mathit{Loc}) \times (\mathbb{N} \rightharpoonup_{\mathit{fin}} I(-))\,.$$

The proof is a simple application of the above propositions. But the definitional one liners call for a few comments: We implicitly equip $\mathcal{L}(\mathcal{P}(V))$, $\mathcal{P}(V)$ and $\mathcal{P}(\mathit{Loc})$ with the discrete metric and, as such, consider them objects of $\mathsf{CBUlt_{ne}}$. The rightmost arrow in the definition of the functor I is the standard hom functor on $\mathsf{PreCBUlt_{ne}}$, i.e., it is the set of all non-expansive and monotone functions equipped with the supremum metric (see Proposition 7.7). And the arrow preceding that is the standard hom functor on $\mathsf{CBUlt_{ne}}$ but reduces to the full function space because of the discrete metric on $\mathcal{P}(V)$.

We need some notation to work with the output of the functors; we strive for compatibility with the nomenclature of LADR [21]. Let A be an object of $\mathsf{PreCBUlt_{ne}}$ and let $\Delta \in W(A)$, we write

$$\Delta = (\Delta.\varsigma_1, \Delta.\varsigma_2, \Delta.\mathcal{I})$$

for $\Delta.\varsigma_1, \Delta.\varsigma_2 \subseteq \mathit{Loc}$ and $\Delta.\mathcal{I} \in \mathbb{N} \rightharpoonup_{\mathit{fin}} I(A)$. Intuitively, a *world*[2] Δ oversees pairs of stores. It has a set of *left locations* $\Delta.\varsigma_1$ and *right locations* $\Delta.\varsigma_2$ that keep track of the allocated locations in the left and right hand side stores, respectively. Also it has an *island map* $\Delta.I$ that holds islands, each of which manages separate parts of the stores.

[2]Outside of this subsection, we speak of worlds only as the results of applying W to the specific fixed-point $\hat{\mathcal{W}}$ that we produce below, not to an arbitrary object of $\mathsf{PreCBUlt_{ne}}$.

For $\Theta \in I(A)$ we write

$$\Theta = (\Theta.CP, \Theta.PL, \Theta.HL)$$

for $\Theta.CP \subseteq V$, $\Theta.PL \subseteq \mathcal{P}(V)$ and $\Theta.HL \in \mathcal{P}(V) \to \frac{1}{2}\,[A \to BohrRel(S)]$. An *island*[3] Θ has three components, the *current population* $\Theta.CP$, the *population law* $\Theta.PL$ and the *heap law* $\Theta.HL$. The population captures the current state of the island, it may vary over time, but only within the bonds given by the population law: because of our use of $\mathcal{L}(\mathcal{P}(V))$ instead of $\mathcal{P}(V) \times \mathcal{P}(\mathcal{P}(V))$ in the definition of the functor I we get that $\Theta.CP \in \Theta.PL$ always holds. The heap law provides the set of pairs of heaps that the island accepts; the idea is to feed it the current population and the current world.

Returning to the technical development, we have a locally $\frac{1}{2}$-Lipschitz functor $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ and are almost ready to apply the fixed-point existence theorem that will give us the space of worlds $\mathcal{W}$. First we must remedy one shortcoming, though: the functor maps into $\mathsf{CBUlt_{ne}}$ so we must equip the images under W of objects with continuous preorders; this will give us a functor that maps into $\mathsf{PreCBUlt_{ne}}$.

Definition 7.13. *Let A be an object of $\mathsf{PreCBUlt_{ne}}$. For any two $\Delta_1, \Delta_2 \in W(A)$ we say that Δ_2* extends *Δ_1 and we write $\Delta_1 \sqsubseteq \Delta_2$ if we have*

$$\Delta_1.\varsigma_1 \subseteq \Delta_2.\varsigma_1 \wedge \Delta_1.\varsigma_2 \subseteq \Delta_2.\varsigma_2 \wedge \forall n \in \mathrm{dom}(\Delta_1.\mathcal{I}).\ \Delta_1.\mathcal{I}(n) \sqsubseteq \Delta_2.\mathcal{I}(n),$$

where we write $\Theta_1 \sqsubseteq \Theta_2$ for any two $\Theta_1, \Theta_2 \in I(A)$ if we have

$$\Theta_1.CP \subseteq \Theta_2.CP \wedge \Theta_1.PL = \Theta_2.PL \wedge \Theta_1.HL = \Theta_2.HL.$$

On the conceptual level, world extension has two separate components. We may add new islands to the island map, often to manage newly allocated store; there are no restrictions on these new islands with respect to the old world. This is known as *width extension* in LADR. But the existing islands can also change: their populations may grow within the bounds of the population law. The population and heap laws are themselves immutable, but as we apply the heap law to the current population, it may permit different pairs of stores in the old and new worlds. Such population growth loosely corresponds to a state change of some existing object in the store; it is termed *depth extension* in LADR.

[3] As for worlds, an island, in general, belongs to the result of applying I to the specific fixed point $\hat{\mathcal{W}}$ built below.

Proposition 7.14. *For any object A of $\mathsf{PreCBUlt_{ne}}$ we have that the above ordering on $W(A)$ is a continuous preorder; for any morphism $f : B \to A$ of $\mathsf{PreCBUlt_{ne}}$ we have that $W(f) : W(A) \to W(B)$ is monotone with respect to this ordering.*

Proof. The ordering is easily a preorder. To show that it is a continuous preorder we take sequences $(\Delta_n)_{n\in\omega}$ and $(\Gamma_n)_{n\in\omega}$ in $W(A)$ with limits $\lim_n \Delta_n = \Delta$ and $\lim_n \Gamma_n = \Gamma$ such that $\Delta_n \sqsubseteq \Gamma_n$ for all $n \in \omega$; we must show that we have $\Delta \sqsubseteq \Gamma$ too. We now pick an $m \in \omega$ such that $\Delta_m =_1 \Delta$ and that $\Gamma_m =_1 \Gamma$. But then by our construction we get

$$\Delta.\varsigma_1 = \Delta_m.\varsigma_1 \subseteq \Gamma_m.\varsigma_1 = \Gamma.\varsigma_1,$$

and by a similar argument we get that $\Delta.\varsigma_2 \subseteq \Gamma.\varsigma_2$ and $\mathrm{dom}(\Delta.\mathcal{I}) \subseteq \mathrm{dom}(\Gamma.\mathcal{I})$. Also, for any $n \in \mathrm{dom}(\Delta.\mathcal{I})$ we have that $\Delta_m.\mathcal{I}(n) =_1 \Delta.\mathcal{I}(n)$ and $\Gamma_m.\mathcal{I}(n) =_1 \Gamma.\mathcal{I}(n)$. But then

$$\Delta.\mathcal{I}(n).CP = \Delta_m.\mathcal{I}(n).CP \subseteq \Gamma_m.\mathcal{I}(n).CP = \Gamma.\mathcal{I}(n).CP$$

and also $\Delta.\mathcal{I}(n).PL = \Gamma.\mathcal{I}(n).PL$. Assume now that we have $\Delta.\mathcal{I}(n).HL \neq \Gamma.\mathcal{I}(n).HL$ for some $n \in \mathrm{dom}(\Delta.\mathcal{I})$, this means that we can pick $l \in \omega$ such that $\Delta.\mathcal{I}(n).HL \neq_l \Gamma.\mathcal{I}(n).HL$. Pick $k \in \omega$ such that $\Delta_k =_l \Delta$ and that $\Gamma_k =_l \Gamma$. But then

$$\Delta.\mathcal{I}(n).HL =_l \Delta_k.\mathcal{I}(n).HL = \Gamma_k.\mathcal{I}(n).HL =_l \Gamma.\mathcal{I}(n).HL$$

which is a contradiction.

We proceed to prove the second property. For $f : B \to A$ in $\mathsf{PreCBUlt_{ne}}$ and $\Delta \in W(A)$ arbitrary we can write out the action of the functor W on the morphism f as follows:

$$W(f)(\Delta.\varsigma_1, \Delta.\varsigma_2, \Delta.\mathcal{I}) = \big(\Delta.\varsigma_1, \Delta.\varsigma_2, \lambda n \in \mathrm{dom}(\Delta.\mathcal{I}).\ I(f)(\Delta.\mathcal{I}(n))\big).$$

For $\Theta \in I(A)$ arbitrary we can similarly write out the action of the functor I on the morphism f as follows:

$$I(f)(\Theta.CP, \Theta.PL, \Theta.HL) = \big(\Theta.CP, \Theta.PL, F(f)(\Theta.HL)\big),$$

where $F : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ is a shorthand for the component functor $\mathcal{P}(V) \to \frac{1}{2}(- \to \mathit{BohrRel}(S))$. From these observations it is immediate that $W(f)$ is monotone with respect to the above ordering. □

Corollary 7.15. *We may extend* $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ *to a locally* $\frac{1}{2}$*-Lipschitz functor* $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{PreCBUlt_{ne}}$ *by equipping the images of objects with the continuous preorder from Definition 7.13.*

Definition 7.16 (Worlds)**.** *Let* $\hat{\mathcal{W}}$ *be an object such that* $Sq : W(\hat{\mathcal{W}}) \cong \hat{\mathcal{W}}$ *holds in* $\mathsf{PreCBUlt_{ne}}$*; existence (and uniqueness up to isomorphism) is guaranteed by Theorem 5.5. We write* $\mathcal{W}$ *for* $W(\hat{\mathcal{W}})$*.*

We conclude with a remark on the metric on worlds. Reasoning in the finished model often require us to define non-expansive maps out of the space $\mathcal{W}$ of worlds, this is the case, e.g., when we build new types as well as heap laws for new islands. An example of this is found in Section 10. It is worthwhile to note, that in many cases we have non-expansiveness for free.

For any two worlds $\Delta_1, \Delta_2 \in \mathcal{W}$ with $\Delta_1 =_1 \Delta_2$, it is immediate by our use of discrete metric spaces and the finite maps functor in the construction of the functor W that we have

$$\Delta_1.\varsigma_1 = \Delta_2.\varsigma_1, \quad \Delta_1.\varsigma_2 = \Delta_2.\varsigma_2, \quad \mathrm{dom}(\Delta_1.\mathcal{I}) = \mathrm{dom}(\Delta_2.\mathcal{I})$$

and for any n in the shared domain of the island maps we have

$$\Delta_1.\mathcal{I}(n).CP = \Delta_2.\mathcal{I}(n).CP, \quad \Delta_1.\mathcal{I}(n).PL = \Delta_2.\mathcal{I}(n).PL,$$

again because of our use of the discrete metric on $\mathcal{L}(\mathcal{P}(V))$. This means that we cannot invalidate non-expansiveness by inspection of these components, or, phrased differently, if we never 'project out' any heap laws then we have non-expansiveness automatically. If, however, we make use of the heap laws, then we must proceed with caution; see, e.g., the proof Proposition 8.2 for an example of this.

8. Logical Relation

We now construct a Kripke logical relation that uses the space of worlds $\mathcal{W}$ obtained above. First up is the definition of types:

Definition 8.1 (Types)**.** *The space of types is* $\mathcal{T} = \mathcal{W} \to_{mon} \mathit{BohrRel}(V)$*, i.e., the set of non-expansive and monotone functions from* $\mathcal{W}$ *to* $\mathit{BohrRel}(V)$*. It comes equipped with the supremum metric, i.e., for* $\mu, \nu \in \mathcal{T}$ *and* $n \in \omega$ *we have*

$$\mu =_n \nu \iff \forall \Delta \in \mathcal{W}.\, \mu(\Delta) =_n \nu(\Delta).$$

This is well defined and the metric a complete, bisected ultrametric by Proposition 7.2.

We need quite a few different function spaces and introduce some section-specific notation to help out. An arrow between metric spaces denotes the set of non-expansive maps as, e.g., in $\mathcal{T} \to \mathcal{T}$. If the metric spaces are ordered and the arrow has the monotonicity subscript then we restrict attention to functions that are both non-expansive and monotone; an example is the definition of the space $\mathcal{T}$ of types above. A superscript 1 on the arrow, on the other hand, indicate that we only require the maps to be one-expansive, i.e., Lipschitz continuous with Lipschitz constant 2. This is a weaker requirement than non-expansive; in the context of bisected metric spaces it means that elements that are $(n+1)$-equal are mapped to elements that are n-equal, for all $n \in \omega$. An example is $\mathcal{W} \to^1 \mathit{BohrRel}(S)$ which we shall meet soon. (In general, a one-expansive function from X to Y is the same as a non-expansive function from X to $\frac{1}{2}Y$.)

There is some room for variation here. If we modified the functor I that builds the islands of worlds by replacing $\frac{1}{2}[- \to_{mon} \mathit{BohrRel}(S)]$ with $\frac{1}{2}(-) \to_{mon} \mathit{BohrRel}(S)$, i.e., by requiring *one-contractive* heap laws, then the operations *states*, *cont* and *comp* defined below would be non-expansive. But then the types would be one-contractive too, *and* we would rely on that to prove the allocation case of the fundamental theorem of logical relations. Similar considerations apply to the slight change of the projection functions compared to BST, see also the discussion at the end of Section 4; none of the variations appear superior to the other, however.

The full definition of the logical relation is shown in Figures 7 and 8. In the rest of this section we show that the logical relation is indeed well-defined. This essentially amounts to checking that all relations involved in the definition are Bohr relations, and that all functions involved in the definition are non-expansive or one-expansive and possibly monotone. In particular, the clause for recursive types is then well-defined by Banach's fixed-point Theorem.

In many (but not all) of the cases where we prove non-expansiveness it is actually possible to prove the stronger property of contractiveness. But this would clutter the picture, and so we skip it as we do not need this in the overall development. Remember also that the sets of values and states, V and S, are uniform predomains whereas the sets of computations and continuations, TV and K, are uniform domains, see Propositions 4.2 and 4.3. In particular, we have Bohr relations with metric on the former two

according to Definition 6.14 and Theorem 6.16 whereas the Bohr relations with metric on the latter two follow Definition 6.8 and Theorem 6.9.

We focus on the cases involving states and references. The remaining cases are essentially as in Birkedal et al. [16], where more details can be found.

8.1. Relations on states, continuations and computations

Proposition 8.2. *The operator states defined in Figure 8 satisfies* $states \in \mathcal{W} \to^1 BohrRel(S)$.

Proof. We must show that for $\Delta \in \mathcal{W}$ we have that $states(\Delta) \in BohrRel(S)$, and we must also show that $states : \mathcal{W} \to BohrRel(S)$ is one-expansive. The first property is a consequence of the definition of order on the set of states S, the finiteness of $\mathrm{dom}(\Delta.\mathcal{I})$ and the fact that for any $n \in \mathrm{dom}(\Delta.\mathcal{I})$ we have that $\Delta.\mathcal{I}(n).HL(\Delta.\mathcal{I}(n).CP)$ maps into $BohrRel(S)$. As for one-expansiveness, assume that $\Delta_1 =_{n+1} \Delta_2$. We must show that $states(\Delta_1) =_n states(\Delta_2)$. By the construction of worlds, we have

$$\Delta_1.\varsigma_1 = \Delta_2.\varsigma_1, \quad \Delta_1.\varsigma_2 = \Delta_2.\varsigma_2, \quad \text{and } \mathrm{dom}(\Delta_1.\mathcal{I}) = \mathrm{dom}(\Delta_2.\mathcal{I})\,.$$

Also we get for all $m \in \mathrm{dom}(\Delta_1.\mathcal{I}) = \mathrm{dom}(\Delta_2.\mathcal{I})$ that $\Delta_1.\mathcal{I}(m).CP = \Delta_2.\mathcal{I}(m).CP$ and hence that

$$\Delta_1.\mathcal{I}(m).HL(\Delta_1.\mathcal{I}(m).CP) =_{n+1} \Delta_2.\mathcal{I}(m).HL(\Delta_2.\mathcal{I}(m).CP),$$

in the space $\frac{1}{2}(\hat{\mathcal{W}} \to BohrRel(S))$. But this means that we only have n-equality in the space $\hat{\mathcal{W}} \to BohrRel(S)$, and as $Sq(\Delta_1) =_{n+1} Sq(\Delta_2)$ holds too, we get

$$\Delta_1.\mathcal{I}(m).HL(\Delta_1.\mathcal{I}(m).CP)(Sq(\Delta_1)) =_n \Delta_2.\mathcal{I}(m).HL(\Delta_2.\mathcal{I}(m).CP)(Sq(\Delta_2))\,.$$

Now let $(s_1, s_2) \in states(\Delta_1)$ and assume that $\pi_n^S(s_1) = \lfloor s_1' \rfloor \neq \bot$. We must show that $(s_1', s_2) \in states(\Delta_2)$. But this follows easily from the above equation. □

It is worthwhile to note that it is the (necessary) use of the shrinking factor $\frac{1}{2}$ in the construction of worlds in Section 7 that prevents us from proving non-expansiveness. This will haunt us throughout this subsection.

Lemma 8.3. *For all* $n \in \omega$ *and all* $\Delta_1, \Delta_2, \Delta_1' \in \mathcal{W}$ *with* $\Delta_1 =_n \Delta_2$ *and* $\Delta_1 \sqsubseteq \Delta_1'$ *there is* $\Delta_2' \in \mathcal{W}$ *with* $\Delta_1' =_n \Delta_2'$ *and* $\Delta_2 \sqsubseteq \Delta_2'$.

For $\Xi \vdash \tau$ we define the non-expansive $[\![\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ by induction on τ:

$$
\begin{aligned}
[\![\alpha]\!]_\Xi\varphi &= \varphi(\alpha)\\
[\![\texttt{int}]\!]_\Xi\varphi &= \lambda\Delta.\ \{(in_{\mathbb{Z}}n, in_{\mathbb{Z}}n) \mid n \in \mathbb{Z}\}\\
[\![1]\!]_\Xi\varphi &= \lambda\Delta.\ \{(in_1{*}, in_1{*})\}\\
[\![\tau_1 \times \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi \times [\![\tau_2]\!]_\Xi\varphi\\
[\![0]\!]_\Xi\varphi &= \lambda\Delta.\ \emptyset\\
[\![\tau_1 + \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi + [\![\tau_2]\!]_\Xi\varphi\\
[\![\texttt{ref}\ \tau]\!]_\Xi\varphi &= \mathit{ref}([\![\tau]\!]_\Xi\varphi)\\
[\![\forall\alpha.\tau]\!]_\Xi\varphi &= \lambda\Delta.\ \{(in_\forall c_1, in_\forall c_2) \mid \forall\nu \in \mathcal{T}.\ \forall\Delta' \sqsupseteq \Delta.\\
&\qquad\qquad (c_1, c_2) \in \mathit{comp}([\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])(\Delta')\}\\
[\![\mu\alpha.\tau]\!]_\Xi\varphi &= \mathit{fix}\,(\lambda\nu.\ \lambda\Delta.\ \{(in_\mu v_1, in_\mu v_2) \mid (v_1, v_2) \in [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu](\Delta)\})\\
[\![\tau_1 \to \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi \to [\![\tau_2]\!]_\Xi\varphi
\end{aligned}
$$

The following operators and elements are used above:

$$
\begin{aligned}
\times &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & \mathit{comp} &: \mathcal{T} \to (\mathcal{W} \to^1 \mathit{BohrRel}(TV))\\
+ &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & \mathit{cont} &: \mathcal{T} \to (\mathcal{W} \to^1_{mon} \mathit{BohrRel}(K))\\
\mathit{ref} &: \mathcal{T} \to \mathcal{T} & \mathit{states} &: \mathcal{W} \to^1 \mathit{BohrRel}(S)\\
\to &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & R_{Ans} &\in \mathit{BohrRel}(Ans)
\end{aligned}
$$

$$
\begin{aligned}
(\nu_1 \times \nu_2)(\Delta) &= \{\, (in_\times(v_1, v_2), in_\times(v_1', v_2')) \mid (v_1, v_1') \in \nu_1(\Delta) \wedge (v_2, v_2') \in \nu_2(\Delta) \,\}\\
(\nu_1 + \nu_2)(\Delta) &= \{\, (in_+(\iota_1\, v_1), in_+(\iota_1\, v_1')) \mid (v_1, v_1') \in \nu_1(\Delta) \,\} \cup\\
&\qquad \{\, (in_+(\iota_2\, v_2), in_+(\iota_2\, v_2')) \mid (v_2, v_2') \in \nu_2(\Delta) \,\}\\
(\nu_1 \to \nu_2)(\Delta) &= \{\, (in_\to f, in_\to f') \mid \forall\Delta' \sqsupseteq \Delta.\, \forall(v, v') \in \nu(\Delta')\,.(f\, v, f'\, v') \in \mathit{comp}(\nu_2)(\Delta') \,\}\\
\mathit{comp}(\nu)(\Delta) &= \{\, (c, c') \mid \forall(k, k') \in \mathit{cont}(\nu)(\Delta).\\
&\qquad\qquad \forall(s, s') \in \mathit{states}(\Delta).\, (c\, k\, s, c'\, k'\, s') \in R_{Ans} \,\}\\
\mathit{cont}(\nu)(\Delta) &= \{\, (k, k') \mid \forall\Delta' \sqsupseteq \Delta.\, \forall(v, v') \in \nu(\Delta').\\
&\qquad\qquad \forall(s, s') \in \mathit{states}(\Delta').\, (k\, v\, s, k'\, v'\, s') \in R_{Ans} \,\}\\
R_{Ans} &= \{\, (\bot, a) \mid a \in Ans\} \cup \{\, (\lfloor\iota_1\, m\rfloor, \lfloor\iota_1\, m\rfloor) \mid m \in \mathbb{Z} \,\}
\end{aligned}
$$

(Continued in Figure 8.)

Figure 7: Logical relation.

Definition of $\mathit{ref} : \mathcal{T} \to \mathcal{T}$

$$
\begin{aligned}
\mathit{ref}(\nu)(\Delta) = \big\{(\lambda_{l_1}, \lambda_{l_2}) \mid\ & \forall \Delta' \sqsupseteq \Delta.\, l_1 \in \Delta'.\varsigma_1 \wedge l_2 \in \Delta'.\varsigma_2 \wedge \\
& \forall (s_1, s_2) \in \mathit{states}(\Delta'). \\
& \quad (s_1(l_1), s_2(l_2)) \in \nu(\Delta') \wedge \\
& \quad \forall (v_1, v_2) \in \nu(\Delta'). \\
& \qquad (s_1[l_1 \mapsto v_1], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta')\big\} \cup \\
\big\{(\lambda^{n+1}_{l_1}, \lambda_{l_2}) \mid\ & \forall \Delta' \sqsupseteq \Delta.\, l_1 \in \Delta'.\varsigma_1 \wedge l_2 \in \Delta'.\varsigma_2 \wedge \\
& \forall (s_1, s_2) \in \mathit{states}(\Delta').\, \pi_n(s_1) = \lfloor s_1' \rfloor \implies \\
& \quad (s_1'(l_1), s_2(l_2)) \in \nu(\Delta') \wedge \\
& \quad \forall (v_1, v_2) \in \nu(\Delta').\, \pi_n(v_1) = \lfloor v_1' \rfloor \implies \\
& \qquad (s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta')\big\}.
\end{aligned}
$$

Definition of $\mathit{states} : \mathcal{W} \to^1 \mathit{BohrRel}(S)$

$$\mathit{states}(\Delta) = \{(s_1, s_2) \mid \mathrm{dom}(s_1) = \Delta.\varsigma_1 \wedge \mathrm{dom}(s_2) = \Delta.\varsigma_2 \wedge (s_1, s_2) \in \mathit{sep}(\Delta)\}$$

where $\mathit{sep}(\Delta)$ is an auxiliary relation on S defined by

$$
\begin{aligned}
(s_1, s_2) \in \mathit{sep}(\Delta) \iff & \exists \sigma_1, \sigma_2 : \mathrm{dom}(\Delta.\mathcal{I}) \to S. \\
& s_1 = \biguplus_{n \in \mathrm{dom}(\Delta.\mathcal{I})} \sigma_1(n) \quad \wedge \quad s_2 = \biguplus_{n \in \mathrm{dom}(\Delta.\mathcal{I})} \sigma_2(n) \quad \wedge \\
& \forall n \in \mathrm{dom}(\Delta.\mathcal{I}). \\
& \quad (\sigma_1(n), \sigma_2(n)) \in \Delta.\mathcal{I}(n).HL(\Delta.\mathcal{I}(n).CP)(Sq(\Delta))
\end{aligned}
$$

Figure 8: Logical relation (ctd.)

Proposition 8.4. *The operator cont defined in Figure 7 satisfies* $cont \in \mathcal{T} \to (\mathcal{W} \to^1_{mon} BohrRel(K))$.

Proof. We must show that for all $\nu \in \mathcal{T}$ and all $\Delta \in \mathcal{W}$ we have that $cont(\nu)(\Delta) \in BohrRel(K)$. We must furthermore show that $cont : \mathcal{T} \to (\mathcal{W} \to BohrRel(S))$ is non-expansive in the first argument and one-expansive in the second argument, and that for all $\nu \in \mathcal{T}$ and all $\Delta_1, \Delta_2 \in \mathcal{W}$ we have that

$$\Delta_1 \sqsubseteq \Delta_2 \implies cont(\nu)(\Delta_1) \subseteq cont(\nu)(\Delta_2).$$

The first property is an immediate consequence of the fact that R_{Ans} is itself a Bohr relation on the domain $\mathbb{Z}_\perp$. The expansiveness properties follow from Proposition 8.2, Lemma 8.3, and the definition of π^K_n in Figure 3. Monotonicity is immediate from the quantification over future worlds. □

Proposition 8.5. *The operator comp defined in Figure 7 satisfies* $comp \in \mathcal{T} \to (\mathcal{W} \to^1 BohrRel(TV))$.

The proof proceeds just as the proof of Proposition 8.4, except that one does not need to check monotonicity. This definition is, by the way, the exact point where we benefit from a continuation passing style semantics. The obvious direct style definition would not have continuations but rather call for some future world in which the results of the computations should be suitably related; this, however, is inherently chain-incomplete, and we would have a hard time producing relations in $BohrRel(TV)$.

8.2. Some Type Constructors

Proposition 8.6. *The operator ref defined in Figure 8 satisfies* $ref \in \mathcal{T} \to \mathcal{T}$.

Proof. Note first that, in both clauses, we quantify over pairs of states $(s_1, s_2) \in states(\Delta')$; in particular we that $l_1 \in \Delta'.\varsigma_1 = \mathrm{dom}(s_1)$ and $l_2 \in \Delta'.\varsigma_2 = \mathrm{dom}(s_2)$ by the definition of $states(\Delta')$ and so we only read and write allocated locations.

We must now show that for all $\nu \in \mathcal{T}$ and all $\Delta \in \mathcal{W}$ we have that $ref(\nu)(\Delta) \in BohrRel(V)$. Furthermore, we must show that $ref : \mathcal{T} \to \mathcal{W} \to BohrRel(V)$ is non-expansive in both arguments, and that for all $\nu \in \mathcal{T}$ and all $\Delta_1, \Delta_2 \in \mathcal{W}$ we have that

$$\Delta_1 \sqsubseteq \Delta_2 \implies ref(\nu)(\Delta_1) \subseteq ref(\nu)(\Delta_2).$$

The first property is a consequence of the fact that $states(\Delta)$ and $\nu(\Delta)$ are themselves Bohr relations for all $\Delta \in \mathcal{W}$. Monotonicity is immediate from the quantification over future worlds. Let us, however, prove non-expansiveness in some detail.

Let $n \in \omega$, $\nu_1, \nu_2 \in \mathcal{T}$ and $\Delta_1, \Delta_2 \in \mathcal{W}$ be given and assume that $\nu_1 =_{n+1} \nu_2$ and $\Delta_1 =_{n+1} \Delta_2$. We aim to show that

$$ref(\nu_1)(\Delta_1) =_{n+1} ref(\nu_2)(\Delta_2)\,.$$

Take $(v_1, v_2) \in ref(\nu_1)(\Delta_1)$ and assume that $\pi_{n+1}(v_1) = \lfloor v_1' \rfloor$ holds, we must prove $(v_1', v_2) \in ref(\nu_2)(\Delta_2)$. There must be $l_2 \in Loc$ such that $v_2 = \lambda_{l_2}$ and there must be $l_1 \in Loc$ and $m \leq n$ such that $v_1' = \lambda_{l_1}^{m+1}$. Since $ref(\nu_1)(\Delta_1) \in BohrRel(V)$ we have $(\lambda_{l_1}^{m+1}, \lambda_{l_2}) \in ref(\nu_1)(\Delta_1)$. We set forth to prove that we have $(\lambda_{l_1}^{m+1}, \lambda_{l_2}) \in ref(\nu_2)(\Delta_2)$ too.

According to definition, we take $\Delta_2' \sqsupseteq \Delta_2$ and must show that $l_1 \in \Delta_2'.\varsigma_1$ and $l_2 \in \Delta_2'.\varsigma_2$. By Lemma 8.3 we pick $\Delta_1' \sqsupseteq \Delta_1$ with $\Delta_1' =_{n+1} \Delta_2'$ and get that $l_1 \in \Delta_1'.\varsigma_1 = \Delta_2'.\varsigma_1$ and $l_2 \in \Delta_1'.\varsigma_2 = \Delta_2'.\varsigma_2$. We now pick $(s_1, s_2) \in states(\Delta_2')$ and assume that $\pi_m(s_1) = \lfloor s_1' \rfloor$ and get that $(s_1', s_2) \in states(\Delta_1')$ since $states(\Delta_1') =_m states(\Delta_2')$. Since $\pi_m(s_1') = \lfloor s_1' \rfloor$ we furthermore get $(s_1'(l_1), s_2(l_2)) \in \nu_1(\Delta_1')$ and

$$\begin{aligned}\forall (v_1, v_2) \in \nu_1(\Delta_1').\ \pi_m(v_1) = \lfloor v_1' \rfloor \implies \\ (s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in states(\Delta_1').\end{aligned}$$

Now $\pi_m(s_1'(l_1)) = \lfloor s_1'(l_1) \rfloor$ and so $(s_1'(l_1), s_2(l_2)) \in \nu_2(\Delta_2')$ as $\nu_1(\Delta_1') =_{n+1} \nu_2(\Delta_2')$. Take now $(v_1, v_2) \in \nu_2(\Delta_2')$ and assume that $\pi_m(v_1) = \lfloor v_1' \rfloor$, we must show that $(s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in states(\Delta_2')$. But since $(v_1', v_2) \in \nu_1(\Delta_1')$ and $\pi_m(v_1') = \lfloor v_1' \rfloor$ we get $(s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in states(\Delta_1')$ which in combination with the fact that $\pi_m(s_1'[l_1 \mapsto v_1']) = \lfloor s_1'[l_1 \mapsto v_1'] \rfloor$ gives us the desired. □

This interpretation of reference types differs markedly from ADR. The interpretation above is extensional whereas the one in ADR is intensional: it requires that the world must have an island that looks exactly as if it had been added according to the proof of the case of allocation in the proof of the fundamental theorem of logical relations. The intensional definition in ADR means that we may fail to recognize values as having reference type even though they, for some reason, behave just as references. The extensional definition above does, on the other hand, only support lookup and assignment.

It would not suffice to model a language with equality testing on references such as the language in ADR. We conjecture that some notion of bijective bookkeeping could be added to remedy this, but we have not pursued the matter.

Proposition 8.7. *The operator $\to$ defined in Figure 7 belongs to $\mathcal{T}\times\mathcal{T} \to \mathcal{T}$.*

We omit a detailed proof but note that the one-expansiveness (in the second argument) of the operator *comp* is cancelled out by the index-shift in projections, see Equation 19 in Figure 3. A similar story can be told about the interpretation of universal types and reference types; in the latter case we do not, however, rely on the projections but rather on the index-shift from $\lambda_{l_1}^{n+1}$ to π_n in the second clause of the definition of *ref*. In some sense, this is as far as the one-expansiveness caused by the shrinking factor gets, confer the comment following the proof of Proposition 8.2.

Proposition 8.8. *The operators $\times$ and $+$ defined in Figure 7 belong to $\mathcal{T} \times \mathcal{T} \to \mathcal{T}$.*

8.3. Interpretation of Types

Theorem 8.9. *For all $\Xi \vdash \tau$ we have that $[\![\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ defined by induction on τ according to Figure 7 is well-defined and non-expansive. Here $\mathcal{T}^\Xi$ is equipped with the product metric.*

Proof. This is immediate from Propositions 8.6, 8.7, and 8.8 for all except universal and recursive types. And verifying the claim for $\Xi \vdash \forall\alpha.\tau$ under the assumption that it holds for $\Xi, \alpha \vdash \tau$ is not hard.

Consider now the case of $\Xi \vdash \mu\alpha.\tau$. We assume that $[\![\tau]\!]_{\Xi,\alpha} : \mathcal{T}^{\Xi,\alpha} \to \mathcal{T}$ is well defined and non-expansive. For $\varphi \in \mathcal{T}^\Xi$ we define

$$\Phi_\varphi = \lambda\nu \in \mathcal{T}.\ \lambda\Delta \in \mathcal{W}.\ \{(in_\mu v_1, in_\mu v_2) \mid (v_1, v_2) \in [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu](\Delta)\}$$

and it is not hard to see that this constitutes a contractive map $\Phi_\varphi : \mathcal{T} \to \mathcal{T}$. This means that $fix(\Phi_\varphi)$ is well defined by Banach's fixed point theorem. Furthermore we have that for any two $\varphi_1, \varphi_2 \in \mathcal{T}^\Xi$ with $\varphi_1 =_n \varphi_2$ for some $n \in \omega$ we get $\Phi_{\varphi_1} =_{n+1} \Phi_{\varphi_2}$. It then follows from Proposition 5.1 that $fix(\Phi_{\varphi_1}) =_{n+1} fix(\Phi_{\varphi_2})$. In summa, $[\![\mu\alpha.\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ is well-defined and contractive. □

9. Fundamental Theorem of Logical Relations

This definition with ensuing lemma will do much of the bookkeeping for us in the proofs to come:

Definition 9.1. *For $\mu, \nu \in \mathcal{T}$ and $\Delta \in \mathcal{W}$ we define a relation on $V \to TV$ by*

$$\mu \to_\Delta \nu = \{(f_1, f_2) \mid \forall \Delta' \sqsupseteq \Delta.\ \forall (v_1, v_2) \in \mu(\Delta').\ (f_1 v_2, f_2 v_2) \in \mathit{comp}(\nu)(\Delta').$$

Lemma 9.2. *For $\nu \in \mathcal{T}$, $\Delta \in \mathcal{W}$ and $(v_1, v_2) \in \nu(\Delta)$ we have*

$$(\eta\, v_1, \eta\, v_2) \in \mathit{comp}(\nu)(\Delta),$$

and for $\mu, \nu \in \mathcal{T}$, $\Delta \in \mathcal{W}$, $(c_1, c_2) \in \mathit{comp}(\mu)(\Delta)$, $(f_1, f_2) \in \mu \to_\Delta \nu$ we have

$$(c_1 \star f_1, c_2 \star f_2) \in \mathit{comp}(\nu)(\Delta).$$

Proof. To prove the first, we take related pairs $(k_1, k_2) \in \mathit{cont}(\nu)(\Delta)$ and $(s_1, s_2) \in \mathit{states}(\Delta)$ and get that

$$((\eta\, v_1)\, k_1 s_2, (\eta\, v_2)\, k_2 s_2) = (k_1 v_1 s_1, k_2, v_2, s_2) \in R_{Ans}$$

by the definition of $\eta : V \to TV$ and $\mathit{cont}(\nu)(\Delta)$.

To prove the second, we similarly take related pairs $(k_1, k_2) \in \mathit{cont}(\nu)(\Delta)$ and $(s_1, s_2) \in \mathit{states}(\Delta)$ and must prove that

$$((c_1 \star f_1)\, k_1 s_1, (c_2 \star f_2)\, k_2 s_2) \in R_{Ans}.$$

By definition of $\star : TV \times (V \to TV) \to TV$ we get that

$$\big((c_1 \star f_1)\, k_1 s_1, (c_2 \star f_2)\, k_2 s_2\big) = \big(c_1(\lambda v_1.\, \lambda t_1.\, f_1 v_1 k_1 t_1) s_1, c_2(\lambda v_2.\, \lambda t_2.\, f_2 v_2 k_2 t_2) s_2\big)$$

and so it remains to prove that

$$(\lambda v_1.\, \lambda s_1'.\, f_1 v_1 k_1 s_1', \lambda v_2.\, \lambda s_2'.\, f_2 v_2 k_2 s_2') \in \mathit{cont}(\mu)(\Delta).$$

So we take $\Delta' \sqsupseteq \Delta$, $(v_1, v_2) \in \mu(\Delta')$, $(s_1', s_2') \in \mathit{states}(\Delta')$ and must prove that we have

$$(f_1 v_1 k_1 s_1', f_2 v_2 k_2 s_2') \in R_{Ans}.$$

But the definition of $\mu \to_\Delta \nu$ gives us that $(f_1 v_1, f_2 v_2) \in \mathit{comp}(\nu)(\Delta')$ and by monotonicity we have $(k_1, k_2) \in \mathit{cont}(\nu)(\Delta')$ and we are done. □

We are now ready to define what it means for two terms of the same type to be *semantically related*. First up is the definition of related environments:

Definition 9.3. *For every term environment $\Xi \vdash \Gamma$, every $\varphi \in \mathcal{T}^{\Xi}$ and every $\Delta \in \mathcal{W}$ we let $[\![\Gamma]\!]_{\Xi}\varphi(\Delta)$ be the binary relation on $V^{\mathrm{dom}(\Gamma)}$ defined by*

$$[\![\Gamma]\!]_{\Xi}\varphi(\Delta) = \{(\rho_1, \rho_2) \mid \forall x \in \mathrm{dom}(\Gamma).\, (\rho_1(x), \rho_2(x)) \in [\![\Gamma(x)]\!]_{\Xi}\varphi(\Delta)\}.$$

Definition 9.4. *Assume $\Xi \vdash \Gamma$ and terms t_1 and t_2 with free variables in* $\mathrm{dom}(\Gamma)$. *We say that t_1 and t_2 are* semantically related, *written $\Xi \mid \Gamma \models t_1 \sim t_2 : \tau$, if for all $\varphi \in \mathcal{T}^{\Xi}$, all $\Delta \in \mathcal{W}$, and all $(\rho_1, \rho_2) \in [\![\Gamma]\!]_{\Xi}\varphi(\Delta)$,*

$$\big([\![t_1]\!]_{\mathrm{dom}(\Gamma)}\rho_1, [\![t_2]\!]_{\mathrm{dom}(\Gamma)}\rho_2\big) \in \mathit{comp}([\![\tau]\!]_{\Xi}\varphi)(\Delta).$$

Theorem 9.5 (Fundamental Theorem)**.** *Semantic relatedness is preserved by all typing rules. In particular, we have that any typed term is semantically related to itself, i.e, for any $\Xi \mid \Gamma \vdash t : \tau$ we have $\Xi \mid \Gamma \models t \sim t : \tau$.*

Proof. We provide proofs for only a few interesting cases, and refer to BST [12] with associated technical report [16] for the remaining. The definitions that concern state and references have changed sufficiently that going through the cases of lookup, assignment and allocation in detail is reasonable.

The Case of Lookup

Consider the case of lookup. Assume that $\Xi \mid \Gamma \vdash t_1 \sim t_2 : \mathtt{ref}\ \tau$ holds, we must show that $\Xi \mid \Gamma \vdash !\, t_1 \sim !\, t_2 : \tau$ holds too. We unroll the definition; take $\varphi \in \mathcal{T}^{\Xi}$, $\Delta \in \mathcal{W}$ and $(\rho_1, \rho_2) \in [\![\Gamma]\!]_{\Xi}\varphi(\Delta)$ and aim to show that

$$([\![!\, t_1]\!]_X\rho_1, [\![!\, t_2]\!]_X\rho_2) \in \mathit{comp}([\![\tau]\!]_{\Xi}\varphi)(\Delta),$$

where we for brevity write X for $\mathrm{dom}(\Gamma)$. By definition we have that

$$([\![!\, t_1]\!]_X\rho_1, [\![!\, t_2]\!]_X\rho_2) = ([\![t_1]\!]_X\rho_1 \star \lambda v_1.\, \mathit{lookup}\, v_1, [\![t_2]\!]_X\rho_2 \star \lambda v_2.\, \mathit{lookup}\, v_2),$$

and by Lemma 9.2 we are down to proving

$$(\lambda v_1.\, \mathit{lookup}\, v_1, \lambda v_2.\, \mathit{lookup}\, v_2) \in [\![\mathtt{ref}\ \tau]\!]_{\Xi}\varphi \to_{\Delta} [\![\tau]\!]_{\Xi}\varphi.$$

Again we unroll, take $\Delta' \sqsupseteq \Delta$ and related pairs $(v_1, v_2) \in [\![\mathtt{ref}\ \tau]\!]_{\Xi}\varphi(\Delta')$, $(k_1, k_2) \in \mathit{cont}([\![\tau]\!]_{\Xi}\varphi)(\Delta')$ and $(s_1, s_2) \in \mathit{states}(\Delta')$; our proof obligation now is

$$(\mathit{lookup}\, v_1\, k_1\, s_1, \mathit{lookup}\, v_2\, k_2\, s_2) \in R_{\mathit{Ans}}.$$

We branch on the possible values of v_1 and v_2 according to the definition of $[\![\texttt{ref}\ \tau]\!]_\Xi\varphi(\Delta')$. The first possibility is that there are l_1 and l_2 in *Loc* such that $v_1 = \lambda_{l_1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$ and $(s_1(l_1), s_2(l_2)) \in [\![\tau]\!]_\Xi\varphi(\Delta')$. But in that case we have

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) = (k_1\, s_1(l_1)\, s_1, k_2\, s_2(l_2)\, s_2) \in R_{Ans}$$

and are done.

In the second possible branch there are $n \in \omega$ and l_1 and l_2 in *Loc* such that $v_1 = \lambda_{l_1}^{n+1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$. Furthermore, we know that if $\pi_n(s_1) = \lfloor s_1' \rfloor$ then $(s_1'(l_1), s_2(l_2)) \in [\![\tau]\!]_\Xi\varphi(\Delta')$. If now $\pi_n(s_1) = \bot$ we get

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) = (\bot, lookup\, v_2\, k_2\, s_2) \in R_{Ans},$$

by the definition of $lookup : V \to TV$. On the other hand, $\pi_n(s_1) = \lfloor s_1' \rfloor$ gives us that

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) = (k_1\, s_1'(l_1)\, s_1, k_2\, s_2(l_2)\, s_2) \in R_{Ans}.$$

The Case of Assignment

We now turn to assignment. Assume that we have $\Xi \mid \Gamma \vdash t_1 \sim t_2 : \texttt{ref}\ \tau$ and $\Xi \mid \Gamma \vdash u_1 \sim u_2 : \tau$, we must prove that $\Xi \mid \Gamma \vdash t_1 := u_1 \sim t_2 := u_2 : 1$. Take $\varphi \in \mathcal{T}^\Xi$, $\Delta \in \mathcal{W}$ and $(\rho_1, \rho_2) \in [\![\Gamma]\!]_\Xi\varphi(\Delta)$ and aim to show that

$$([\![t_1 := u_1]\!]_X\rho_1, [\![t_2 := u_2]\!]_X\rho_2) \in comp([\![1]\!]_\Xi\varphi)(\Delta),$$

where we for brevity write X for $\mathrm{dom}(\Gamma)$. As was the case for lookup, we proceed by recalling the interpretation of the terms; we have that

$$[\![t_1 := u_1]\!]_X\rho_1 = [\![t_1]\!]_X\rho_1 \star \lambda v_1.\, [\![u_1]\!]_X\rho_1 \star \lambda w_1.\, assign\, v_1\, w_1$$

and similarly that

$$[\![t_2 := u_2]\!]_X\rho_2 = [\![t_2]\!]_X\rho_2 \star \lambda v_2.\, [\![u_2]\!]_X\rho_2 \star \lambda w_2.\, assign\, v_2\, w_2.$$

By an application of Lemma 9.2 in conjunction with the first assumption of this case we need to prove only that

$$(\lambda v_1.\, [\![u_1]\!]_X\rho_1 \star \lambda w_1.\, assign\, v_1\, w_1, \lambda v_2.\, [\![u_2]\!]_X\rho_2 \star \lambda w_2.\, assign\, v_2\, w_2)$$

is a member of $[\![\texttt{ref}\ \tau]\!]_\Xi\varphi \to_\Delta [\![1]\!]_\Xi\varphi$. Take $\Delta' \sqsupseteq \Delta$, $(v_1, v_2) \in [\![\texttt{ref}\ \tau]\!]_\Xi\varphi(\Delta')$ and apply Lemma 9.2 with the second assumption of this case to arrive at the proof obligation

$$(\lambda w_1.\, assign\, v_1\, w_1, \lambda w_2.\, assign\, v_2\, w_2) \in [\![\tau]\!]_\Xi\varphi \to_{\Delta'} [\![1]\!]_\Xi\varphi.$$

We pick $\Delta'' \sqsupseteq \Delta'$ and $(w_1, w_2) \in [\![\tau]\!]_\Xi\varphi(\Delta'')$, $(k_1, k_2) \in cont([\![1]\!]_\Xi\varphi)(\Delta'')$ and $(s_1, s_2) \in states(\Delta'')$ and arrive – finally – at the core of this case, as we plan to show

$$(assign\, v_1\, w_1\, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) \in R_{Ans}.$$

As above, we branch on the possible values of v_1 and v_2 according to the definition of $[\![\texttt{ref}\ \tau]\!]_\Xi\varphi(\Delta')$. The first possibility is that there are l_1 and l_2 in Loc such that $v_1 = \lambda_{l_1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$ and $(s_1[l_1 \mapsto w_1], s_2[l_2 \mapsto w_2]) \in states(\Delta'')$. This means that

$$(assign\, v_1\, w_1, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) = \\ (k_1\, (in_1 *)\, s_1[l_1 \mapsto w_1], k_2\, (in_1 *)\, s_2[l_2 \mapsto w_2]).$$

and this branch is done.

The second possibility is that there are $n \in \omega$ and l_1 and l_2 in Loc such that $v_1 = \lambda_{l_1}^{n+1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$. Furthermore, if $\pi_n(s_1) = \lfloor s_1' \rfloor$ we have that $\pi_n(w_1) = \lfloor w_1' \rfloor$ means that we have $(s_1'[l_1 \mapsto w_1'], s_2[l_2 \mapsto w_2]) \in states(\Delta'')$. If either $\pi_n(s_1) = \bot$ or $\pi_n(w_1) = \bot$ we get that

$$(assign\, v_1\, w_1, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) = (\bot, assign\, v_2\, w_2\, k_2\, s_2) \in R_{Ans}$$

by the definition of $assign : V \to V \to TV$. Otherwise we get $\pi_n(s_1) = \lfloor s_1' \rfloor$ and $\pi_n(w_1) = \lfloor w_1' \rfloor$ for some $s_1' \in S$ and $w_1' \in V$. And this buys us

$$(assign\, v_1\, w_1, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) = \\ (k_1\, (in_1 *)\, s_1'[l_1 \mapsto w_1'], k_2\, (in_1 *)\, s_2[l_2 \mapsto w_2])$$

which is an element of R_{Ans}.

The Case of Allocation

We will now go into the allocation of new references. Assume that we have $\Xi \mid \Gamma \vdash t_1 \sim t_2 : \tau$, we must prove $\Xi \mid \Gamma \vdash \texttt{ref}\ t_1 \sim \texttt{ref}\ t_2 : \texttt{ref}\ \tau$. We

make the canonical choices of $\varphi \in \mathcal{T}^{\Xi}$, $\Delta \in \mathcal{W}$ and $(\rho_1, \rho_2) \in [\![\Gamma]\!]_{\Xi}\varphi(\Delta)$ and proceed to show

$$([\![\texttt{ref}\ t_1]\!]_X\rho_1, [\![\texttt{ref}\ t_2]\!]_X\rho_2) \in \mathit{comp}([\![\texttt{ref}\ \tau]\!]_{\Xi}\varphi)(\Delta),$$

where we, as usual, write X for $\mathrm{dom}(\Gamma)$. Now, we have by definition that

$$([\![\texttt{ref}\ t_1]\!]_X\rho_1, [\![\texttt{ref}\ t_2]\!]_X\rho_2) = ([\![t_1]\!]_X\rho_1 \star \lambda v_1.\, \mathit{alloc}\, v_1, [\![t_2]\!]_X\rho_2 \star \lambda v_2.\, \mathit{alloc}\, v_2)$$

and so we apply the assumption of the case together with Lemma 9.2, pick $\Delta' \sqsupseteq \Delta$, $(v_1, v_2) \in [\![\tau]\!]_{\Xi}\varphi(\Delta')$, $(k_1, k_2) \in \mathit{cont}([\![\texttt{ref}\ \tau]\!]_{\Xi}\varphi)(\Delta')$ and $(s_1, s_2) \in \mathit{states}(\Delta')$ and are now down to proving

$$(\mathit{alloc}\, v_1\, k_1\, s_1, \mathit{alloc}\, v_2\, k_2\, s_2) \in R_{Ans}.$$

As a first step, we rewrite the above pair according to the definition of $\mathit{alloc} : V \to TV$ to get

$$(\mathit{alloc}\, v_1\, k_1\, s_1, \mathit{alloc}\, v_2\, k_2\, s_2) = (k_1\, \lambda_{l_1}\, s_1[l_1 \mapsto v_1], k_2\, \lambda_{l_2}\, s_2[l_2 \mapsto v_2])$$

where $l_1 \in \mathit{Loc}$ is the least with $l_1 \notin \mathrm{dom}(s_1)$ and $l_2 \in \mathit{Loc}$ is the least with $l_2 \notin \mathrm{dom}(s_2)$. As we have allocated new locations we should extend the world correspondingly. We define for each $\hat{\Delta} \in \hat{\mathcal{W}}$ a relation $\Phi(\hat{\Delta})$ on S by

$$\{(s_1, s_2) \mid l_1 \in \mathrm{dom}(s_1) \wedge l_2 \in \mathrm{dom}(s_2) \wedge (s_1(l_1), s_2(l_2)) \in [\![\tau]\!]_{\Xi}\varphi(Sq^{-1}(\hat{\Delta}))\}$$

and remark that $\Phi : \hat{\mathcal{W}} \to \mathit{BohrRel}(S)$ is well-defined, monotone and non-expansive. But then $\Theta = \{\emptyset, \{\emptyset\}, \lambda\_.\, \Phi\}$ easily is an island, i.e., a member of $I(\hat{\mathcal{W}})$. We define $\Delta'' \in \mathcal{W}$ by

$$\Delta''.\varsigma_1 = \Delta'.\varsigma_1 \cup \{l_1\}, \Delta''.\varsigma_2 = \Delta'.\varsigma_2 \cup \{l_2\}, \Delta''.\mathcal{I} = \Delta'.\mathcal{I}[n \mapsto \Theta]$$

where $n \in \omega$ is the least with $n \notin \mathrm{dom}(\Delta'.\mathcal{I})$. It is immediate by definition that $\Delta'' \sqsupseteq \Delta'$ and so it remains to prove that $(\lambda_{l_1}, \lambda_{l_2}) \in [\![\texttt{ref}\ \tau]\!]_{\Xi}\varphi(\Delta'')$ and that $(s_1[l_1 \mapsto v_1], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta'')$.

Addressing the first issue, take $\Delta''' \sqsupseteq \Delta''$, we have $l_1 \in \Delta''.\varsigma_1 \subseteq \Delta'''.\varsigma_1$ and $l_2 \in \Delta''.\varsigma_2 \subseteq \Delta'''.\varsigma_2$ by definition of world extension. Assume now that we have $(q_1, q_2) \in \mathit{states}(\Delta''')$. This would imply the existence of subheaps $q_1' \subseteq q_1$ and $q_2' \subseteq q_2$ with $(q_1', q_2') \in \Phi(Sq(\Delta'''))$, also by the definition of world extension. This means that $l_1 \in \mathrm{dom}(q_1')$, $l_2 \in \mathrm{dom}(q_2')$ and that $(q_1(l_1), q_2(l_2)) = (q_1'(l_1), q_2'(l_2)) \in [\![\tau]\!]_{\Xi}\varphi(\Delta''')$. And if we pick

$(w_1, w_2) \in [\![\tau]\!]_\Xi\varphi(\Delta''')$ then we have $(q_1'[l_1 \mapsto w_1], q_2'[l_2 \mapsto w_2]) \in \Phi(Sq(\Delta'''))$ and hence $(q_1[l_1 \mapsto w_1], q_2[l_2 \mapsto w_2]) \in \mathit{states}(\Delta''')$. In conclusion, $(\lambda_{l_1}, \lambda_{l_2}) \in [\![\texttt{ref}\ \tau]\!]_\Xi\varphi(\Delta'')$. Showing that $(s_1[l_1 \mapsto v_1], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta'')$ holds is not hard as we recall that $(s_1, s_2) \in \mathit{states}(\Delta')$ and that for any $m \in \mathrm{dom}(\Delta'.\mathcal{I})$ we have $\Delta''.\mathcal{I}(m) = \Delta'.\mathcal{I}(m)$, but do notice that this where we crucially rely on monotonicity of types and of the heap law of an island. □

10. Examples

10.1. Syntactic Sugar: Existential Types

Our language has universal types with associated term constructs but does not, a priori, come with existential types. But we can apply the standard encoding of existential types as universal types [25, Section 24.3] as follows:

Definition 10.1. *We write $\exists\alpha.\,\tau$ for the type $\forall\beta.\,(\forall\alpha.\,\tau \to \beta) \to \beta$ where β is not in τ. And we write $\texttt{pack}\ \sigma, t$ for the term $\Lambda\beta.\,\lambda f.\,f\,[\sigma]\,t$.*

It is easy to show that

$$\frac{\Xi \vdash \sigma \qquad \Xi \mid \Gamma \vdash t : \tau[\sigma/\alpha]}{\Xi \mid \Gamma \vdash \texttt{pack}\ \sigma, t : \exists\alpha.\,\tau}$$

is a derived typing rule. We do not need to unpack existential packages in the examples to come, but this could be encoded too. Instead we provide the following semantic lemma that is both useful and reassuring:

Lemma 10.2. *Define $in_\exists : TV \to V$ by $in_\exists(c) = in_\forall(\eta(in_\to(\psi_c)))$ where*

$$\psi_c = \lambda u.\,\eta(u) \star \lambda v.\,\begin{cases} d & v = in_\forall(d) \\ \mathit{error} & \text{otherwise} \end{cases} \star \lambda w.\,c \star \lambda x.\,\begin{cases} f\,x & w = in_\to(f) \\ \mathit{error} & \text{otherwise} \end{cases}.$$

We then have for $\Xi, \alpha \vdash \tau$, $\varphi \in \mathcal{T}^\Xi$, $\Delta \in \mathcal{W}$ and $c_1, c_2 \in TV$ that

$$\big(\exists\nu \in \mathcal{T}.\,\forall\Delta' \sqsupseteq \Delta.\,(c_1, c_2) \in [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu](\Delta')\big) \implies \big((in_\exists(c_1), in_\exists(c')) \in [\![\exists\alpha.\,\tau]\!]_\Xi\varphi(\Delta)\big).$$

Notice here the similarity with the interpretation of types, only the quantification is different. And that we cannot reason both ways; we do not know whether the reverse implication holds. The map $in_\exists : TV \to V$ was constructed by unrolling the interpretation of $\texttt{pack}\ \sigma, t$ to the point where no syntax was left; indeed, we have $[\![\texttt{pack}\ \sigma, t]\!]_X\rho = \eta(in_\exists([\![t]\!]_X\rho))$ whenever all term variables of t are in X.

10.2. More Sugar: Let Bindings and Sequencing

Definition 10.3. *For terms s and t and a variable x we write* $\texttt{let}\ x = s\ \texttt{in}\ t$ *for the term* $(\lambda x.\, t)\, s$. *For terms s and t we write $s;t$ for* $\texttt{let}\ x = s\ \texttt{in}\ t$ *where x is not in t.*

We have the obvious derived typing rules

$$\frac{\Xi \mid \Gamma \vdash s : \tau \qquad \Xi \mid \Gamma, x : \tau \vdash t : \sigma}{\Xi \mid \Gamma \vdash \texttt{let}\ x = s\ \texttt{in}\ t : \sigma} \qquad \frac{\Xi \mid \Gamma \vdash s : \tau \qquad \Xi \mid \Gamma \vdash t : \sigma}{\Xi \mid \Gamma \vdash s; t : \sigma}$$

and by using the convenient fact that we have $\eta(v) \star f = f(v)$ for any $v \in V$ and $f \in V \to TV$ we easily have the following lemma:

Lemma 10.4. *We have that* $[\![\texttt{let}\ x = s\ \texttt{in}\ t]\!]_X \rho = [\![s]\!]_X \rho \star \lambda v.\, [\![t]\!]_{X,x} \rho[x \mapsto v]$ *and that* $[\![s;t]\!]_X \rho = [\![s]\!]_X \rho \star \lambda\_.\, [\![t]\!]_X \rho$.

10.3. Booleans

We need the type `bool` of booleans in the example to come. Abbreviate

$$\begin{aligned} \texttt{bool} &= 1 + 1 \\ \texttt{true} &= \mathsf{inl}\,() \\ \texttt{false} &= \mathsf{inr}\,() \end{aligned}$$

We also introduce some convenient notation on the semantic side: Let $\mathbb{B} = \{0, 1\}$ be the discrete two-point predomain, and define $in_{\mathbb{B}} : \mathbb{B} \to V$ by $in_{\mathbb{B}}(1) = in_+(\iota_1(*))$ and $in_{\mathbb{B}}(0) = in_+(\iota_2(*))$. Then $[\![\texttt{true}]\!]_X \rho = \eta(in_{\mathbb{B}}\, 1)$ and $[\![\texttt{false}]\!]_X \rho = \eta(in_{\mathbb{B}}\, 0)$.

It is furthermore convenient to add an integer comparison operator $t_1 \leq t_2$ to the language. It has the following typing rule and semantics:

$$\frac{\Xi \mid \Gamma \vdash t_1 : \texttt{int} \qquad \Xi \mid \Gamma \vdash t_2 : \texttt{int}}{\Xi \mid \Gamma \vdash t_1 \leq t_2 : \texttt{bool}}$$

$$[\![t_1 \leq t_2]\!]_X \rho = [\![t_1]\!]_X \rho \star \lambda v_1.\, [\![t_2]\!]_X \rho \star \lambda v_2. \begin{cases} \eta(in_{\mathbb{B}}\, 1) & v_1 = in_{\mathbb{Z}}\, n,\ v_2 = in_{\mathbb{Z}}\, m,\ n \leq m \\ \eta(in_{\mathbb{B}}\, 0) & v_1 = in_{\mathbb{Z}}\, n,\ v_2 = in_{\mathbb{Z}}\, m,\ n > m \\ \mathit{error} & \text{otherwise.} \end{cases}$$

(One can encode this operator using ifz and fix, but the encoding is fairly complicated.)

10.4. Name Generator

Consider the program t_1 given by

$$t_1 = \texttt{let x} = \texttt{ref 0 in pack int}, (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{z} \leq !\,\texttt{x}).$$

It is not hard to assign it the type $\exists\alpha.\,(1 \to \alpha) \times (\alpha \to \texttt{bool})$. The idea is that of a name generator, each call to the first function returns a fresh name of type α by incrementing and then returning the value stored at location x. The second function is a sanity check, it asserts that a supplied value of type α is valid, i.e., does not exceed the largest name supplied so far. Put roughly, it can never return false because there is no way of producing stray values of α. And indeed, we shall prove e_1 contextually equivalent to the program t_2 given by

$$t_2 = \texttt{let x} = \texttt{ref 0 in pack int}, (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{true}),$$

where we have replaced the second function with a dummy that always returns true. The approach is, of course, to prove the interpretation of e_1 semantically related to the interpretation of e_2 at type $\exists\alpha.\,(1 \to \alpha) \times (\alpha \to \texttt{bool})$ and the other way round, we shall do only the first.

So, let us take on the task. We must show that $\models t_1 \sim t_2 : \exists\alpha.\,(1 \to \alpha) \times (\alpha \to \texttt{bool})$ where we note that both the type and term contexts are empty. This means picking $\Delta \in \mathcal{W}$ arbitrary, taking $(k_1, k_2) \in \mathit{cont}(\exists\alpha.\,(1 \to \alpha) \times (\alpha \to \texttt{bool}))(\Delta)$ and $(s_1, s_2) \in \mathit{states}(\Delta)$ and proving

$$(\llbracket t_1 \rrbracket\, k_1\, s_1, \llbracket t_2 \rrbracket\, k_2\, s_2) \in R_{Ans}.$$

A few calculations gives us that the left component $\llbracket t_1 \rrbracket\, k_1\, s_1$ equals

$$\llbracket \texttt{pack int}, (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{z} \leq !\,\texttt{x}) \rrbracket_x [x \mapsto \lambda_{l_1}]\, k_1\, s_1 [l_1 \mapsto 0]$$

where $l_1 \in \omega$ is the least such that $l_1 \notin \mathrm{dom}(s_1)$. Similarly we have that right component $\llbracket t_2 \rrbracket\, k_2\, s_2$ equals

$$\llbracket \texttt{pack int}, (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{true}) \rrbracket_x [x \mapsto \lambda_{l_2}]\, k_2\, s_2 [l_2 \mapsto 0]$$

where $l_2 \in \omega$ is the least such that $l_2 \notin \mathrm{dom}(s_2)$. Writing out a few more lines we arrive at

$$k_1\, \mathit{in}_{\exists}(\llbracket (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{z} \leq !\,\texttt{x}) \rrbracket_x \rho_1)\, s_1 [l_1 \mapsto 0],$$

and at

$$k_2\, in_\exists(\llbracket(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{true})\rrbracket_x \rho_2)\, s_2[l_2 \mapsto 0]$$

as our left and right hand side components, respectively. For brevity we write ρ_1 for $[x \mapsto \lambda_{l_1}]$ and ρ_2 for $[x \mapsto \lambda_{l_2}]$.

We are now, so to speak, at a point where allocation has been made by both programs and so we aim to extend the world to reflect this. First up, we define for each $n \in \omega$ a relation on S indexed by $\hat{\Delta} \in \hat{\mathcal{W}}$ as follows:

$$\Phi_n(\hat{\Delta}) = \{(s_1, s_2) \mid l_1 \in \mathrm{dom}(s_1) \wedge l_2 \in \mathrm{dom}(s_2) \wedge s_1(l_1) = s_2(l_2) = in_{\mathbb{Z}}\, n\}$$

It is easy to verify that $\Phi_n : \hat{\mathcal{W}} \to \mathit{BohrRel}(S)$ is well-defined and since it is constant it is monotone and non-expansive too. Let now $P_n = \{1, 2, \ldots, n\}$ for any $n \in \omega$, in particular we have $P_0 = \emptyset$. We then define

$$\Theta = \left(P_0, \{P_n \mid n \in \omega\}, \lambda X.\, \begin{cases} \Phi_n & X = P_n \\ \_ & \text{otherwise} \end{cases}\right)$$

and note that this is island, i.e., $\Theta \in I(\hat{\mathcal{W}})$. The population corresponds to the names generated so far; as the left and right name generators work in lock-step they always have the same set of generated names. Notice that it is initially empty because no names have been generated so far and that we restrict it to values from $\{P_0, P_1, \ldots\}$. The heap law just matches populations with the indexed relations on states; the definition requires us to define images of all of the subsets of V but we shall only ever need images of $\{P_0, P_1, \ldots\}$ and hence leave the remaining unspecified. We now define $\Delta' \in \mathcal{W}$ by

$$\Delta'.\varsigma_1 = \Delta.\varsigma_1 \cup \{l_1\}, \Delta'.\varsigma_2 = \Delta.\varsigma_2 \cup \{l_2\}, \Delta'.\mathcal{I} = \Delta.\mathcal{I}[n \mapsto \Theta]$$

where $n \in \omega$ is the least with $n \notin \mathrm{dom}(\Delta.\mathcal{I})$. It is immediate that $\Delta' \sqsupseteq \Delta$.

Having extended the world with an island that keeps track of the counters of both name generators we now build the type of generated names. These are exactly the population of the new island, so we just read them off; define a relation on V for $\Delta^* \in \mathcal{W}$ by

$$\nu(\Delta^*) = \begin{cases} \{(in_{\mathbb{Z}}\, v, in_{\mathbb{Z}}\, v) \mid v \in \Delta^*.\mathcal{I}(n).CP\} & n \in \mathrm{dom}(\Delta^*.\mathcal{I}) \wedge \\ & \Delta^*.\mathcal{I}(n).PL = \Theta.PL \\ \emptyset & \text{otherwise.} \end{cases}$$

We shall only apply this type to the world Δ' and possible extensions of this and so the second clause is really unreachable. But we cannot do without it, as the definition of $\mathcal{T}$ requires us to give values to all worlds. It is not hard to prove $\nu : \mathcal{W} \to BohrRel(V)$ well defined, non-expansive and monotone; we rely on the fact that island populations cannot shrink under world extension for the latter.

We now return to the issue at hand. As continuations are required to behave in future worlds and as $(s_1[l_1 \mapsto 0], s_2[l_2 \mapsto 0])$ easily is a member of $states(\Delta')$, it shall suffice to show that the pair

$$\big(in_\exists([\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + \mathtt{1}; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{z} \leq !\,\mathtt{x})]\!]_x \rho_1), \\ in_\exists([\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + \mathtt{1}; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{true})]\!]_x \rho_2)\big)$$

is a member of $[\![\exists \alpha.\, (1 \to \alpha) \times (\alpha \to \mathtt{bool})]\!](\Delta')$. Now take $\Delta'' \sqsupseteq \Delta'$ arbitrary, by Lemma 10.2 it shall suffice to show that

$$\big([\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + \mathtt{1}; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{z} \leq !\,\mathtt{x})]\!]_x \rho_1, [\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + \mathtt{1}; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{true})]\!]_x \rho_2\big)$$

is a member of $comp([\![(1 \to \alpha) \times (\alpha \to \mathtt{bool})]\!]_\alpha[\alpha \mapsto \nu])(\Delta'')$. This again comes down to the following two obligations:

1. Prove that $([\![\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + \mathtt{1}; !\,\mathtt{x}]\!]_x \rho_1, [\![\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + \mathtt{1}; !\,\mathtt{x}]\!]_x \rho_2)$ is a member of $comp([\![1 \to \alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta'')$.
2. Prove that for $\Delta''' \sqsupseteq \Delta''$ we have that $([\![\lambda \mathtt{z}.\, \mathtt{z} \leq !\,\mathtt{x}]\!]_x \rho_1, [\![\lambda \mathtt{z}.\, \mathtt{true}]\!]_x \rho_2)$ is a member of $comp([\![\alpha \to \mathtt{bool}]\!]_\alpha[\alpha \mapsto \nu])(\Delta''')$.

By inspection of proof obligation 1 we arrive at the following two sub-obligations that we must address:

1.a. Let $\Delta''' \sqsupseteq \Delta''$ be arbitrary. Prove that $([\![\mathtt{x} := !\,\mathtt{x} + \mathtt{1}]\!]_x \rho_1, [\![\mathtt{x} := !\,\mathtt{x} + \mathtt{1}]\!]_x \rho_2)$ is a member of $comp([\![1]\!]_\alpha[\alpha \mapsto \nu])(\Delta''')$.

1.b. Let $\Delta^\dagger \sqsupseteq \Delta'''$ be arbitrary. Prove that $([\![!\,\mathtt{x}]\!]_x \rho_1, [\![!\,\mathtt{x}]\!]_x \rho_2)$ is a member of $comp([\![\alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta^\dagger)$.

We now attack the sub-obligation 1.a head-on. Let $\Delta''' \sqsupseteq \Delta''$ be arbitrary. By definition of the untyped interpretation we derive that $[\![\mathtt{x} := !\,\mathtt{x} + \mathtt{1}]\!]_x \rho_1$ is

$$lookup\, \lambda_{l_1} \star \lambda v_1.\, \begin{cases} \eta(in_{\mathbb{Z}}(m+1)) & v_1 = in_{\mathbb{Z}}\, m \\ error & \text{otherwise} \end{cases} \star \lambda w_1.\, assign\, \lambda_{l_1}\, w_1$$

and the same for $[\![\mathtt{x} := !\,\mathtt{x} + \mathtt{1}]\!]_x\rho_2$, only exchange λ_{l_2} for λ_{l_1}. Take $(k_1, k_2) \in \mathit{cont}([\![1]\!]_\alpha[\alpha \mapsto \nu])(\Delta''')$ and $(t_1, t_2) \in \mathit{states}(\Delta''')$. Since $\Delta''' \sqsupseteq \Delta'$ we know that $\Delta'''.\mathcal{I}(n)$ is well-defined and equals Θ defined above, modulo a change of population. In particular there must be $m \in \omega$ such that $\Delta'''.\mathcal{I}(n).CP = P_m$ and we know that $l_1 \in \mathrm{dom}(t_1)$, $l_2 \in \mathrm{dom}(t_2)$ and that $t_1(l_1) = t_2(l_2) = \mathit{in}_{\mathbb{Z}}\, m$. Summing up we get

$$\begin{aligned}[\![\mathtt{x} := !\,\mathtt{x} + \mathtt{1}]\!]_x\rho_1\, k_1\, t_1 &= (\eta(\mathit{in}_{\mathbb{Z}}(m+1)) \star \lambda w_1.\, \mathit{assign}\, \lambda_{l_1}\, w_1)\, k_1\, t_1 \\ &= \mathit{assign}\, \lambda_{l_1}\, \mathit{in}_{\mathbb{Z}}(m+1)\, k_1\, t_1 \\ &= k_1\, \mathit{in}_1(*)\, t_1[l_1 \mapsto \mathit{in}_{\mathbb{Z}}(m+1)]\end{aligned}$$

and similarly that

$$[\![\mathtt{x} := !\,\mathtt{x} + \mathtt{1}]\!]_x\rho_2\, k_2\, t_2 = k_2\, \mathit{in}_2(*)\, t_2[l_2 \mapsto \mathit{in}_{\mathbb{Z}}(m+1)].$$

We now build $\Delta^\dagger$ as a copy of Δ''' with the one exception that $\Delta^\dagger.\mathcal{I}(n).CP = P_{m+1}$ which gives us $\Delta^\dagger \sqsupseteq \Delta'''$ and $(t_1[l_1 \mapsto \mathit{in}_{\mathbb{Z}}(m{+}1)], t_2[l_2 \mapsto \mathit{in}_{\mathbb{Z}}(m{+}1)]) \in \mathit{states}(\Delta^\dagger)$ and this sub-obligation is done. Note, amidst the technicalities, that we have just generated a new name $m+1$ and updated the population of island n correspondingly.

Sub-obligation 1.b is a bit shorter. Let $\Delta^\dagger \sqsupseteq \Delta'''$ be arbitrary. Take $(k_1, k_2) \in \mathit{cont}([\![\alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta^\dagger)$ and $(t_1, t_2) \in \mathit{states}(\Delta^\dagger)$. As above, there must be $m \in \omega$ such that $\Delta^\dagger.\mathcal{I}(n).CP = P_m$ and we know that $l_1 \in \mathrm{dom}(t_1)$, $l_2 \in \mathrm{dom}(t_2)$ and that $t_1(l_1) = t_2(l_2) = \mathit{in}_{\mathbb{Z}}\, m$. But then we get

$$([\![!\,\mathtt{x}]\!]_x\rho_1\, k_1\, t_1, [\![!\,\mathtt{x}]\!]_x\rho_2\, k_2\, t2) = (k_1\, (\mathit{in}_{\mathbb{Z}}\, m)\, t_1, k_2\, (\mathit{in}_{\mathbb{Z}}\, m)\, t_2).$$

And all we need to finish this sub-obligation is just to remark that

$$[\![\alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta^\dagger) = \nu(\Delta^\dagger) = \{(\mathit{in}_{\mathbb{Z}}\, v, \mathit{in}_{\mathbb{Z}}\, v) \mid v \in P_m\} \ni (\mathit{in}_{\mathbb{Z}}\, m, \mathit{in}_{\mathbb{Z}}\, m).$$

Finally we tackle obligation 1. Let $\Delta''' \sqsupseteq \Delta''$ be arbitrary. We can derive that $[\![\mathtt{z} \le !\,\mathtt{x}]\!]_{x,z}\rho_1[z \mapsto v_1]\, k_1\, t_1$ is

$$\mathit{lookup}\, \lambda_{l_1} \star \lambda w_1. \begin{cases} \eta(\mathit{in}_{\mathbb{B}}\, 1) & v_1 = \mathit{in}_{\mathbb{Z}}\, k, w_1 = \mathit{in}_{\mathbb{Z}}\, m, k \le m \\ \eta(\mathit{in}_{\mathbb{B}}\, 0) & v_1 = \mathit{in}_{\mathbb{Z}}\, k, w_1 = \mathit{in}_{\mathbb{Z}}\, m, k > m \\ \mathit{error} & \text{otherwise.} \end{cases}$$

Pick $\Delta^\dagger \sqsupseteq \Delta'''$, $(v_1, v_2) \in [\![\alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta^\dagger) = \nu(\Delta^\dagger)$, $(k_1, k_2) \in \mathit{cont}([\![\mathtt{bool}]\!]_\alpha[\alpha \mapsto \nu](\Delta^\dagger)$ and $(t_1, t_2) \in \mathit{states}(\Delta^\dagger)$, we must show that

$$([\![\mathtt{z} \le !\,\mathtt{x}]\!]_{x,z}\rho_1[z \mapsto v_1]\, k_1\, t_1, [\![\mathtt{true}]\!]_{x,z}\rho_2[z \mapsto v_2]\, k_2\, t_2) \in R_{Ans}.$$

As above, there must be $m \in \omega$ such that $\Delta^\dagger.\mathcal{I}(n).CP = P_m$ and we know that $l_1 \in \mathrm{dom}(t_1)$, $l_2 \in \mathrm{dom}(t_2)$ and that $t_1(l_1) = t_2(l_2) = in_{\mathbb{Z}}\, m$. This also means that $(v_1, v_2) = (in_{\mathbb{Z}}\, k, in_{\mathbb{Z}}\, k)$ for some $1 \leq k \leq m$. Combining our efforts we arrive at

$$[\![\mathtt{z} \leq !\,\mathtt{x}]\!]_{x,z}\rho_1[z \mapsto v_1]\, k_1\, t_1 = k_1(in_{\mathbb{B}}\, 1)\, t_1$$

and as we immediately have

$$[\![\mathtt{true}]\!]_{x,z}\rho_2[z \mapsto v_2]\, k_2\, t_2 = k_2\, (in_{\mathbb{B}} 1)\, t_2$$

we are done. Taking a few steps back, all that happens is that the interpretation of the type $[\![\alpha]\!]_\alpha[\alpha \mapsto \nu] = \nu$ ensures that the input values must be in the population of island n; also the heap law enforces that the related states both contain the maximum name of the population at location l_1 and l_2 respectively.

10.5. Discussion

We have written out the proof of the Name Generator example in much detail so as to make it easy to compare this proof with the one in the ADR model [3]. Looking at the two proofs we can conclude (as claimed in the introduction) that the semantic techniques from the BST model scale to state-of-the-art world descriptions and that the resulting model can be used to prove programs equivalent at a fairly abstract level, without any form of low-level step-indexed reasoning. Indeed the proof we have given here in the model is at an abstraction level which is similar to the one provided by the LADR logic [22, Pages 56–58].

The same is the case for the other examples involving local state in (L)ADR.[4] The model similarly gives rise to fairly abstract proofs of the Plotkin-Power axioms for global state and local state [29] as formulated by Staton [33].[5]

For proving some equivalences of programs involving recursive types and/or reference types, LADR uses a so-called "later" modality and Löb's rule to abstractly account for induction over step-indices (This idea comes from [7].)

[4]LADR cannot handle the "callback-with-lock" example of ADR and the same is the case with our model here (see [21] for further discussion).

[5]Three of the axioms, GS6, GS7, and B3, cannot be formulated as simple, typed equations in our language, but equivalent semantic formulations do hold in the model.

For example, the later modality and Löb's rule are used to prove that Landin's knot—the construction of a fixed-point using backpatching—works [21, Section 9.3]. In the present model, this example would instead proceed via fixed-point induction.

Note that the proof of the name generator involves references and thus locations, but that there are no approximate locations in the proof at all. This is typical of examples that involve allocation and uses of references (such as the (L)ADR examples); approximate locations do not appear in the proofs since they are not used as denotations of references allocated in the programs. But, of course, there are examples, where they show up. Indeed, when one considers examples involving free variables of reference type (or, equivalently, functions that take arguments of reference type), then one also has to consider approximate locations. These approximate locations are in some sense additional junk in the denotational model, and because of them there are equivalences that we cannot prove using the model. The simplest concrete example we know of is the following [12]:

$$\emptyset \mid \emptyset \models \lambda x.\mathsf{true} \not\sim \lambda x.\mathsf{false} : \mathsf{ref}\, 0 \to \mathsf{bool}$$

Intuitively, these two functions should be contextually equivalent: since references are initialized when allocated, no closed value encountered in a running program can ever have the type $\mathsf{ref}\, 0$, and therefore neither of the two functions can ever be applied. However, the two functions are not semantically related in our model. Loosely speaking, the reason is that approximate locations can be related at the type $\mathsf{ref}\, 0$.

We leave it as future work to investigate further if one can find a more abstract model, which does not involve either a form of semantic location or some form of step-indexing. We believe this is a challenging problem — for an earlier version of the BST model we could show that a putative logical relation formulated without approximate locations did not exist! — and it is related to questions of existence of recursively defined relations in [11].

Other future work includes the formulation of a program logic for reasoning about equivalence based on the present model. Such a logic would naturally combine ideas from LADR concerning syntactic formulations of islands, etc., with ideas from earlier domain-theoretically inspired logics for call-by-value (see, e.g., [1]). In particular it would not include the later modality and the Löb rule of LADR but rather have a fixed point induction rule.

11. Acknowledgements

We would like to thank Derek Dreyer and Georg Neis for helpful discussions and insightful comments including, but not limited to, observations on the consequences of choosing either intensional or extensional interpretations of references.

References

[1] M. Abadi and M. P. Fiore. Syntactic considerations on recursive types. In *Proceedings of LICS*, pages 242–252. IEEE Computer Society, 1996.

[2] M. Abadi and G. D. Plotkin. A per model of polymorphism and recursive types. In *Proceedings of LICS*, pages 355–365. IEEE Computer Society, 1990.

[3] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *Proceedings of POPL*, pages 340–353. ACM Press, 2009.

[4] A. J. Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In *Proceedings of ESOP*, volume 3924 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2006.

[5] R. M. Amadio. Recursion over realizability structures. *Inf. Comput.*, 91 (1):55–85, 1991.

[6] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.*, 39(3): 343–375, 1989.

[7] A. Appel, P.-A. Melliès, C. Richards, and J. Vouillon. A very modal model of a modern, major, general type system. In *Proceedings of POPL*, pages 109–122. ACM Press, 2007.

[8] A. W. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *Transactions on Programming Languages and Systems*, 23(5):657–683, 2001.

[9] C. Baier and M. E. Majster-Cederbaum. The connection between initial and unique solutions of domain equations in the partial order and metric approach. *Formal Aspects of Computing*, 9(4):425–445, 1997.

[10] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *Proceedings of TLCA*, volume 3461 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2005.

[11] N. Benton, A. Kennedy, L. Beringer, and M. Hofmann. Relational semantics for effect-based program transformations: higher-order store. In *Proceedings of PPDP*, pages 301–312. ACM Press, 2009.

[12] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In *Proceedings of FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 456–470. Springer, 2009.

[13] L. Birkedal, K. Støvring, and J. Thamsborg. Relational parametricity for references and recursive types. In *Proceedings of TLDI*, pages 91–104. ACM Press, 2009.

[14] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations. Technical Report TR-2009-119, IT University of Copenhagen, August 2009.

[15] L. Birkedal, K. Støvring, and J. Thamsborg. Solutions of generalized recursive metric-space equations. In *Proceedings of the 6th Workshop on Fixed Points in Computer Science*, pages 18–24, 2009.

[16] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. Technical Report TR-2010-124, IT University of Copenhagen, January 2010.

[17] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Proceedings of APLAS*, volume 4279 of *Lecture Notes in Computer Science*, pages 79–96. Springer, 2006.

[18] J. de Bakker and E. de Vink. *Control flow semantics*. MIT Press, Cambridge, MA, USA, 1996.

[19] J. de Bakker and J. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[20] D. Dreyer, A. Ahmed, and L. Birkedal. Logical step-indexed logical relations. In *Proceedings of LICS*, pages 71–80. IEEE Computer Society, 2009.

[21] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In *Proceedings of POPL*, pages 185–198. ACM Press, 2010.

[22] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs (technical appendix). Available at `http://www.mpi-sws.org/~dreyer/papers/ladr/appendix.pdf`, 2010.

[23] P.-A. Melliès and J. Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *Proceedings of LICS*, pages 82–91. IEEE Computer Society, 2005.

[24] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[25] B. C. Pierce. *Types and programming languages*. MIT Press, Cambridge, MA, USA, 2002.

[26] A. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. MIT Press, 2005.

[27] A. M. Pitts. Relational properties of domains. *Inf. Comput.*, 127(2): 66–90, 1996.

[28] A. M. Pitts and I. D. B. Stark. Operational reasoning for functions with local state. In *Higher order operational techniques in semantics*, pages 227–274. Cambridge University Press, New York, NY, USA, 1998.

[29] G. D. Plotkin and J. Power. Notions of computation determine monads. In *Proceedings of FOSSACS*, volume 2303 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2002.

[30] J. C. Reynolds. Types, abstraction, and parametric polymorphism. In *Information Processing 83, Paris, France*, pages 513–523. Elsevier, 1983.

[31] M. B. Smyth. Topology. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[32] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.*, 11(4):761–783, 1982.

[33] S. Staton. Completeness for algebraic theories of local state. In *Proceedings of FOSSACS*, 2010. To appear. Available at `http://www.cl.cam.ac.uk/~ss368/fossacs10.pdf`.

[34] G. Winskel. *The formal semantics of programming languages: an introduction*. MIT Press, Cambridge, MA, USA, 1993.

Chapter 6

Kripke Models over Recursively Defined Metric Worlds: Steps and Domains

Kripke Models over Recursively Defined Metric Worlds: Steps and Domains

Lars Birkedal
IT University of Copenhagen
birkedal@itu.dk

Kristian Støvring
IT University of Copenhagen
kss@itu.dk

Jacob Thamsborg
IT University of Copenhagen
thamsborg@itu.dk

Abstract

We show that models of higher-order store phenomena naturally can be given as Kripke models over worlds that are recursively defined in a category of metric spaces. It leads to a unification of methods based on classical domain theory and on step-indexed operational models. We show that our metric approach covers a wide range of step-indexed models, by demonstrating how it can be specialized to Hobor et. al.'s recent indirection theory, and by developing a new step-indexed model of separation logic for higher-order store.

1 Introduction

Over the last decade, there has been a lot of research on semantic models for reasoning about advanced programming language features involving recursive structures arising from various forms of *higher-order store*, see, e.g. [12, 19, 23, 25, 29, 30]. Many proposed methods have been based on either traditional domain theory or on more recent step-indexed models [2–4, 8, 9]. In this paper, we argue that the essence of these models is that they can be seen as Kripke models over recursively defined sets of worlds. Indeed, we show how to define such worlds using appropriate recursively-defined metric spaces and, moreover, show how this method applies both to domain-theoretic models and to step-indexed models, thus achieving a unification of methods. In earlier work, we have used solutions to recursive metric-space equations in connection with domain-theoretic models [18], so in this paper, we focus mostly on step-indexed models. In particular, we show how our metric approach can be specialized to Hobor et. al.'s recent abstract description of step-indexed models [24] and argue why it is useful to take the metric viewpoint we suggest. The latter is done, in part, by presenting a step-indexed model of a separation logic for higher-order store [32], the soundness of which involves the use of a recursively defined operation on the recursively-defined set of worlds.

Higher-order Store We use "higher-order store" loosely to refer to programming language features that involve some form of dynamic allocation of data whose type / specification depends on the types / specifications of already stored data. Thus higher-order store can, e.g., describe the ability to dynamically allocate heap storage and store code directly in the heap; C function pointers; ML references; but also dynamically allocated locks that protect resource invariants that depend on already allocated locks' resource invariants. For expressiveness, type systems and logics for higher-order store often involve some form of recursive types / specifications, and also some form of universal quantification over types (impredicative polymorphism) or specifications. The semantic methods we present scale well to these features (see, e.g., [18]), but they are not at the heart of the challenge of modeling higher-order store, so we shall not dwell too long on them in this paper, but just sketch how the methods apply.

Semantic Models of What Semantic models of higher-order store can, among other things, be used to show soundness of type systems and logics for reasoning about programs. The latter can involve Hoare-style logics for reasoning about a single program or logics for relational reasoning about equality of programs. Our methodology applies to all of these, but relational reasoning involves a host of other mostly orthogonal issues, so we focus on models for type systems and logics for reasoning about a single program using unary predicates instead of relations, except for some discussion in the related work section.

2 Introductory Example: ML references

By way of introduction of our general setup, let us consider how to model a programming language with impredicative polymorphism and general ML-like references; i.e., an extension of the polymorphic lambda calculus with a standard call-by-value operational semantics. We first describe the general idea at an intuitive level and then, in the following two subsections, we explain how to realize the

general idea in a domain-theoretic setting (based on an adequate denotational semantics of the programming langauge) and in a step-indexed stetting (based purely on the operational semantics of the programming language).

Recall that for the polymorphic lambda calculus, without general references, we can model types as predicates (subsets) on some fixed set of values. But since our language of interest now includes dynamic allocation, it is natural to follow earlier work on models of languages with dynamic allocation of simple integer cells (e.g., [11, 25]), and use a Kripke-style possible-worlds model. Here, however, the set of worlds $\mathcal{W}$ needs to be recursively defined since we consider general references: Semantically, a world maps locations (modeled as natural numbers) to semantic types in $\mathcal{T}$, and we thus arrive at the following recursive equations:

$$\begin{array}{rcl} V & = & \text{set of values, including locations} \\ \mathcal{W} & = & \mathbb{N} \rightharpoonup_{fin} \mathcal{T} \\ \mathcal{T} & = & \mathcal{W} \to \mathit{Pred}(V) \end{array}$$

With such a semantic model of types, one can give meaning to types in $\mathcal{T}$, in particular, the meaning of a reference type $\mathsf{ref}\,\tau$ can be defined roughly as

$$(\mathsf{ref}\,\tau)w = \{l \mid w(l) = \tau\},$$

i.e, for a world w, it is the set of locations l such that the semantic type recorded in the world at l is the same as τ.

Some readers might have expected that semantic types would be monotone wrt. an extension ordering of worlds. Indeed, it is often advantageous to build monotonicity into the model of types, but since this issue is mostly orthogonal to the point we are trying to make now, we will omit discussion of monotonicity until Section 3.2.

Observe that the natural model of types here is a *Kripke model over a recursively-defined set of worlds*. It is a Kripke model because the semantic types are parameterized over worlds. The problem is, of course, that, for cardinality reasons, there is no solution to the above equations in the category of sets (unfolding the above equation we get $\mathcal{W} = \mathbb{N} \rightharpoonup_{fin} (\mathcal{W} \to \mathit{Pred}(V))$ with $\mathcal{W}$ in a negative position, see also [3]).

This observation leads Hobor et. al. [24] to propose that we should give up solving the equation and instead use an approximate solution, where the equations are not solved up to isomorphism, but where one only finds a retraction between two sets, equipped with some additional approximation information akin to the indices used in step-indexed models. As Hobor et. al. show, this suffices for many step-indexed models.

Instead, we here propose to solve the equation in a certain simple category of metric spaces. Our approach applies not only to step-indexed models but also to domain-theoretic models; it can be specialized to Hobor et. al.'s indirection theory (see Section 3.1), and has a number of other advantages that we shall explain in Section 3.2. In the following two subsections we exemplify the approach in a domain-theoretic setting and in step-indexed setting, but let us first call to mind some facts about the metric spaces we are going to use.

Recap of ultrametric spaces A 1-bounded ultrametric space (X, d) is a metric space where the distance function $d : X \times X \to \mathbb{R}$ takes values in the closed interval $[0, 1]$ and satisfies the strong triangle inequality $d(x, y) \leq \max\{d(x, z), d(z, y)\}$, for $x, y, z \in X$. An (ultra-)metric space is complete if every Cauchy sequence has a limit. A function $f : X_1 \to X_2$ between metric spaces $(X_1, d_1), (X_2, d_2)$ is *non-expansive* if for all $x, y \in X_1$, $d_2(f(x), f(y)) \leq d_1(x, y)$, i.e., if application does not increase the distance between points. It is *contractive* if for some $\delta < 1$, $d_2(f(x), f(y)) \leq \delta \cdot d_1(x, y)$ for all $x, y \in X_1$.

The complete, 1-bounded, non-empty, ultrametric spaces and non-expansive functions between them form a Cartesian closed category $\mathsf{CBUlt}_{\mathsf{ne}}$. Products in $\mathsf{CBUlt}_{\mathsf{ne}}$ are given by the set-theoretic product where the distance is the maximum of the componentwise distances, and exponentials are given by the non-expansive functions equipped with the sup-metric. (i.e., the exponential $(X_1, d_1) \to (X_2, d_2)$ has the set of non-expansive functions from (X_1, d_1) to (X_2, d_2) as underlying set, and distance function: $d_{X_1 \to X_2}(f, g) = \sup\{d_2(f(x), g(x)) \mid x \in X_1\}$). For any set S and space $(X, d) \in \mathsf{CBUlt}_{\mathsf{ne}}$, the set of finite partial functions $S \rightharpoonup_{fin} X$ from S to X is again a complete bounded ultrametric space with distance function given by $d(f, g) = 1$, if the domain of f and g are not equal, and $d(f, g) = \max\{d(f(s), g(s)) \mid s \in \mathrm{dom}(f)\}$, if the domain of f and g are equal.

A functor $F : \mathsf{CBUlt}_{\mathsf{ne}}{}^{\mathrm{op}} \times \mathsf{CBUlt}_{\mathsf{ne}} \longrightarrow \mathsf{CBUlt}_{\mathsf{ne}}$ is *locally non-expansive* if $d(F(f, g), F(f', g')) \leq \max\{d(f, f'), d(g, g')\}$ for all non-expansive f, f', g, g', and it is *locally contractive* if $d(F(f, g), F(f', g')) \leq \delta \cdot \max\{d(f, f'), d(g, g')\}$ for some $\delta < 1$. By multiplication of the distances of (X, d) with a shrinking factor $\delta < 1$ one obtains a new ultrametric space, $\delta \cdot (X, d) = (X, d')$ where $d'(x, y) = \delta \cdot d(x, y)$. By shrinking, a locally non-expansive functor F yields a locally contractive functor $(\delta \cdot F)(X_1, X_2) = \delta \cdot (F(X_1, X_2))$. For a less condensed introduction to ultrametric spaces we refer to [33].

It is well-known that one can solve recursive domain-equations in $\mathsf{CBUlt}_{\mathsf{ne}}$, by an adaptation of the inverse-limit method from classical domain theory:

Theorem 2.1 (America-Rutten [7])**.** Let $F : \mathsf{CBUlt}_{\mathsf{ne}}{}^{\mathrm{op}} \times \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ be a locally contractive functor. Then there exists a unique (up to isomorphism) $(X, d) \in \mathsf{CBUlt}_{\mathsf{ne}}$ such that $F((X, d), (X, d)) \cong (X, d)$.

2.1 Domain-Theoretic Model

In [18] we gave a relationally parametric domain-theoretic model of a call-by-value language with impredicative polymorphism, general references, and recursive types. We now explain how it fits the intuitive model from above.

The model is based on an adequate domain-theoretic "untyped" model of the programming language that is defined in a mostly standard way, using a recursively defined predomain (complete partial order) V of values. The predomain V comes equipped with a family of projections $\pi_n : V \to V_\perp$, satisfying the usual conditions for projections arising from solutions to recursive domain equations. In particular, the minimal invariance property: the least upper bound of the projections $\bigsqcup_n \pi_n$ is the identity on V.

For the modeling of types, we use not all predicates on V, but only those that are *complete* (admissible), i.e., closed under least upper bounds of chains, and *uniform*. A subset P of V is *uniform* if it is closed under all the projections, i.e., if $\forall v \in P. \forall n. \pi_n(v) \in P_\perp$. For subsets P and Q of V, we write $\pi_n : P \to Q$ to mean that $\forall v \in P.\, \pi_n(v) \in Q_\perp$. It is well-known from earlier work on interpreting recursive types and impredicative polymorphism [1, 5, 6, 20, 26] that the set $CUPred(V)$ of all complete uniform predicates on V form a complete 1-bounded ultrametric space. The distance function d is defined by

$$d(P,Q) = \begin{cases} 2^{-\max\{\, n \in \mathbb{N} \mid \pi_n \in P \to Q \,\wedge\, \pi_n \in Q \to P \,\}} & \text{if } P \neq Q \\ 0 & \text{if } P = Q. \end{cases}$$

The distance is well-defined by properties of the projection functions, in particular the minimal invariance property.

It is easy to see that the functor $X \mapsto \mathbb{N} \multimap_{fin} \frac{1}{2}(X \to CUPred(V))$ from $\mathsf{CBUlt_{ne}}^{op}$ to $\mathsf{CBUlt_{ne}}$ is locally contractive and thus, by Theorem 2.1, there exists a complete 1-bounded ultrametric space $\mathcal{W}$ satisfying:

$$\mathcal{W} \cong \mathbb{N} \multimap_{fin} \tfrac{1}{2}(\mathcal{W} \to CUPred(V)) \quad \text{in } \mathsf{CBUlt_{ne}}.$$

Note that the rightmost function-space arrow in the display above denotes the function space in $\mathsf{CBUlt_{ne}}$, i.e., the set of non-expansive functions. The $\frac{1}{2}$ is an example of a shrinking factor and, technically, ensures that the functor is locally contractive; it is a standard technique [7]. The intuitive reason for why it is ok to use the $\frac{1}{2}$ shrinking factor is that "it takes a computation step to dereference a location" (in [18] this is modelled via so-called semantic locations).

Having now succeeded in establishing the existence of the recursively defined set of worlds $\mathcal{W}$, we can define semantic types $\mathcal{T}$ to be the set of non-expansive functions from $\mathcal{W}$ to $CUPred(V)$. With this semantic model of types, we can give an interpretation of all the types of the programming language. (For recursive types, we employ Banach's fixed point theorem to find a solution as the fixed point of a contractive operator on $\mathcal{T}$.) Finally, we can define the typed meaning of terms by proving the fundamental theorem of logical relations wrt. the untyped semantics of terms. See [18] for a detailed treatment.

2.2 Step-Indexed Model

Now suppose that we want to build a semantic model over the operational semantics directly, without passing through a domain-theoretic model of the programming language. Then we can let V be the set of closed syntactic values v used in the operational semantics (i.e., v can be a pair of syntactic values v_1 and v_2, a syntactic lambda abstraction, etc.), and, in keeping with the ideas of step-indexed models [2, 3, 8, 9], we can model types as (world-indexed) subsets of $\mathbb{N} \times V$ that are downwards-closed in the step ($\mathbb{N}$) component. More precisely, we define $UPred(V)$ to be

$$\{P \subseteq \mathbb{N} \times V \mid \forall (k,v) \in P. \forall j \leq k. (j,v) \in P\}.$$

We can define a distance function on $UPred(V)$, which measures "up-to-what-level" two uniform predicates agree, as follows: First, for $P \in UPred(V)$, let $\overline{P}^k$ denote $\{(m,v) \in P \mid m < k\}$, and then define distance function d by:

$$d(P,Q) = \begin{cases} 2^{-n} & \text{if } P \neq Q \text{ and n = } \max\{k \mid \overline{P}^k = \overline{Q}^k\} \\ 0 & \text{if } P = Q. \end{cases}$$

Lemma 2.2. $(UPred(V), d)$ is a well-defined object in $\mathsf{CBUlt_{ne}}$.

Thus, by an application of Theorem 2.1, there exists a complete 1-bounded ultrametric space $\mathcal{W}$ satisfying

$$\mathcal{W} \cong \mathbb{N} \multimap_{fin} \tfrac{1}{2}(\mathcal{W} \to UPred(V)) \quad \text{in } \mathsf{CBUlt_{ne}},$$

and we can then define semantic types $\mathcal{T}$ to be the set of non-expansive functions from $\mathcal{W}$ to $UPred(V)$.

Thus by working in $\mathsf{CBUlt_{ne}}$ we can indeed solve the wished-for equations, even in a setting based on operational semantics. With this semantic model of types, one can then define an interpretation of all the types of the programming language, with definitions similar to those used in existing step-indexed models [3], but knowing that one has a solution to the wished-for recursive equation of worlds. We show how this can be done in Appendix A. Again, the intuitive reason for why it is ok to use the $\frac{1}{2}$ shrinking factor is that "it takes a computation step to dereference a location".

We remark that it is not surprising that there is a connection between metric spaces and step-indexed models; this was already pointed out in [8]. The point is that it is useful not to forget this connection because it, e.g., allows us to define solutions to recursive world equations such as the one above. (See also the discussion in Section 3.2.)

We do not present a formal relationship to existing models for this particular example, but rather show, in the following section, how all the step-indexed models described via the indirection theory of Hobor et. al. can be obtained by a specialization of our general approach. In Section 3.2 we explain why it is useful to solve the world equation using metric spaces, and in Section 4 we present a new application.

3 The Essence of Step-Indexed Models

3.1 Specialization to Indirection Theory

Faced with a higher-order store recursive equation, Hobor, Dockins and Appel [24] provide an approximative solution. This is a section-retraction pair characterized by the two axioms of indirection theory that elegantly capture the approximative nature of the solution. Our approach is different, we solve the recursion proper in a certain category of metric spaces. In both cases, however, the solution provides a notion of worlds[1] to be used in Kripke models as exemplified amply in *loc.cit.* and in the previous section. In this section we shall argue that our approach is the more general in the sense that, for the same recursive equation, one may build the approximative solution of Hobor et. al. from our solution – this is Theorem 3.6. A consequence is that, somewhat indirectly, we have shown our method applicable to all examples considered by Hobor et. al.

This specialization to indirection theory is not unconditional. The construction presented by Hobor et. al. is parameterized over a set-theoretic functor $F : \mathsf{Set} \to \mathsf{Set}$ but our approach deals in metric-space equations phrased in terms of a locally non-expansive functor on $\mathsf{CBUlt}_{\mathsf{ne}}$. So we must have one of the latter corresponding to the former or, more precisely, we must have a *plain lift* of $F : \mathsf{Set} \to \mathsf{Set}$, as defined below, to apply the specialization. Fortunately, in many cases such a lift exists; indeed it always holds for functors on Set built with standard constructors as is made precise in Proposition 3.7. In particular we have plain lifts of the functors of all the examples of Hobor et. al.[2]

Definition 3.1. A functor $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is called *non-shrinking* if for any object X and any morphism $\varphi : X \to X$ of $\mathsf{CBUlt}_{\mathsf{ne}}$ and any $m > 0$ such that

$$\forall x, y \in X.\ x =_m y \Rightarrow \varphi(x) = \varphi(y)$$

we also have that

$$\forall x, y \in \hat{F}(X).\ x =_m y \Rightarrow \hat{F}(\varphi)(x) = \hat{F}(\varphi)(y).$$

[1] There is a conflict of nomenclature, what we call *worlds* are known as *knots* to Hobor et. al. Their worlds are pairs of knots and values.

[2] With the possible exception of [24, Example 2.7.]. The functor in that example is complex and the presentation dense to it is a bit hard to tell.

Here $x =_m y$ is short for $d(x, y) \leq 2^{-m}$ where d is the distance on X. Intuitively, elements of $\hat{F}(X)$ contain components from X. If closeness of two elements of $\hat{F}(X)$ implies similar closeness between the components, then $\hat{F}$ is non-shrinking because $\hat{F}(\varphi)$ applies φ to all components. Note that the condition is required only to hold for $m > 0$; the case $m = 0$ comes down to preserving constant functions and that would preclude, e.g., constant functors.

Definition 3.2. A metric space is *bisected* if any non-zero distance is of the form 2^{-m} for some $m \in \mathbb{N}$.

For bisected metric spaces we have the following proposition which is useful for showing maps non-expansive:

Proposition 3.3. A map $\varphi : X \to X$ on a bisected metric space X is non-expansive if and only if we have

$$\forall m \in \mathbb{N}.\ \forall x, y \in X.\ x =_m y \Rightarrow \varphi(x) =_m \varphi(y).$$

Definition 3.4. We say that a functor $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is the *lift* of a functor $F : \mathsf{Set} \to \mathsf{Set}$ if the following diagram commutes

$$\begin{array}{ccc} \mathsf{CBUlt}_{\mathsf{ne}} & \xrightarrow{\hat{F}} & \mathsf{CBUlt}_{\mathsf{ne}} \\ {\scriptstyle U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathsf{Set} & \xrightarrow[F]{} & \mathsf{Set}, \end{array}$$

where $U : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{Set}$ is the obvious forgetful functor. Furthermore, we say that a functor $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is *plain* if it is non-shrinking, locally non-expansive and, on objects, preserves the property of being bisected.

Theorem 3.5. Let $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ be a locally non-expansive functor and O a non-empty set. Then there is a non-empty, complete, 1-bounded ultrametric space X and an isometry

$$\Phi : X \cong \hat{F}\left(\tfrac{1}{2}\left(X \to_{ne} \mathit{UPred}(O)\right)\right).$$

This is an easy consequence of Theorem 2.1: $\hat{F}$ is assumed locally non-expansive and the functor $\frac{1}{2}((-) \to_{ne} \mathit{UPred}(O))$ is a locally contractive contravariant functor on $\mathsf{CBUlt}_{\mathsf{ne}}$ and so is the composite of the two.

Envision now a functor $F : \mathsf{Set} \to \mathsf{Set}$, a non-empty set O of values and a request for a solution to the recursive equation $K \cong F(K \times O \to 2)$. Indirection theory provides an approximative such, the above theorem another and the next theorem builds the former from the latter, thus demonstrating the generality of our approach.

We deviate from indirection theory as introduced by Hobor et. al. on two counts: We do not parameterize over the set of truth values but stick to $2 = \{0, 1\}$; the generalization, while probably technically feasible, appears unmotivated. More importantly, we build a solution that features

only *hereditary* maps from $K \times O$ to 2, see the definition below. This is a direct consequence of the uniformity required of members of $UPred(O)$, lifting the latter constraint would most likely remove the former too. But we regard it as a strength, not a shortcoming, as we really would like to stay hereditary all the way and now we know that 'unsquashing a knot' does not invalidate this wish – compare with the discussion in the last paragraph of [24, Section 10].

Theorem 3.6. Let $F : \mathsf{Set} \to \mathsf{Set}$ be a functor with a plain lift $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$. We can, from the isometry of Theorem 3.5, build a set K, a subset of *hereditary* maps $K \times O \to_{her} 2$ of the full function space $K \times O \to 2$ and two maps

$$K \underset{\text{squash}}{\overset{\text{unsquash}}{\rightleftarrows}} \mathbb{N} \times F(K \times O \to_{her} 2)$$

with the following three properties:

1. $\mathsf{squash} \circ \mathsf{unsquash} = 1_K$.
2. $(\mathsf{unsquash} \circ \mathsf{squash})(m, \nu) = (m, F(\mathsf{approx}_m)(\nu))$.
3. $\forall \psi \in K \times O \to 2.\ \psi \in K \times O \to_{her} 2 \Leftrightarrow \psi = \Box\, \psi$.

Here the $\mathsf{level} = \mathsf{fst} \circ \mathsf{unsquash} : K \to \mathbb{N}$ and the map $\mathsf{approx}_m : (K \times O \to_{her} 2) \to (K \times O \to_{her} 2)$ is defined, for each $m \in \mathbb{N}$, by

$$\mathsf{approx}_m(\psi)(k, o) = \psi(k, o) \wedge \mathsf{level}(k) < m.$$

And for $\psi \in K \times O \to 2$ we define $\Box\, \psi \in K \times O \to 2$ by

$$(\Box\, \psi)(k, o) = \forall l \in K.\ k\, \mathsf{A}^*\, l \Rightarrow \psi(l, o),$$

where A^* is the reflexive, transitive closure of the relation A on K defined, for any two $k, l \in K$, by

$$k\, \mathsf{A}\, l \Leftrightarrow \mathsf{unsquash}(k) = (m + 1, \nu) \wedge l = \mathsf{squash}(m, \nu).$$

Proposition 3.7. There is a plain lift of any functor built from the identity, constant non-empty sets, products, sums and (possibly finite and partial) maps from a constant set.

3.2 Advantages of Metric Solution Approach

Having proved that our metric-space approach specializes to the indirection theory of Hobor et. al. we now proceed to argue some advantages of our approach in general.

A first remark is this: We do not think of the operational semantics based version of our metric-space approach as more expressive than standard step-indexed models. Rather we view it as a framework for doing step-indexing, a conceptual guideline of sorts. This goes even if we disregard higher-order store circularities. Consider, e.g., the interpretation of recursive types in Section 2.2 above and in Appendix A below. The idea of 'stepping one down' when interpreting $\mu\alpha.\tau$ seems natural to anyone familiar with step-indexed models. But coming up with the correct criteria on the interpretation function for this to work out properly, also with nested recursive types, is not, a priori, so easy. If, however, we employ the metric approach, including Banach's fixed-point theorem, then writing down the requirements as done in the appendix is straightforward. Another example is the step-indexed model of Section 4, where we crucially rely on the metric on the worlds to define the $\otimes$ operator by Banach's fixed-point theorem. A similar construction could possibly be pushed through either with hand-built approximate worlds as employed by Ahmed et. al. [4] or with the indirection theory of Hobor et. al. [24]. But the precise course of action is less immediate and we expect that one could end up reinventing parts of metric theory on the way.

In direct comparison with the indirection theory in [24], we believe that our alternative approach of solving recursive metric equations has benefits. Both yield worlds to be used in Kripke models. There is, however, already a body of supporting theory for the metric-space approach that makes available a far greater range of worlds. To illustrate this point, let us focus on the step-indexed model of ML references discussed in Section 2.2 above and in Sections 2.1, 4.1 and 5 of [24]. In the model provided by indirection theory, types are arbitrary maps from worlds to values, modulo currying and nomenclature. But, as argued in [24, Section 5.1], we really want types that are both hereditary and monotone. In [24, Section 5.1] such types are elegantly identified using modal operators, but this does not change the problem that the types in a world may fail these criteria. This is recognized in the last paragraph of [24, Section 10] where an alternative, and less straightforward, model with only hereditary types in the worlds is sketched. But that means starting the model construction all over from scratch and does not buy us monotonicity. On the other hand, to obtain hereditary types with the metric approach we just use the uniformity condition on $UPred(V)$, verify Lemma 2.2 and apply Theorem 2.1. And to work with monotone types we can apply a slightly stronger existence result, cf. Appendix A and [17, Proposition 5.4] for *pre-ordered* metric spaces. A similar argument goes for the extension to mixed variance functors discussed [24, Section 10]: it is already supported by the metric-space approach. Indeed, in unpublished work we have used mixed-variance functors to verify that the metric-space approach scales to the elaborate worlds of [4].

Finally, we think that it is advantageous that the metric approach applies both to models based on domain theory and to models based on operational semantics.

4 Application: Step-Indexed Model of Separation Logic for Nested Hoare Triples

We next turn to a new application of recursively-defined sets of Kripke worlds: a step-indexed model of separation logic for nested Hoare triples.

In recent work, Schwinghammer et al. [32] present a domain-theoretic model of a variant of separation logic for a language that allows code to be stored in the heap (a form of "higher-order store"). The model is used to prove soundness of rules for "recursion through the heap" as well as soundness of higher-order frame rules that take stored code into account. (Both kinds of rules will be explained in more detail below.) The model is based on the solution of a recursive world equation using complete uniform subsets of a domain, akin to the situation in Section 2.1.

In this section we present a new, *operational* model of the same logic, following the approach outlined in Section 2.2. We do this for three reasons: first, to substantiate the claim that the metric-space technique works for both domain-theoretic and step-indexed models, and second, to illustrate the use of obtaining a *solution* (rather than an approximation of a solution) of a recursive equation for "worlds," and three, to obtain a *simpler* model than the one in [32]. (We do not claim that the new model is sound for more inference rules than the one in [32].)

The development in the rest of this section mostly follows the one in [32]; we shall highlight some key differences. The reader is assumed to be familiar with basic properties of separation logic [31].

4.1 Programming language

Figure 1 presents the language we consider [32]. It deviates from the "standard" core programming language of separation logic [31] in two ways. First, stack variables are immutable: only heap cells can be updated. Second, commands are first-class values that can be stored in the heap: there is a new form of expression called a *quoted command*, written $`C`$, and a new command for evaluating stored commands, written $\mathsf{eval}\ [e]$. Informally, if the value of e is an integer n, and if the current heap contains the quoted command $`C`$ at location n, then the command $\mathsf{eval}\ [e]$ executes C as a subroutine.

We write $\mathrm{fv}(C)$ for the set of free variables of the command C, and similarly for expressions. Let V be the set of closed values of the language, and let H be the set of *heaps*, i.e., finite maps from integers to closed values:

$$V = \mathbb{Z} \cup \{`C` \mid \mathrm{fv}(C) = \emptyset\},$$
$$H = \mathbb{Z} \rightharpoonup_{fin} V.$$

For two heaps $h_1, h_2 \in H$ we write $h_1 \# h_2$ if they have disjoint domains and $h_1 \cdot h_2$ for their union if this is the case. An *environment* is a finite map η from variables to closed values. When C is a command satisfying that $\mathrm{fv}(C) \subseteq \eta$, we let $\eta(C)$ denote the result of applying η to C as a capture-avoiding substitution. Given an expression e and an environment η such that $\mathrm{fv}(e) \subseteq \mathrm{dom}(\eta)$, we define $[\![e]\!]_\eta \in V$ as follows. When e is a quoted command $`C`$ we let $[\![e]\!]_\eta = [\![`C`]\!]_\eta = \eta(C)$. When e is an arithmetic expression, $[\![e]\!]_\eta$ is defined in the expected way, *except* that arithmetic operations on quoted commands are, for definiteness, given the meaning 0. Thence we avoid the complications of introducing undefined expressions in a Hoare-style logic.[3]

Expressions:

$$e ::= x \mid `C` \mid n \mid e_1 + e_2 \mid \ldots \qquad (n \in \mathbb{Z})$$

Commands:

$$\begin{aligned} C ::= {} & [e_1] := e_2 \mid \mathsf{let}\ x = [e]\ \mathsf{in}\ C \mid \mathsf{eval}\ [e] \\ & \mid \mathsf{let}\ x = \mathsf{new}(e)\ \mathsf{in}\ C \mid \mathsf{free}\ e \\ & \mid \mathsf{skip} \mid C_1; C_2 \mid \mathsf{if}\ (e_1 = e_2)\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \end{aligned}$$

Figure 1. Programming language.

The operational semantics of the language is defined by a small-step semantics, with configurations of the form (C, h) or abort. Configurations of the form (skip, h) or abort are terminal; An abort configuration indicates a memory fault or a runtime "type error" due to confusion between integers and quoted commands. The semantics is standard, all the reduction rules can be found in Appendix; here we just present the reduction rule for $\mathsf{eval}\ [e]$:

$$(\mathsf{eval}\ [e],\ h) \rightsquigarrow (C, h) \qquad \text{if } [\![e]\!] = n \text{ and } h(n) = `C`.$$

Example 4.1 (Iteration). The language does not include any high-level constructs for iteration. One can encode a "while" loop by means of "Landin's knot" in the heap:

$$\begin{aligned} &\mathsf{while}\ [e] \neq 0\ \mathsf{do}\ C \\ &\qquad \stackrel{def}{=}\ \mathsf{let}\ x = \mathsf{new}(`\mathsf{skip}`)\ \mathsf{in} \\ &\qquad\quad ([x] := `\ \mathsf{let}\ y = [e]\ \mathsf{in} \\ &\qquad\qquad\quad \mathsf{if}\ (y = 0)\ \mathsf{then}\ \mathsf{free}\ x \\ &\qquad\qquad\quad \mathsf{else}\ (C; \mathsf{eval}\ [x])`; \\ &\qquad\quad \mathsf{eval}\ [x]) \end{aligned}$$

(Here $x, y \notin \mathrm{fv}(e, C)$.) With that abbreviation, the following rule is derivable in the logic we present below:

$$\frac{\Gamma \vdash \{\exists y.\, e \mapsto y * I(y) \wedge y \neq 0\} C \{\exists y.\, e \mapsto y * I(y)\}}{\Gamma \vdash \{\exists y.\, e \mapsto y * I(y)\} \mathsf{while}\ [e] \neq 0\ \mathsf{do}\ C \{e \mapsto 0 * I(0)\}}$$

[3] A more robust approach would be to introduce a simple type system that distinguishes integers from quoted commands; for simplicity we do not do so here.

4.2 Logic

The formulas of the logic [32] are called *assertions* and are generated by the grammar:

$$\begin{aligned} P,Q \;::=\; & \mathsf{false} \mid \mathsf{true} \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid \\ & \forall x.P \mid \exists x.P \mid \mathsf{int}(e) \mid e_1 = e_2 \mid e_1 \leq e_2 \mid \\ & e_1 \mapsto e_2 \mid \mathsf{emp} \mid P * Q \mid P \mathrel{-\!\!*} Q \\ & \{P\}e\{Q\} \mid P \otimes Q \mid \cdots \end{aligned}$$

where the dots refer to atomic predicates and recursively defined predicates of the form $(\mu\alpha(x).P)(e)$ with α in P only occurring in "contractive" positions. (For space reasons, we de not formalize recursively defined assertions syntactically, but just treat them semantically, see below.) Unlike in standard separation logic, assertions are used both to describe predicates on heaps and to describe specifications of commands.

Indeed, the assertion $\{P\}e\{Q\}$ means, intuitively, that the value of e is a quoted command 'C' which satisfies the Hoare triple with precondition P and postcondition Q in the usual sense of separation logic. Since Hoare triples are assertions, they can appear in pre- and post-conditions of other triples. Such *nested* triples are useful for reasoning about stored code: the specification of a command C can depend on the specification of other code in the heap, e.g.,

$$\{P * \exists y.\, x \mapsto y \wedge \{P'\}y\{Q'\}\}\text{`}C\text{`}\{Q\}\,. \tag{1}$$

Here a part of the precondition of C is that x points to a command y satisfying $\{P'\}y\{Q'\}$. Presumably, the reason is that C contains one or more occurrences of $\mathsf{eval}\ [x]$.

The assertion $P \otimes Q$ should be thought of as "the assertion P extended with the invariant Q" and this assertion form is used to codify higher-order frame rules [15]. See [32] for detailed discussion of soundness and unsoundness of variations higher-order frame rules in the presence of higher-order store.

Proof Rules The proof rules include the standard rules for intuitionistic predicate logic and the logic of bunched implications [28]. Moreover, there are variations of standard separation logic proof rules (for dereferencing, sequencing, and so on). For brevity we only show the rule for dereferencing (the rest can be found in Appendix):

$$\frac{\Gamma, x \vdash \{P * e \mapsto x\}\text{`}C\text{`}\{Q\}}{\Gamma \vdash \{\exists x.\, P * e \mapsto x\}\text{`}\mathsf{let}\ x = [e]\ \mathsf{in}\ C\text{`}\{Q\}}\ (x \notin \mathrm{fv}(e, Q))$$

Here Γ ranges over finite sets of variables.

In addition to these standard rules, there are two kinds of frame rules and a rule for executing stored code, see Figure 2. Rule ($\otimes$-FRAME) is a deep frame rule in which the invariant Q intuitively is added to all pre- and post-conditions inside P. The latter intuition is captured by the

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \otimes Q}\ (\otimes\text{-FRAME})$$

$$\frac{}{\Gamma \vdash \{P\}e\{Q\} \Rightarrow \{P * R\}e\{Q * R\}}\ (*\text{-FRAME})$$

$$\frac{\Gamma, k \vdash R[k] \Rightarrow \{P * e \mapsto R[\_]\}k\{Q\}}{\Gamma \vdash \{P * e \mapsto R[\_]\}\text{`}\mathsf{eval}\ [e]\text{`}\{Q\}}\ (\text{EVAL})$$

Figure 2. Selected Proof Rules.

axioms in Figure 3. Rule ($*$-FRAME) is a shallow (first-order) frame axiom. Finally, rule (EVAL) is the rule for executing stored code. Here, $e \mapsto R[\_]$ is an abbreviation of $\exists x.\, e \mapsto x \wedge R[x]$ (for an x not free in R).

4.3 A step-indexed model

To model invariant extension $P \otimes Q$, Schwinghammer et. al. [32] models an assertion as a *function* that takes the meaning of a second, arbitrary assertion (to be thought of as the "invariant" that the first assertion is extended with) and gives a predicate on heaps.[4] This approach introduces a circularity, however, since such a function will in particular be applicable to itself. In the next section we show how to formalize and solve the circularity using metric spaces.

4.3.1 Semantic predicates

Following Section 2.2, we let $UPred(H)$ be the set of subsets of $\mathbb{N} \times H$ that are downwards closed in the first component:

$$\{p \subseteq \mathbb{N} \times H \mid \forall (k,h) \in p.\, \forall j \leq k.\, (j,h) \in p\}.$$

We give $UPred(H)$ the same distance function as in Section 2.2; the set then becomes a complete, bounded ultrametric space. Using Theorem 2.1 we obtain a unique $\mathcal{W} \in \mathsf{CBUlt}_{\mathsf{ne}}$ satisfying

$$\mathcal{W} \cong \tfrac{1}{2}(\mathcal{W} \to UPred(H))\,. \tag{2}$$

Define $Pred = \frac{1}{2}(\mathcal{W} \to UPred(H))$ and let $i : Pred \to \mathcal{W}$ be the isomorphism. We will model assertions as elements of $Pred$.

Let the letters p and q range over elements of $Pred$. We order the elements of $Pred$ pointwise:

$$p \leq q \iff \forall w \in \mathcal{W}.\, p(w) \subseteq q(w)$$

[4] This idea follows earlier work on invariant extension [15, 16], which does not, however, deal with nested Hoare triples.

$$
\begin{array}{rcll}
P \circ R & \stackrel{def}{=} & (P \otimes R) * R & \{P\}e\{Q\} \otimes R \iff \{P \circ R\}e\{Q \circ R\} \\
(\kappa x.P) \otimes R & \iff & \kappa x.(P \otimes R) & (\kappa \in \{\forall, \exists\},\ x \notin \mathrm{fv}(R)) \\
(P \oplus Q) \otimes R & \iff & (P \otimes R) \oplus (Q \otimes R) & (\oplus \in \{\Rightarrow, \wedge, \vee, *, -\!\!*\}) \\
P \otimes R & \iff & P & (P \text{ is } \mathsf{true}, \mathsf{false}, \mathsf{emp}, e_1 = e_2, e_1 \mapsto e_2, \text{ or } \mathsf{int}(e)) \\
(P \otimes R) \otimes R' & \iff & P \otimes (R \circ R') & P \otimes \mathsf{emp} \iff P
\end{array}
$$

Figure 3. Axioms for invariant extension.

Lemma 4.2. With the ordering above and the following operations, $Pred$ is a complete BI-algebra [14]:

$$
\begin{aligned}
emp(w) &= \{(n, []) \mid n \in \mathbb{N}\} \\
(p * q)(w) &= \{(n, h) \mid \exists h_1, h_2.\, h = h_1 \cdot h_2 \\
&\qquad \wedge (n, h_1) \in p(w) \wedge (n, h_2) \in q(w)\} \\
(p -\!\!* q)(w) &= \{(n, h) \mid \forall m \leq n. \\
&\qquad ((m, h') \in p(w) \wedge h \# h') \Rightarrow (m, h \cdot h') \in q(w)\}
\end{aligned}
$$

The fact that $Pred$ is a complete BI algebra immediately gives us a sound interpretation of most of the assertions in the logic [14], but to interpret recursive predicates we also need to know that the operations are non-expansive:

Lemma 4.3. The BI-algebra operations on $Pred$ given by the previous lemma are non-expansive:

$$
\begin{gathered}
, -\!\!, \to, \wedge, \vee : Pred \times Pred \to Pred \\
\textstyle\bigvee_I, \bigwedge_I : (I \to Pred) \to Pred.
\end{gathered}
$$

(In the last two operations, the indexing set I is given the discrete metric.)

Proof. Easy verification. One first shows the analogous property for $UPred(H)$. To illustrate what follows, consider $*$: $UPred(H) \times UPred(H) \to UPred(H)$: It suffices to show that if $p \stackrel{n}{=} p'$ and $q \stackrel{n}{=} q'$, then also $(p * q) \stackrel{n}{=} (p' * q')$. The latter is equivalent to showing that $\forall m < n.\, (m, h) \in p * q \iff (m, h) \in p' * q'$, which follows easily by the assumption. □

4.3.2 Interpretation of invariant extension

To interpret invariant-extension assertions $P \otimes Q$, we need a operator $\otimes$ on the set of semantic predicates $Pred$. The most convenient way to specify $\otimes$ is to give a certain recursive equation that it must satisfy. Using the metric-space setup we can then prove that there exists a unique operator satisfying this specification, by an easy application of Banach's fixed point theorem, as in [32].

Proposition 4.4. There exists a unique function $\otimes : Pred \times \mathcal{W} \to Pred$ in $\mathsf{CBUlt}_{\mathsf{ne}}$ satisfying

$$
p \otimes w = \lambda w'.\, p(w \circ w')
$$

where $\circ : \mathcal{W} \times \mathcal{W} \to \mathcal{W}$ is given by

$$
w_1 \circ w_2 = i((i^{-1}(w_1) \otimes w_2) * i^{-1}(w_2)).
$$

Observe that it is here that we exploit that we have obtained a proper solution to the world equation (2) as a metric space such that we can now easily establish the existence of the recursively-defined $\otimes$-operation.

The basic properties of $\otimes$ and $\circ$ are conveniently summarized as follows:

Proposition 4.5. 1. $(\mathcal{W}, \circ, \mathsf{emp})$ is a monoid.

2. The operator $\otimes$ is a monoid action of $\mathcal{W}$ on $Pred$: for all $P \in Pred$ and $w_1, w_2 \in \mathcal{W}$ we have $P \otimes \mathsf{emp} = P$ and $(P \otimes w_1) \otimes w_2 = P \otimes (w_1 \circ w_2)$.

4.3.3 Interpretation of assertions

We next define a semantic interpretation of Hoare triples. To this end we let Safe_m be the set of configurations in the operational semantics that are safe for m reduction steps, that is, those configurations that do not reduce to abort in m (or fewer) steps. We write $\rightsquigarrow_k$ for the k-step reduction relation of the operational semantics.

Now say that $w \models_n (p, C, q)$ holds iff: For all $r \in UPred$, all $m < n$ and all heaps h, if $(m, h) \in p(w) * i^{-1}(w)(\mathsf{emp}) * r$, then:

1. $(C, h) \in \mathrm{Safe}_m$.
2. For all $k \leq m$ and all $h' \in H$, if $(C, h) \rightsquigarrow_k (\mathsf{skip}, h')$, then $(m - k, h') \in q(w) * i^{-1}(w)(\mathsf{emp}) * r$.

This definition is similar to the one in [32] with its use of the invariant w and the baking-in of the first order frame rule, i.e., the quantification over r. The difference is that the meaning is now relative to the operational semantics (rather than denotational) and that we use step-indexing to measure to what extent pre- and post-conditions should hold.

The intention is, of course, that a Hoare-triple assertion is interpreted using the above semantic construct. However, to see that this interpretation gives a well-defined member of $Pred$, we need to know that a semantic Hoare triple is "non-expansive in w":

Proposition 4.6. If $w =_k w'$ and $w \models_n (p, C, q)$, then $w' \models_{n \wedge (k-1)} (p, C, q)$.

Proof. Easy verification, using the fact that the separating conjunction $*$ on $UPred(V)$ is non-expansive (Lemma 4.3). □

The interpretation of an assertion $\Gamma \vdash P$ is now defined to be an element $[\![P]\!]_\eta$ in $Pred$, for η an environment mapping the variables in the domain of Γ to V. The definition uses the complete BI-algebra structure on $Pred$ given earlier to interpret the standard logical connectives, e.g.,

$$[\![P * Q]\!]_\eta w = [\![P]\!]_\eta w * [\![Q]\!]_\eta w.$$

Invariant extension is interpreted as follows:

$$[\![P \otimes Q]\!]_\eta w = \Big([\![P]\!]_\eta \otimes i([\![Q]\!]_\eta)\Big)\, w$$

and, finally, Hoare triples are interpreted like this:

$$[\![\{P\}e\{Q\}]\!]_\eta w = \begin{cases} \{(n,h) \mid w \models_n ([\![P]\!]_\eta, C, [\![Q]\!]_\eta)\} & \text{if } [\![e]\!]_\eta = \text{`}C\text{`} \\ \emptyset & \text{otherwise.} \end{cases}$$

The concrete interpretation of all the logical connectives can be found in Appendix. As in [32], recursively defined predicates are interpreted via Banach's fixed point theorem:

Proposition 4.7. Let I be a set and suppose that, for each $i \in I$, $F_i : Pred^I \to Pred$ is a contractive function. Then there exists a unique $\vec{p} = (p_i)_{i\in I} \in Pred^I$ such that $F_i(\vec{p}) = p_i$, for all $i \in I$.

4.3.4 Soundness of proof rules

We define semantic validity of (open) assertions as follows: For an assertion P with free variables belonging to Γ, say that $\Gamma \models P$ iff: For all environments η with $\Gamma \subseteq \mathrm{dom}(\eta)$ and all $w \in \mathcal{W}$ we have $[\![P]\!]_\eta w = \mathbb{N} \times H$. This amounts to saying that $[\![P]\!]_\eta$ is the top element of the BI algebra $Pred$.

Theorem 4.8. If $\Gamma \vdash P$, then $\Gamma \models P$.

Proof. By showing the stronger property that each proof rule holds semantically, that is, with $\vdash$ replaced by $\models$. We only include the proof case for $\mathsf{eval}\ [e]$ (the other interesting cases are the ones for invariant extension; there one uses Proposition 4.5). We must show: if $\Gamma, z \models R[z] \Rightarrow \{P * e \mapsto R[\_]\}z\{Q\}$, then $\Gamma \models \{P * e \mapsto R[\_]\}\text{`}\mathsf{eval}\ [e]\text{`}\{Q\}$.

Let η be an environment with $\Gamma \subseteq \mathrm{dom}(\eta)$, and let w and n be arbitrary. We must show that

$$w \models_n \{[\![P * e \mapsto R[\_]]\!]\eta\}\text{`}\eta(\mathsf{eval}\ [e])\text{`}\{[\![Q]\!]\eta\}. \qquad (3)$$

So let $k < n$ and $r \in UPred$ and let $(k,h) \in [\![P * e \mapsto R[\_]]\!]\eta(w) * i^{-1}(w)(\mathsf{emp}) * r$. Then $h = h_1 * [l \mapsto v] * h_2 * h_3$, where $(k,h_1) \in [\![P]\!]\eta(w)$ and $[\![e]\!]\eta = l$ and $(k,[l \mapsto v]) \in [\![R[z]]\!]\eta[z \mapsto v](w)$ and $(k,h_2) \in i^{-1}(w)(\mathsf{emp})$ and $(k,h_3) \in r$. Using validity of the premise, we get that $(k,[l \mapsto v]) \in [\![\{P * e \mapsto R[z]\}z\{Q\}]\!]\eta[z \mapsto v](w)$, which means that $v = \text{`}C\text{`}$ for some C, and that $w \models_k \{[\![P * e \mapsto R[z]]\!]\eta\}C\{[\![Q]\!]\eta\}$.

Now, if $k = 0$, then conditions 1 and 2 in the definition of $\models$ are clearly satisfied (item 2 because $(\eta(\mathsf{eval}\ [e]), h)$ takes a reduction step), so (3) holds, as required. If $k > 0$ then, first observe that by downwards closure we have $(k-1,h) \in [\![P * e \mapsto R[\_]]\!]\eta(w) * i^{-1}(w)(\mathsf{emp}) * r$. Therefore, $(C,h) \in \mathrm{Safe}_{k-1}$, which implies that $(\eta(\mathsf{eval}\ [e]), h) \in \mathrm{Safe}_k$, so condition 1 in definition of $\models$ is satisfied. For condition 2, we finally assume that $(\eta(\mathsf{eval}\ [e]), h) \rightsquigarrow_m (\mathsf{skip}, h')$ for some h' and $m \leq k$. Then $(C,h) \rightsquigarrow_{m-1} (\mathsf{skip}, h')$. Since $m-1 \leq k-1$, we then get $((k-1)-(m-1), h') \in [\![Q]\!]\eta(w) * i^{-1}(w)(\mathsf{emp}) * r$, as required. □

4.4 Discussion

In summa, we have developed a new step-indexed model of separation logic with nested Hoare triples for reasoning about higher-order store. The new model is arguably simpler than the one in [32], since it is phrased directly in terms of the operational semantics without passing through a domain-theoretic denotational semantics. A usual advantage of using domain-theory is a more abstract semantics, but in [32], it was in necessary to employ certain "step-like," rank functions, so in the end the model of *loc.cit.* was not more abstract than the new one presented here.

5 Related and Future Work

Relational Reasoning We have focused on unary reasoning in this paper, but as mentioned in the introduction the techniques developed here also apply to relational reasoning. Relational reasoning about higher-order store, e.g., logical relations for reasoning about contextual equivalence of programs, have been developed both based on domain theory, e.g., [12, 18], and on step-indexed models, e.g., [4]. For such relational reasoning, the worlds are typically more sophisticated than the worlds we have discussed so far, in order to describe situations in which programs are contextually equivalent even though they use local state in different ways. The third author has recently phrased the state-of-the-art world model from [4] as a recursive world equation over a domain-theoretic model. The reason for doing so is to obtain more abstract reasoning principles when using the resulting model for proving actual program equivalences, without having to reason about step-indices. As an alternative, Dreyer et. al. [22] have shown how to extend the relational step-indexed model [4] to a model of a modal logic for more abstract reasoning about program equivalences. The latter modal logic has been derived from the step-indexed model; we believe it is still a challenge to develop relational step-indexed models of some existing logics; e.g., for Hoare Type Theory [27]. Alternatively, one might try to develop a new formulation of (the ideas of) Hoare Type Theory based on a step-indexed model.

Formalization An often mentioned advantage of the traditional step-indexed approach is that it lends itself well to formalization in theorem provers and, indeed, impressive formalization work has been carried out in, e.g., Coq [10].

Thus, one may wonder, whether our proposed metric approach will hinder formalizations. We claim that it will not. Indeed, Varming has recently formalized the solution of recursive metric-space equations in Coq [34], following the treatment in [17]. This formalization can be used in concert with a formalization of operational semantics to yield, e.g., formalizations of the models in Sections 2.2 4, or with the formalizations of Benton et. al. of domain theory [13] to yield formalizations of models based on domain theory.

6 Conclusion and Acknowledgements

In conclusion, the key *conceptual contributions* of this paper are (1) the realization that models of higher-order store phenomena naturally can be given as Kripke models over worlds that are recursively defined in a category of metric spaces; and (2) a unification of methods based on classical domain theory and on step-indexed operational models. Our *technical contributions* include that (1) we have shown how to solve world equations for concrete step-indexed models; (2) we have shown that our metric approach can be specialized to Hobor et. al.'s recent proposal [24] (and argued that the metric approach has some advantages); and (3) we have developed a new model of separation logic with nested Hoare triples for reasoning about higher-order store, a model which shows the utility of the metric approach since it relies on a recursively defined operator on worlds.

We would like to thank Aquinas Hobor, Robert Dockins, Andrew W. Appel and Francois Pottier for helpful discussions and insightful comments.

References

[1] Martín Abadi and Gordon D. Plotkin. A per model of polymorphism and recursive types. In *Proceedings of LICS*, pages 355–365, 1990.

[2] A. J. Ahmed, A. W. Appel, and R. Virga. A stratified semantics of general references. In *Proceedings of LICS 2002*, 2002.

[3] Amal Ahmed. *Semantics of Types for Mutable State*. PhD thesis.

[4] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. State-dependent representation independence. In *Proceedings of POPL*, 2009. To appear.

[5] Roberto M. Amadio. Recursion over realizability structures. *Information and Computation*, 91(1):55–85, 1991.

[6] Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*. Cambridge University Press, 1998.

[7] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.*, 39(3):343–375, 1989.

[8] A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5), 2001.

[9] A. W. Appel, P. Melliès, C. D. Richards, and J. Vouillon. A very modal model of a modern, major, general type system. In *Proceedings of POPL 2007*, 2007.

[10] A.W. Appel, R. Dockins, and A. Hobor. Mechanized semantic library. `http://msl.cs.princeton.edu/`, 2009.

[11] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *Proceedings of TLCA*, 2005.

[12] N. Benton, L. Beringer, M. Hofmann, and A. Kennedy. Relational semantics for effect-based program transformations: Higher-order store. In *Proceedings of PPDP'09*, 2009.

[13] N. Benton, A. Kennedy, and C. Varming. Some domain theory and denotational semantics in coq. In *Proceedings of TPHOLs'09*, 2009.

[14] B. Biering, L. Birkedal, and N. Torp-Smith. Bi-hyperdoctrines, higher-order separation logic, and abstraction. *ACM Trans. Program. Lang. Syst.*, 29(5), 2007.

[15] L. Birkedal, N. Torp-Smith, and H. Yang. Semantics of separation-logic typing and higher-order frame rules for Algol-like languages. *LMCS*, 2(5:1), 2006. URL `http://dx.doi.org/10.2168/LMCS-2(5:1)2006`.

[16] L. Birkedal, B. Reus, J. Schwinghammer, and H. Yang. A Simple Model of Separation Logic for Higher-order Store. In *Proceedings of ICALP 2008*, 2008.

[17] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space quations. Technical Report ITU-2009-119, IT University of Copenhagen, 2009.

[18] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In *Proceedings of FOSSACS*, number 5504 in LNCS, pages 456–470, 2009.

[19] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Proceedings of APLAS*, number 4279 in LNCS, pages 79–96, 2006.

[20] F. Cardone. Relational semantics for recursive types and bounded quantification. In *Proceedings of ICALP*, 1989.

[21] Jaco de Bakker and Erik de Vink. *Control flow semantics*. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-04154-5.

[22] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In *Proceedings of POPL'2010*, 2010.

[23] A. Hobor, A.W. Appel, and F.Z. Nardelli. Oracle semantics for concurrent separation logic. In *Proceedings of ESOP'08*, 2008.

[24] A. Hobor, R. Dockins, and A.W. Appel. A theory of indirection via approximation. In *Proceedings of POPL'2010*, 2010.

[25] P. B. Levy. Possible world semantics for general storage in call-by-value. In *CSL: 16th Workshop on Computer Science Logic*, volume 2471 of *LNCS*.

[26] David B. MacQueen, Gordon D. Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71(1/2):95–130, 1986.

[27] A. Nanevski, G. Morrisett, and L. Birkedal. Polymorphism and separation in hoare type theory. In *Proceedings of ICFP'06*, 2006.

[28] Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.

[29] Bernhard Reus and Jan Schwinghammer. Separation logic for higher-order store. In *Proc. CSL'06*, volume 4207 of *LNCS*, pages 575–590, 2006.

[30] Bernhard Reus and Thomas Streicher. Semantics and logic of object calculi. In *Proceedings of 17th Annual IEEE Symposium Logic in Computer Science*, pages 113–124. IEEE Computer Society Press, 2002.

[31] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of LICS*, pages 55–74, 2002.

[32] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In *Proceedings of CSL*, number 5771 in LNCS, pages 440–454, 2009.

[33] Michael B. Smyth. Topology. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[34] C. Varming. Recursive metric-space equations in coq. Personal Communication, 2009.

A Step-Indexed Model of References

A.1 Language

The language is as in [22] , except that we split the context for type variables and term variables in two so that term judgments take the form:

$$\Delta; \Gamma; \Sigma \vdash M : \tau$$

for Δ a context of type variables $\alpha_1, \ldots, \alpha_n$, Γ a context of term variables $x_1 : \tau_1, \ldots, x_m : \tau_m$, and Σ a context of locations $l_1 : \sigma_1, \ldots, l_k : \sigma_k$.

Detailed typing judgments and operational semantics can be found in the online appendix to [22].

A.2 Model

Let V denote the set of closed syntactic values and let $UPred(V)$ be the set

$$\{P \subseteq \mathbb{N} \times V \mid \forall (k, v) \in P. \forall j \leq k. (j, v) \in P\}.$$

We can define a distance function on $UPred(V)$, which measures "up-to-what-level" two uniform predicates agree, as follows: First, for $P \in UPred(V)$, let $\overline{P}^k$ denote $\{(m, v) \in P \mid m < k\}$, and then define distance function d by:

$$d(P, Q) = \begin{cases} 2^{-n} & \text{if } P \neq Q \text{ and n = } \max\{k \mid \overline{P}^k = \overline{Q}^k\} \\ 0 & \text{if } P = Q. \end{cases}$$

Lemma A.1. $(UPred(V), d)$ is a well-defined object in $\mathsf{CBUlt_{ne}}$.

In the same manner, we let E denote the set of closed syntactic expressions and define $UPred(E)$ to the corresponding set of uniform predicates on E.

Let $\mathsf{PreCBUlt_{ne}}$ denote the category with objects $(A, \leq)$ where A an object of $\mathsf{CBUlt_{ne}}$ and $\leq$ is a preorder on the underlying set of A such that the following condition holds: if $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ are converging sequences in A with $a_n \leq b_n$ for all n, then also $\lim_{n\to\infty} a_n \leq \lim_{n\to\infty} b_n$. Morphisms are functions that are both monotone and non-expansive. By Proposition 5.4 in [17] we then immediately get:

Theorem A.2. There exists a preordered non-empty complete bounded ultra-metric space W with an isomorphism

$$W \cong \mathbb{N} \rightharpoonup_{fin} \tfrac{1}{2}(W \to_{mon} UPred(V))$$

in $\mathsf{PreCBUlt_{ne}}$. One member of the right hand side is less than another if the domain of the first is included in the domain of the second and they agree on the former.

Semantic value types will be modeled as elements of

$$T \quad = \quad W \to_{mon} UPred(V),$$

and the semantic computation types will be modeled as elements of

$$T_E \quad = \quad W \to UPred(E).$$

For Δ a context of type variables, we use φ to range over the product space $T^{|\Delta|}$ in $\mathsf{CBUlt_{ne}}$.

We now define the interpretation of types in context as a function

$$\mathcal{V}[\![\Delta \vdash \tau]\!] : T^{|\Delta|} \to T$$

in $\mathsf{CBUlt_{ne}}$ (i.e., note that the function space consists of non-expansive functions). The function is defined like this:

$$\begin{array}{l}
\mathcal{V}[\![\Delta \vdash 1]\!]\varphi = \lambda w. \{(k, *) \mid k \in \mathbb{N}\} \\
\mathcal{V}[\![\Delta \vdash \alpha]\!]\varphi = \varphi(\alpha) \\
\mathcal{V}[\![\Delta \vdash \tau_1 \to \tau_2]\!]\varphi = \lambda w. \{(k, v) \mid \forall w' \geq w. \forall j \leq k. \\
\quad \forall v_1. (j, v_1) \in \mathcal{V}[\![\Delta \vdash \tau_1]\!]\varphi w' \\
\quad\quad \Rightarrow (j, vv_1) \in \mathcal{E}[\![\Delta \vdash \tau_2]\!]\varphi w'\} \\
\mathcal{V}[\![\Delta \vdash \mathsf{ref}\, \tau]\!]\varphi = \lambda w. \{(k, l) \mid \\
\quad (l \in \mathrm{dom}(w) \wedge w(l) \overset{k}{=}_T \mathcal{V}[\![\Delta \vdash \tau]\!]\varphi)\} \\
\mathcal{V}[\![\Delta \vdash \mu\alpha.\tau]\!]\varphi = \mathit{fix}(\lambda r. \lambda w. \{(k, \mathsf{fold}\, v) \mid k > 0 \Rightarrow \\
\quad (k-1, v) \in \mathcal{V}[\![\Delta, \alpha \vdash \tau]\!]\varphi[\alpha \mapsto r]w\}) \\
\mathcal{V}[\![\Delta \vdash \forall\alpha.\tau]\!]\varphi = \lambda w. \{(k, v) \mid \forall \tau_0 \in \mathit{Type}. \forall r \in T. \\
\quad \forall w' \geq w. \forall j \leq k. \\
\quad\quad (j, v[\tau_0]) \in \mathcal{E}[\![\Delta, \alpha \vdash \tau]\!]\varphi[\alpha \mapsto r]w'\}) \\
\mathcal{E}[\![\Delta \vdash \tau]\!]\varphi = \lambda w. \{(k, e) \mid \forall j \leq k. \forall s, v, s'. \\
\quad \big((e, s) \Downarrow^j (v, s') \wedge s :_k w\big) \\
\quad \Rightarrow \big(\exists w' \geq w. s' :_{k-j} w' \wedge (k-j, v) \in \mathcal{V}[\![\Delta \vdash \tau]\!]\varphi w'\big)\}
\end{array}$$

$$\begin{array}{l}
s :_k w \iff \forall j < k. \mathrm{dom}(s) = \mathrm{dom}(w) \wedge \\
\qquad\qquad \forall l \in \mathrm{dom}(w). (j, s(l)) \in w(l)(w)
\end{array}$$

Remark A.3.

- Note that in the case for $\mathcal{V}[\![\Delta \vdash \mathsf{ref}\, \tau]\!]$, we use k-equality in the space T.

Lemma A.4. If $w \overset{n}{=} w'$ and $w_0 \geq w$ then there exists w_0' with

$$w_0'(l) = \begin{cases} w'(l) & \text{if } l \in \mathrm{dom}(w') \\ w_0(l) & \text{otherwise} \end{cases}$$

and $w_0' \in W$ and $w_0' \overset{n}{=} w_0$.

Lemma A.5. If $s :_k w$ and $w \overset{n}{=}_W w'$ and $k < n$, then also $s :_k w'$.

Proof. We are to show that $\forall j < k. \mathrm{dom}(s) = \mathrm{dom}(w') \wedge \forall l \in \mathrm{dom}(w'). (j, s(l)) \in w'(l)(w')$. It holds vacously if $k = 0$, so assume $k > 0$. Then also $n > 0$. Let $j < k$ be arbitrary. By the assumption that $w \overset{n}{=}_W w'$, we get that

$\mathrm{dom}(w) = \mathrm{dom}(w') \wedge \forall l \in \mathrm{dom}(w).\, \forall w_0.\, w(l)(w_0) \stackrel{n-1}{=} w'(l)(w_0)$. Since $\mathrm{dom}(s) = \mathrm{dom}(w)$ by the assumption that $s :_k w$ (using $k > 0$), we get $\mathrm{dom}(s) = \mathrm{dom}(w')$, as required. Moreover, we find that

$$w(l)(w) \stackrel{n}{=} w(l)(w') \stackrel{n-1}{=} w'(l)(w')$$

with the first equality since $w(l)$ is non-expansive, and the second equality by the assumption that $w \stackrel{n}{=}_W w'$. Thus, as $(j, s(l)) \in w(l)(w)$ by assumption, and since $j < k \leq n - 1$, we also get $(j, s(l)) \in w'(l)(w')$, as desired. □

Lemma A.6. $\mathcal{V}[\![\Delta \vdash \tau]\!]$ and $\mathcal{E}[\![\Delta \vdash \tau]\!]$ are well-defined. In particular,

- $\mathcal{V}[\![\Delta \vdash \tau]\!]\varphi \in T$ (so non-expansive and monotone),
- $\mathcal{E}[\![\Delta \vdash \tau]\!]\varphi \in T_E$ (so non-expansive),
- $\mathcal{V}[\![\Delta \vdash \tau]\!]$ is a non-expansive map,
- The function $r \mapsto \lambda w.\, \{(k, \mathsf{fold}\, v) \mid k > 0 \Rightarrow (k-1, v) \in \mathcal{V}[\![\Delta, \alpha \vdash \tau]\!]\varphi[\alpha \mapsto r]w\}$ is contractive so the fixed point taken in $\mathcal{V}'[\![\Delta \vdash \mu\alpha.\tau]\!]$ exists (this boils down to the use of $k - 1$).

We now define interpretations of contexts and the logical relation interpretation of well-typed expressions:

$$\begin{array}{l}
\mathcal{D}[\![\Delta]\!] : T^{|\Delta|} \\
\mathcal{D}[\![\emptyset]\!] = \emptyset \\
\mathcal{D}[\![\Delta, \alpha]\!] = \{\varphi[\alpha \mapsto r] \mid \\
\quad \varphi \in \mathcal{D}[\![\Delta]\!] \wedge r \in T\}
\end{array}$$

$$\begin{array}{l}
\mathcal{G}[\![\Delta \vdash \Gamma]\!] : T^{|\Delta|} \to W \to \mathit{UPred}(V^{|\Gamma|}) \\
\mathcal{G}[\![\Delta \vdash \emptyset]\!]\varphi = \emptyset \\
\mathcal{G}[\![\Delta \vdash \Gamma, x : \tau]\!]\varphi = \lambda w.\, \{(k, \gamma[x \mapsto v]) \mid \\
\quad (k, \gamma) \in \mathcal{G}[\![\Gamma]\!]\varphi w \wedge (k, v) \in \mathcal{V}[\![\Delta \vdash \tau]\!]\varphi w\}
\end{array}$$

$$\begin{array}{l}
\mathcal{S}[\![\Sigma]\!] = \{(k, w) \mid \forall (l : \tau) \in \Sigma. \\
\quad (k, l) \in \mathcal{V}[\![\emptyset \vdash \mathsf{ref}\, \tau]\!]\emptyset w\}
\end{array}$$

$$\begin{array}{l}
\Delta; \Gamma; \Sigma \vdash e :^{\log} \tau \iff \\
\quad \exists \alpha_1, \ldots, \alpha_n.\, \Delta = \alpha_1, \ldots, \alpha_n \wedge \forall \tau_1, \ldots, \tau_n. \\
\quad \forall k \geq 0.\, \forall \varphi.\, \forall \gamma.\, \forall w. \\
\quad \big(\varphi \in \mathcal{D}[\![\Delta]\!] \wedge (k, \gamma) \in \mathcal{G}[\![\Delta \vdash \Gamma]\!]\varphi w \wedge (k, w) \in \mathcal{S}[\![\Sigma]\!]\big) \\
\quad \Rightarrow \big((k, [\alpha_1 \mapsto \tau_1, \ldots, \alpha_n \mapsto \tau_n](\gamma(e))) \in \mathcal{E}[\![\Delta \vdash \tau]\!]\varphi w\big)
\end{array}$$

Theorem A.7 (Fundamental Theorem of Logical Relations)**.** If $\Delta; \Gamma; \Sigma \vdash e : \tau$, then $\Delta; \Gamma; \Sigma \vdash e :^{\log} \tau$.

B Specialization to Indirection Theory, Three Proofs

Proof of Proposition 3.3. It is immediate that any non-expansive φ has the stated property. Assume, on the other hand, that we need to show φ non-expansive. Let $x, y \in X$, we must show that $d(\varphi(x), \varphi(y)) \leq d(x, y)$, where d is the metric on X. We may without loss of generality assume that $d(x, y) \neq 0$. But then there is $m \in \mathbb{N}$ with $d(x, y) = 2^{-m}$, in particular we have $d(x, y) \leq 2^{-m}$ which we usually write $x =_m y$. From the assumption we get that $\varphi(x) =_m \varphi(y)$, i.e., that $d(\varphi(x), \varphi(y)) \leq 2^{-m}$ and we are done. □

Proof of Theorem 3.6. Let X and Φ be the result of applying Theorem 3.5 to $\hat{F}$. Note initially that X must be bisected. This is by definition the case for $\mathit{UPred}(O)$ and hence any two elements of $X \to_{ne} \mathit{UPred}(O)$ have a distance that is the supremum of a nonempty subset of $\{0\} \cup \{2^{-m} \mid m \in \mathbb{N}\}$. But this set is closed under nonempty suprema and so $X \to_{ne} \mathit{UPred}(O)$ is bisected too. Both of the functors $\frac{1}{2}(-)$ and $\hat{F}$ preserve the property of being bisected, the former by construction and the latter by assumption. And so X, which is isometric to $\hat{F}\left(\frac{1}{2}(X \to_{ne} \mathit{UPred}(O))\right)$, must be bisected.

Without further ado, let us plunge into the construction. For every $m \in \mathbb{N}$ we know that $=_m$ is an equivalence relation on X, for $x \in X$ we denote by $[x]_m$ the equivalence class containing x. We let K be the sum of all but the first of the sets of equivalence classes:

$$K = \sum_{m \geq 1} X/ =_m$$

Furthermore we let $K \times O \to_{her} 2$ consist of the set-theoretic maps $\psi : K \times O \to 2$ such that for any $(m, [x]_m) \in K$, any $o \in O$ and any $0 < n < m$ we have

$$\psi((m, [x]_m), o) \Rightarrow \psi((n, [x]_n), o).$$

To build squash and unsquash we need auxiliary maps:

$$\tfrac{1}{2}(X \to_{ne} \mathit{UPred}(O)) \underset{B}{\overset{H}{\rightleftarrows}} K \times O \to_{her} 2$$

defined by

$$H(\varphi) = \lambda((m, [x]_m), o) \in K \times O.\, \varphi(x) \ni (m-1, o)$$

respectively by

$$B(\psi) = \lambda x \in X.\, \{(m, o) \mid \psi((m+1, [x]_{m+1}), o)\}.$$

These are well-defined. To verify this for H take $\varphi \in \frac{1}{2}(X \to_{ne} \mathit{UPred}(O))$, $(m, [x]_m) \in K$ and $o \in O$. Notice initially that the choice of the representative x does not matter for if $x =_m y$ holds for two $x, y \in X$ we have $\varphi(x) =_m \varphi(y)$ too, in particular $(m-1, o) \in \varphi(x)$ if and only if $(m-1, o) \in \varphi(y)$. To prove $H(\varphi) \in K \times O \to_{her} 2$ we furthermore take $0 < n < m$ and assume that $H(\varphi)((m, [x]_m), o)$ holds, i.e., that $(m-1, o) \in$

$\varphi(x)$. Proving $H(\varphi)((n,[x]_n),o)$ comes down to showing $(n-1,o)\in\varphi(x)$ which is true by uniformity of $\varphi(x)$.

To verify that B is well-defined we take $\psi\in K\times O\to_{her}2$. First we take $x\in X$ and must prove $\{(m,o)\mid \psi((m+1,[x]_{m+1}),o)\}$ uniform. So assume that we have $n<m\in\mathbb{N}$ and $o\in O$ with $\psi((m+1,[x]_{m+1}),o)$, we immediately get $\psi((n+1,[x]_{n+1}),o)$. Second we take $x,y\in X$ with $x=_m y$ for some $m\in\mathbb{N}$, we must show that $B(\psi)(x)=_m B(\psi)(y)$, i.e., that for all $n<m\in\mathbb{N}$ and all $o\in O$ we have $\psi((n+1,[x]_{n+1}),o)$ iff $\psi((n+1,[y]_{n+1}),o)$ but this is immediate since $[x]_{n+1}=[y]_{n+1}$. Here we used Proposition 3.3 to prove non-expansiveness of $B(\psi)$.

Going back and forth with H and B gets you nowhere. For $\psi\in K\times O\to_{her}2$ we get that

$$\begin{aligned}H(B(\psi))&=H(\lambda x.\,\{(m,o)\mid\psi((m+1,[x]_{m+1}),o)\})\\&=\lambda((m,[x]_m),o).\,\psi((m,[x]_m),o)\\&=\psi\end{aligned}$$

and for $\varphi\in\frac{1}{2}(X\to_{ne}UPred(O))$ we get

$$\begin{aligned}B(H(\varphi))&=B(\lambda((m,[x]_m),o).\,\varphi(x)\ni(m-1,o))\\&=\lambda x.\,\{(m,o)\mid\varphi(x)\ni(m,o)\}\\&=\varphi.\end{aligned}$$

Up until this point, the maps H and B have been merely set-theoretic and not morphisms in $\mathsf{CBUlt_{ne}}$, indeed, $K\times O\to_{her}2$ is itself just a set. But now we equip it with the metric induced by the bijection with $\frac{1}{2}(X\to_{ne}UPred(O))$, i.e., the distance between to elements is the distance between the images of these elements under application of B. With this metric we obviously get an object of $\mathsf{CBUlt_{ne}}$ and the maps H and B are morphisms of $\mathsf{CBUlt_{ne}}$, indeed, they are isometries. We need this to be able to apply $\hat{F}$ to them.

Also we need, for each $m\in\mathbb{N}$, to define π_m on $\frac{1}{2}(X\to_{ne}UPred(O))$ by pointwise application of the restriction map, i.e., for $\varphi\in\frac{1}{2}(X\to_{ne}UPred(O))$ we define

$$\pi_m(\varphi)(x)=\varphi(x)|_m.$$

We should verify that this is a non-expansive map. It has been argued above that $\frac{1}{2}(X\to_{ne}UPred(O))$ is bisected so by Proposition 3.3 we take $\varphi_0,\varphi_1\in\frac{1}{2}(X\to_{ne}UPred(O))$, $n\in\mathbb{N}$, assume $\varphi_0=_n\varphi_1$ and aim to prove $\pi_m(\varphi_0)=_n\pi_m(\varphi_1)$. We may without loss of generality assume $n>0$. For $x\in X$ we get by assumption that $\varphi_0(x)=_{n-1}\varphi_1(x)$ which implies that $\varphi_0(x)|_m=_{n-1}\varphi_1(x)|_m$ too and we are done. Really we would like to talk about the maps $(\mathsf{approx}_m)_{m\in\mathbb{N}}$ on $K\times O\to_{her}2$ but we cannot since squash and $\mathsf{unsquash}$ have not been defined yet; instead we deal in $(\pi_m)_{m\in\mathbb{N}}$ on $\frac{1}{2}(X\to_{ne}UPred(O))$. We shall need and prove a close correspondence between the two below.

We are now ready to construct the promised set-theoretic maps squash and $\mathsf{unsquash}$. For $(m,[x]_m)\in K$ we define

$$\mathsf{unsquash}(m,[x]_m)=\left(m-1,(\hat{F}(H)\circ\hat{F}(\pi_{m-1})\circ\Phi)(x)\right)$$

and for $(m,\nu)\in\mathbb{N}\times\hat{F}(K\times O\to_{her}2)$ we set

$$\mathsf{squash}(m,\nu)=\left(m+1,[(\Phi^{-1}\circ\hat{F}(B))(\nu)]_{m+1}\right).$$

Our first aim is to verify that $\mathsf{unsquash}$ is indeed well-defined, i.e., that the choice of the representative x does not matter. For $x,y\in X$ with $x=_m y$ for some $m>0$ we get $\Phi(x)=_m\Phi(y)$ too. For any two $\varphi_0,\varphi_1\in\frac{1}{2}(X\to_{ne}UPred(O))$ we get that if $\varphi_0=_m\varphi_1$ then for any $z\in X$ we have $\varphi_0(z)=_{m-1}\varphi_1(z)$. But then

$$\begin{aligned}\pi_{m-1}(\varphi_0)(z)&=\varphi_0(z)|_{m-1}\\&=\varphi_1(z)|_{m-1}\\&=\pi_{m-1}(\varphi_1)(z)\end{aligned}$$

so we have $\pi_{m-1}(\varphi_0)=\pi_{m-1}(\varphi_1)$. As $\hat{F}$ was assumed non-shrinking, we can now conclude that $\hat{F}(\pi_{m-1})(x)=\hat{F}(\pi_{m-1})(y)$ and we know that $\mathsf{unsquash}$ is well defined.

Before we go on, we need a quick comment on an easily overlooked issue. The maps squash and $\mathsf{unsquash}$ are both set-theoretic as desired but really they go between K and $\mathbb{N}\times U(\hat{F}(K\times O\to_{her}2))$ where $U:\mathsf{CBUlt_{ne}}\to\mathsf{Set}$ is the forgetful functor. But we assumed $\hat{F}$ a lift of F so

$$U(\hat{F}(K\times O\to_{her}2))=F(U(K\times O\to_{her}2))$$

and the latter is what we usually just write $F(K\times O\to_{her}2)$. So the domain respectively codomain of squash and $\mathsf{unsquash}$ really are what they are supposed to be.

With the issues of well-definedness taken care of, we now pursue the promised equalities. For $(m,[x]_m)\in K$ we calculate as follows:

$$\begin{aligned}&(\mathsf{squash}\circ\mathsf{unsquash})(m,[x]_m)\\&\quad=\mathsf{squash}\left(m-1,(\hat{F}(H)\circ\hat{F}(\pi_{m-1})\circ\Phi)(x)\right)\\&\quad=(m,[(\Phi^{-1}\circ\hat{F}(B)\circ\hat{F}(H)\circ\hat{F}(\pi_{m-1})\circ\Phi)(x)]_m)\\&\quad=(m,[(\Phi^{-1}\circ\hat{F}(\pi_{m-1})\circ\Phi)(x)]_m).\end{aligned}$$

A bit of reasoning remains to show that this is indeed $(m,[x]_m)$. Notice first that we may rewrite $x=(\Phi^{-1}\circ\hat{F}(1_{\frac{1}{2}(X\to_{ne}UPred(O))})\circ\Phi)(x)$. This means that if we can prove

$$\pi_{m-1}=_m 1_{\frac{1}{2}(X\to_{ne}UPred(O))}$$

then we are done as $\hat{F}$ was assumed locally non-expansive. So take $\varphi\in\frac{1}{2}(X\to_{ne}UPred(O))$. For any $y\in X$ we get that

$$\pi_{m-1}(\varphi)(y)=\varphi(y)|_{m-1}=_{m-1}\varphi(y)$$

so in $X \to_{ne} UPred(O)$ we have $\pi_{n-1}(\varphi) =_{m-1} \varphi$ and we are done because of the shrinking factor.

For $(m, \nu) \in \mathbb{N} \times \hat{F}(K \times O \to_{her} 2)$ we get

$$\begin{aligned}
&(\mathsf{unsquash} \circ \mathsf{squash})(m, \nu) \\
&\quad = \mathsf{unsquash}\left(m+1, [(\Phi^{-1} \circ \hat{F}(B))(\nu)]_{m+1}\right) \\
&\quad = (m, (\hat{F}(H) \circ \hat{F}(\pi_m) \circ \Phi \circ \Phi^{-1} \circ \hat{F}(B))(\nu)) \\
&\quad = (m, (\hat{F}(H) \circ \hat{F}(\pi_m) \circ \hat{F}(B))(\nu)).
\end{aligned}$$

To finish this we need to look into the relationship between π_m and the map approx_m. Take $\psi \in K \times O \to_{her} 2$, we start from one end and get that

$$\begin{aligned}
&(\pi_m \circ B)(\psi) \\
&\quad = \pi_m(\lambda x.\, \{(n, o) \mid \psi((n+1, [x]_{n+1}), o)\}) \\
&\quad = \lambda x.\, \{(n, o) \mid \psi((n+1, [x]_{n+1}), o) \wedge n < m\} \\
&\quad = \lambda x.\, \{(n, o) \mid \mathsf{approx}_m(\psi)((n+1, [x]_{n+1}), o)\} \\
&\quad = (B \circ \mathsf{approx}_m)(\psi)
\end{aligned}$$

where we remember that $\mathsf{level}(n+1, [x]_{n+1})) = n$ since level is the composite of the first projection and unsquash. Summing up we have proved that

$$\begin{aligned}
&(\mathsf{unsquash} \circ \mathsf{squash})(m, \nu) \\
&\qquad = (m, (F(H) \circ F(B) \circ F(\mathsf{approx}_m))(\nu)) \\
&\qquad = (m, F(\mathsf{approx}_m)(\nu)
\end{aligned}$$

as desired – again we applied that $\hat{F}$ is a lift of F.

We now consider the third property; we need to prove that the subset $K \times O \to_{her} 2$ of the full function space $K \times O \to 2$ coincides with the functions that are hereditary in the sense that they are fixed under application of $\Box$. Take initially $(m, [x]_m)$ and $(n, [y]_n)$ in K, we get that $(m, [x]_m)\, \mathsf{A}\, (n, [y]_n)$ holds iff we have

$$\begin{aligned}
&\mathsf{unsquash}(m, [x]_m) = (l+1, \nu) \wedge (n, [y]_n) = \mathsf{squash}(l, \nu) \\
&\quad \Leftrightarrow (m-1, (\hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi))(x)) = (l+1, \nu) \wedge \\
&\qquad (n, [y]_n) = (l+1, [(\Phi^{-1} \circ \hat{F}(B))(\nu)]_{l+1}) \\
&\quad \Leftrightarrow m = n+1 \wedge \\
&\qquad [y]_n = [(\Phi^{-1} \circ \hat{F}(B) \circ \hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)]_n \\
&\quad \Leftrightarrow m = n+1 \wedge [y]_n = [x]_n.
\end{aligned}$$

Here the last biimplication is a consequence of the fact that $x =_m (\Phi^{-1} \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)$ by arguments used to prove $\mathsf{squash} \circ \mathsf{unsquash} = 1_K$ above. It is immediate from this that for the closure A^* of A we have

$$(m, [x]_m)\, \mathsf{A}^*\, (n, [y]_n) \iff m \geq n \wedge [y]_n = [x]_n.$$

We know by definition that for $\psi \in K \times O \to 2$ we have $\psi = \Box\, \psi$ iff for all $(m, [x]_m) \in K$ and all $o \in O$ we have

$$\psi((m, [x]_m), o) = (\Box\, \psi)((m, [x]_m), o).$$

But by our characterization of A^* we have that the right hand side again equals

$$\begin{aligned}
&\forall (n, [y]_n) \in K.\, (m, [x]_m)\, \mathsf{A}^*(n, [y]_n) \Rightarrow \psi((n, [y]_n), o) \\
&\qquad = \forall n \leq m.\, \psi((n, [x]_n), o).
\end{aligned}$$

From this it is immediate that $\psi = \Box\, \psi$ holds iff we have that $\psi \in K \times O \to_{her} 2$ and we are done. $\square$

Proof of Proposition 3.7. We shall consider only three of the cases.

Constant Non-empty Sets Let X be some fixed non-empty set. Let $F : \mathsf{Set} \to \mathsf{Set}$ be the constant functor mapping any set to X and any function to the identity map 1_X. We need to come up with a plain lift $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ of F. We naturally choose $\hat{F}$ to be the constant functor mapping any object of $\mathsf{CBUlt}_{\mathsf{ne}}$ to X equipped with the discrete metric d_1 and any morphism of $\mathsf{CBUlt}_{\mathsf{ne}}$ to the identity map 1_X. This easily constitutes a locally non-expansive functor $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ and obviously is a lift of F. For any $\varphi : (Y, d) \to (Z, e)$ whatsoever we get that for any $m > 0$ and any two $x, y \in \hat{F}(Y, d) = (X, d_1)$ with $x =_m y$ we have $x = y$, in particular we have that $\hat{F}(\varphi)(x) = 1_X(x) = 1_X(y) = \hat{F}(\varphi)(y)$. Hence $\hat{F}$ is non-shrinking. Finally note that since (X, d_1) is bisected we have that $\hat{F}$ maps all objects to bisected objects, in particular those that were bisected already.

Products Let us consider the case of products; we shall work with binary products only but the construction generalizes to any finite product. Take two functors $F, G : \mathsf{Set} \to \mathsf{Set}$ and define $H : \mathsf{Set} \to \mathsf{Set}$ by mapping a set X to the set $F(X) \times G(X)$ and a map $\varphi : X \to Y$ to $F(\varphi) \times G(\varphi) : F(X) \times G(X) \to F(Y) \times G(Y)$. Under the assumption that we have plain lifts $\hat{F}, \hat{G} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ of F and G, we have to build a plain lift $\hat{H}$ of H.

For an object $(X, d) \in \mathsf{CBUlt}_{\mathsf{ne}}$ we write $(Y, e) = \hat{F}(X, d)$ and $(Z, f) = \hat{G}(X, d)$ and assign

$$\hat{H}(X, d) = (Y \times Z, e \times f),$$

where the product metric $e \times f$ on $Y \times Z$ is defined by $(e \times f)((y_0, z_0), (y_1, z_1)) = \max(e(y_0, y_1), f(z_0, z_1))$ for any two $(y_0, z_0), (y_1, z_1) \in Y \times Z$. For a morphism $\varphi : (X_0, d_0) \to (X_1, d_1) \in \mathsf{CBUlt}_{\mathsf{ne}}$ we write $\hat{F}(\varphi) = (Y_0, e_0) \to (Y_1, e_1)$ and $\hat{G}(\varphi) = (Z_0, f_0) \to (Z_1, f_1)$ and assign

$$\hat{H}(\varphi) = \hat{F}(\varphi) \times \hat{G}(\varphi) : Y_0 \times Z_0 \to Y_1 \times Z_1.$$

It is well known that this yields a well-defined and locally non-expansive functor $\hat{H} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$. For the action on objects, this is spelled out in Lemmas 1.24 and 1.28 of [21].

We now proceed to prove that the functor $\hat{H}$ is indeed a plain lift of H. First up is the property of being a lift, take an object (X, d) of $\mathsf{CBUlt}_{\mathsf{ne}}$. We write $(Y, e) = \hat{F}(X, d)$ and $(Z, f) = \hat{G}(X, d)$ and get that

$$\begin{aligned} U(\hat{H}(X,d)) &= U(Y \times Z, e \times f) \\ &= Y \times Z \\ &= U(\hat{F}(X,d)) \times U(\hat{G}(X,d)) \\ &= F(U(X,d)) \times G(U(X,d)) \\ &= H(U(X,d)). \end{aligned}$$

For a morphism $\varphi : (X_0, d_0) \to (X_1, d_1) \in \mathsf{CBUlt}_{\mathsf{ne}}$ we get the weirdly easy calculation $U(\hat{H}(\varphi)) = \hat{H}(\varphi) = \hat{F}(\varphi) \times \hat{G}(\varphi) = F(\varphi) \times G(\varphi) = H(\varphi)$ since the forgetful functor has no action on morphisms.

Next up is proof that $\hat{H}$ is non-shrinking. Take a morphism $\varphi : (X_0, d_0) \to (X_1, d_1) \in \mathsf{CBUlt}_{\mathsf{ne}}$, we write $\hat{F}(\varphi) = (Y_0, e_0) \to (Y_1, e_1)$ and $\hat{G}(\varphi) = (Z_0, f_0) \to (Z_1, f_1)$. Assume that for some $m > 0$ we have that

$$\forall x, y \in X_0.\ x =_m y \Rightarrow \varphi(x) = \varphi(y).$$

Now take $(y_0, z_0), (y_1, z_1) \in Y_0 \times Z_0$ and assume that we have $(y_0, z_0) =_m (y_1, z_1)$. But then $y_0 =_m y_1$ and $z_0 =_m z_1$ by the definition of the metric $e_0 \times f_0$. And so we have

$$\begin{aligned} \hat{H}(\varphi)(y_0, z_0) &= \left(\hat{F}(\varphi) \times \hat{G}(\varphi)\right)(y_0, z_0) \\ &= \left(\hat{F}(\varphi)(y_0), \hat{G}(\varphi)(z_0)\right) \\ &= \left(\hat{F}(\varphi)(y_1), \hat{G}(\varphi)(z_1)\right) \\ &= \left(\hat{F}(\varphi) \times \hat{G}(\varphi)\right)(y_1, z_1) \\ &= \hat{H}(\varphi)(y_1, z_1) \end{aligned}$$

since both $\hat{F}$ and $\hat{G}$ were assumed non-shrinking. Finally we remark that $\hat{H}$ preserves the property of being bisected since that holds by assumption for $\hat{F}$ and $\hat{G}$ and because the product metric introduces no new distances.

Finite, Partial Maps from a Constant Set Now on to finite, partial maps from a constant set. Take a set X and a functor $F : \mathsf{Set} \to \mathsf{Set}$, define $G : \mathsf{Set} \to \mathsf{Set}$ by mapping a set Y to the set $X \rightharpoonup_{fin} F(Y)$ of partial maps with a finite domain. A map $\varphi : Y \to Z$ is mapped to $\lambda\psi : X \rightharpoonup_{fin} F(Y).\ F(\varphi) \circ \psi$. Under the assumption that we have a plain lift $\hat{F} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ of F, we have to build a plain lift $\hat{G}$ of G.

For an object $(Y, d) \in \mathsf{CBUlt}_{\mathsf{ne}}$ we write $(Z, e) = \hat{F}(Y, d)$ and assign

$$\hat{G}(Y, d) = (X \rightharpoonup_{fin} Z, e_{X \rightharpoonup_{fin}}),$$

where $e_{X \rightharpoonup_{fin}}(\psi_0, \psi_1)$ is $\max_{x \in \mathrm{dom}(\varphi)} e(\psi_0(x), \psi_1(x))$ for any two $\psi_0, \psi_1 : X \rightharpoonup_{fin} Z$ with identical domain, otherwise the distance is 1. For a morphism $\varphi : (Y_0, d_0) \to (Y_1, d_1)$ in $\mathsf{CBUlt}_{\mathsf{ne}}$ we write $\hat{F}(\varphi) : (Z_0, e_0) \to (Z_1, e_1)$ and employ that $\hat{F}$ is a lift of F to simply assign

$$\hat{G}(\varphi) = G(\varphi) : (X \rightharpoonup_{fin} Z_0) \to (X \rightharpoonup_{fin} Z_1).$$

It is easily verifiable – if not exactly well known – that this yields a well-defined and locally non-expansive functor $\hat{G} : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$, a high level argument is given in the proof of Proposition 22 of [18].

We now proceed to prove that the functor $\hat{G}$ is a plain lift of G. First we verify that it is a lift, take an object (Y, d) of $\mathsf{CBUlt}_{\mathsf{ne}}$. We write $(Z, e) = \hat{F}(Y, d)$ and get that

$$\begin{aligned} U(\hat{G}(Y,d)) &= U(X \rightharpoonup_{fin} Z, e_{X \rightharpoonup_{fin}}) \\ &= X \rightharpoonup_{fin} Z \\ &= X \rightharpoonup_{fin} U(\hat{F}(Y,d)) \\ &= X \rightharpoonup_{fin} F(U(Y,d)) \\ &= G(U(Y,d)). \end{aligned}$$

The case of morphisms holds by definition.

Next up is proof that $\hat{G}$ is non-shrinking. Take a morphism $\varphi : (Y_0, d_0) \to (Y_1, d_1) \in \mathsf{CBUlt}_{\mathsf{ne}}$, we write $\hat{F}(\varphi) = (Z_0, e_0) \to (Z_1, e_1)$. Assume that for some $m > 0$ we have that

$$\forall x, y \in Y_0.\ x =_m y \Rightarrow \varphi(x) = \varphi(y).$$

Now take $\psi_0, \psi_1, \in X \rightharpoonup_{fin} Z_0$ and assume that we have $\psi_0 =_m \psi_1$. We have $\mathrm{dom}(\psi_0) = \mathrm{dom}(\psi_1)$ and furthermore know that for all $x \in \mathrm{dom}(\psi_0)$ we have $\psi_0(x) =_m \psi_1(x)$. We obviously have $\mathrm{dom}(\hat{G}(\varphi)(\psi_0)) = \mathrm{dom}(\psi_0) = \mathrm{dom}(\psi_1) = \mathrm{dom}(\hat{G}(\varphi)(\psi_0))$ and for any x in this domain we get

$$\begin{aligned} \hat{G}(\varphi)(\psi_0)(x) &= G(\varphi)(\psi_0)(x) \\ &= (F(\varphi) \circ \psi_0)(x) \\ &= F(\varphi)(\psi_0(x)) \\ &= F(\varphi)(\psi_1(x)) \\ &= (F(\varphi) \circ \psi_1)(x) \\ &= G(\varphi)(\psi_1)(x) \\ &= \hat{G}(\varphi)(\psi_1)(x) \end{aligned}$$

Finally we remark that $\hat{G}$ preserves the property of being bisected since that holds by assumption for $\hat{F}$ and because we introduce no new distances by taking a maximum of finitely many existing distances. □

$$\frac{\Gamma, x \vdash \{P * e \mapsto x\}\text{`}C\text{`}\{Q\}}{\Gamma \vdash \{\exists x.\, P * e \mapsto x\}\text{`let } x = [e] \text{ in } C\text{`}\{Q\}}\ (x \notin \mathrm{fv}(e, Q)) \quad \text{(DEREF)}$$

$$\frac{}{\Gamma \vdash \{e \mapsto \_ * P\}\text{`}[e] := e_0\text{`}\{e \mapsto e_0 * P\}} \quad \text{(UPDATE)}$$

$$\frac{\Gamma, x \vdash \{P * x \mapsto e\}\text{`}C\text{`}\{Q\}}{\Gamma \vdash \{P\}\text{`let } x = \mathsf{new}(e) \text{ in } C\text{`}\{Q\}}\ (x \notin \mathrm{fv}(P, e, Q)) \quad \text{(NEW)}$$

$$\frac{}{\Gamma \vdash \{e \mapsto \_ * P\}\text{`free } e\text{`}\{P\}} \quad \text{(FREE)}$$

$$\frac{}{\Gamma \vdash \{P\}\text{`skip`}\{P\}} \quad \text{(SKIP)}$$

$$\frac{\Gamma \vdash \{P\}\text{`}C_1\text{`}\{R\} \qquad \Gamma \vdash \{R\}\text{`}C_2\text{`}\{Q\}}{\Gamma \vdash \{P\}\text{`}C_1; C_2\text{`}\{Q\}} \quad \text{(SEQ)}$$

$$\frac{\Gamma \vdash \{P \wedge e_1 = e_2\}\text{`}C_1\text{`}\{Q\} \qquad \Gamma \vdash \{P \wedge e_1 \neq e_2\}\text{`}C_2\text{`}\{Q\}}{\Gamma \vdash \{P\}\text{`if } (e_1 = e_2) \text{ then } C_1 \text{ else } C_2\text{`}\{Q\}} \quad \text{(IF)}$$

$$\frac{\Gamma \vdash P' \Rightarrow P \qquad \Gamma \vdash Q \Rightarrow Q'}{\Gamma \vdash \{P\}e\{Q\} \Rightarrow \{P'\}e\{Q'\}} \quad \text{(CONSEQ)}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \otimes Q} \quad (\otimes\text{-FRAME})$$

$$\frac{}{\Gamma \vdash \{P\}e\{Q\} \Rightarrow \{P * R\}e\{Q * R\}} \quad (*\text{-FRAME})$$

$$\frac{\Gamma, k \vdash R[k] \Rightarrow \{P * e \mapsto R[\_]\}k\{Q\}}{\Gamma \vdash \{P * e \mapsto R[\_]\}\text{`eval } [e]\text{`}\{Q\}} \quad \text{(EVAL)}$$

Figure 5. Proof rules for Hoare triples.

C Step-Indexed Model of Separation Logic for Nested Hoare Triples

In this section we include additional definitions and proofs for the step-indexed model of separation logic for nested Hoare triples.

The operational semantics is specified in Figure 4.

See Figure 5 for additional separation logic proof rules.

The interpretation of assertions is written out in full in Figure 6.

$$
\begin{array}{rcll}
([e_1] := e_2,\ h) & \rightsquigarrow & (\mathsf{skip}, h[n \mapsto [\![e_2]\!]) & \text{if } [\![e_1]\!] = n \text{ and } n \in \mathrm{dom}(h) \\
(\mathsf{let}\ x = [e]\ \mathsf{in}\ C,\ h) & \rightsquigarrow & (C[v/x], h) & \text{if } [\![e]\!] = n \text{ and } h(n) = v \\
(\mathsf{eval}\ [e],\ h) & \rightsquigarrow & (C, h) & \text{if } [\![e]\!] = n \text{ and } h(n) = \text{`}C\text{`} \\
(\mathsf{let}\ x = \mathsf{new}(e)\ \mathsf{in}\ C,\ h) & \rightsquigarrow & (C[n/x],\ h * [n \mapsto [\![e]\!]]) & \text{if } n \notin \mathrm{dom}(h) \\
(\mathsf{free}\ e,\ h) & \rightsquigarrow & (\mathsf{skip}, h') & \text{if } [\![e]\!] = n \text{ and } h = h' * [n \mapsto v] \\
(C_1; C_2,\ h) & \rightsquigarrow & (C_1'; C_2,\ h') & \text{if } (C_1, h) \rightsquigarrow (C_1', h') \\
(\mathsf{skip}; C_2,\ h) & \rightsquigarrow & (C_2, h) & \\
(\mathsf{if}\ (e_1 = e_2)\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2,\ h) & \rightsquigarrow & (C_1, h) & \text{if } [\![e_1]\!] = [\![e_2]\!] \\
(\mathsf{if}\ (e_1 = e_2)\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2,\ h) & \rightsquigarrow & (C_2, h) & \text{if } [\![e_1]\!] \neq [\![e_2]\!] \\
\\
([e_1] := e_2,\ h) & \rightsquigarrow & \mathsf{abort} & \text{if } [\![e_1]\!] = \text{`}C\text{`} \text{ or } [\![e_1]\!] = n \text{ where } n \notin \mathrm{dom}(h) \\
(\mathsf{let}\ x = [e]\ \mathsf{in}\ C,\ h) & \rightsquigarrow & \mathsf{abort} & \text{if } [\![e]\!] = \text{`}C\text{`} \text{ or } [\![e]\!] = n \text{ where } n \notin \mathrm{dom}(h) \\
(\mathsf{eval}\ [e],\ h) & \rightsquigarrow & \mathsf{abort} & \text{if } [\![e]\!] = \text{`}C\text{`} \text{ or } [\![e]\!] = n \text{ where } n \notin \mathrm{dom}(h) \\
(\mathsf{eval}\ [e],\ h) & \rightsquigarrow & \mathsf{abort} & \text{if } [\![e]\!] = n \text{ where } h(n) = m \\
(\mathsf{free}\ e,\ h) & \rightsquigarrow & \mathsf{abort} & \text{if } [\![e]\!] = \text{`}C\text{`} \text{ or } [\![e]\!] = n \text{ where } n \notin \mathrm{dom}(h) \\
(C_1; C_2,\ h) & \rightsquigarrow & \mathsf{abort} & \text{if } (C_1, h) \rightsquigarrow \mathsf{abort}
\end{array}
$$

Figure 4. Operational semantics.

$$
\begin{aligned}
[\![\mathsf{false}]\!]_\eta w &= \emptyset \\
[\![\mathsf{true}]\!]_\eta w &= \mathbb{N} \times H \\
[\![P \wedge Q]\!]_\eta w &= [\![P]\!]_\eta w \cap [\![Q]\!]_\eta w \\
[\![P \vee Q]\!]_\eta w &= [\![P]\!]_\eta w \cup [\![Q]\!]_\eta w \\
[\![P \Rightarrow Q]\!]_\eta w &= \{(n,h) \mid \forall m \leq n.\, (m,h) \in [\![P]\!]_\eta w \Rightarrow (m,h) \in [\![Q]\!]_\eta w \\
[\![\forall x.P]\!]_\eta w &= \textstyle\bigcap_{v \in V} [\![P]\!]_{\eta[x \mapsto v]} w \\
[\![\exists x.P]\!]_\eta w &= \textstyle\bigcup_{v \in V} [\![P]\!]_{\eta[x \mapsto v]} w \\
[\![\mathsf{int}(e)]\!]_\eta w &= \begin{cases} \mathbb{N} \times H & \text{if } [\![e]\!]_\eta = m \text{ for some } m \in \mathbb{Z} \\ \emptyset & \text{otherwise} \end{cases} \\
[\![e_1 = e_2]\!]_\eta w &= \begin{cases} \mathbb{N} \times H & \text{if } [\![e_1]\!]_\eta = [\![e_2]\!]_\eta \\ \emptyset & \text{otherwise} \end{cases} \\
[\![e_1 \leq e_2]\!]_\eta w &= \begin{cases} \mathbb{N} \times H & \text{if } [\![e_1]\!]_\eta = m_1 \text{ and } [\![e_2]\!]_\eta = m_2 \text{ where } m_1 \leq m_2 \\ \emptyset & \text{otherwise} \end{cases} \\
[\![e_1 \mapsto e_2]\!]_\eta w &= \begin{cases} \{(n, [m \mapsto [\![e_2]\!]_\eta]) \mid n \in \mathbb{N}\} & \text{if } [\![e_1]\!]_\eta = m \text{ for some } m \in \mathbb{Z} \\ \emptyset & \text{otherwise} \end{cases} \\
[\![\mathsf{emp}]\!]_\eta w &= \mathbb{N} \times \{[\,]\} \\
[\![P * Q]\!]_\eta w &= [\![P]\!]_\eta w * [\![Q]\!]_\eta w \\
[\![P \mathrel{-\!\!*} Q]\!]_\eta w &= [\![P]\!]_\eta w \mathrel{-\!\!*} [\![Q]\!]_\eta w \\
[\![\{P\}e\{Q\}]\!]_\eta w &= \begin{cases} \{(n,h) \mid w \models_n ([\![P]\!]_\eta, C, [\![Q]\!]_\eta)\,\} & \text{if } [\![e]\!]_\eta = \text{`}C\text{`} \\ \emptyset & \text{otherwise} \end{cases} \\
[\![P \otimes Q]\!]_\eta w &= \Big([\![P]\!]_\eta \otimes i([\![Q]\!]_\eta)\Big)\, w \\
[\![(\mu\alpha(x).P)(e)]\!]_\eta w &= \ldots \\
[\![\alpha(e)]\!]_\eta w &= \ldots
\end{aligned}
$$

Figure 6. Interpretation of assertions.

Chapter 7

Two for the Price of One: Lifting Separation Logic Assertions

Two for the Price of One: Lifting Separation Logic Assertions

Jacob Thamsborg and Lars Birkedal
IT University of Copenhagen

Hongseok Yang
Queen Mary University of London

Abstract

Recently, data abstraction has been studied in the context of separation logic, with noticeable practical successes: the developed logics have enabled clean proofs of tricky challenging programs, such as subject-observer patterns, and they have become the basis of efficient verification tools for Java (jStar), C (VeriFast) and Hoare Type Theory (Ynot). In this paper, we give a new semantic analysis of such logic-based approaches using Reynolds's relational parametricity. The core of the analysis is our lifting theorems, which give a sound and complete condition for when a true implication between assertions in the standard interpretation entails that the same implication holds in a relational interpretation. Using these theorems, we provide an algorithm for identifying abstraction-respecting client-side proofs; the proofs ensure that clients cannot distinguish two appropriately-related module implementations.

1 Introduction

Data abstraction is one of the key design principles for building computer software, and it has been the focus of active research from the early days of computer science. Recently, data abstraction has been studied in the context of separation logic [14, 4, 11, 15, 7], with noticeable practical successes: the developed logics have enabled clean proofs of tricky challenging programs, such as the subject-observer pattern, and they have become the basis of efficient verification tools for Java (jStar [8]), C (VeriFast [10]) and Hoare Type Theory (Ynot [12]).

In this paper, we give a new semantic analysis of these logic-based approaches using Reynolds's relational parametricity. Our techniques can be used to prove *representation independence*, i.e., that clients cannot distinguish between related module implementations, a consequence that we would expect from using data abstraction, but (as we shall see) a consequence that only holds for certain good clients.

Interface Specification

$\{1\hookrightarrow\_\}\texttt{init}\{a\}$ $\{a\}\texttt{nxt}\{b\}$ $\{b\}\texttt{fin}\{1\hookrightarrow\_\}$
$\{a\}\texttt{inc}\{a\}$ $\{b\}\texttt{dec}\{b\}$

Two Implementations of a Counter

$\texttt{init}_1 \stackrel{def}{=} [1]:=0$ $\texttt{nxt}_1 \stackrel{def}{=} \texttt{skip}$ $\texttt{fin}_1 \stackrel{def}{=} \texttt{skip}$
$\texttt{inc}_1 \stackrel{def}{=} [1]:=[1]+1$ $\texttt{dec}_1 \stackrel{def}{=} [1]:=[1]-1$

$\texttt{init}_2 \stackrel{def}{=} [1]:=0$ $\texttt{nxt}_2 \stackrel{def}{=} [1]:=-[1]$ $\texttt{fin}_2 \stackrel{def}{=} [1]:=-[1]$
$\texttt{inc}_2 \stackrel{def}{=} [1]:=[1]+1$ $\texttt{dec}_2 \stackrel{def}{=} [1]:=[1]+1$

Definitions of Abstract Predicates

$a_1 \stackrel{def}{=} 1\hookrightarrow\_,\ b_1 \stackrel{def}{=} 1\hookrightarrow\_, \quad a_2 \stackrel{def}{=} 1\hookrightarrow\_,\ b_2 \stackrel{def}{=} 1\hookrightarrow\_$

Client-side Proof Attempts

$\{1\hookrightarrow\_\}$ `init`; $\{a\}$
`inc`; $\{a\}$
`nxt`; $\{b\}$
`dec`; $\{b\}$
`fin` $\{1\hookrightarrow\_\}$

$\{1\hookrightarrow\_\}$ `init`; $\{a\}$
`inc`; $\{a\}$
`nxt`; $\{b\}$
$[1]:=[1]-1$; $\{???\}$

Figure 1. Two-stage Counter

Logic-based Data Abstraction The basic idea of the logic-based approaches is that the private states of modules are exposed to clients only abstractly using *assertion variables* [4], also known as *abstract predicates* [14]. For concreteness, we consider a two-stage counter module and client programs in Figure 1. The module realizes a counter with increment and decrement operations, called `inc` and `dec`. An interesting feature is that the counter goes through two stages in its lifetime; in the first stage, it can perform only the increment operation, but in the second, it can only run the decrement. The *interface specification* in the figure formalizes this intended behavior of the counter using assertion variables a and b, where a means that the counter is in the first stage and b that the counter is in the second. The triple for `init` says that the initialization can turn the assertion $1\hookrightarrow\_$, denoting heaps with cell 1, to the assertion variable a, which describes

an abstract state where we can only call `inc` or `nxt` (since a is the precondition of only those operations). The abstract state a can be changed to b by calling `nxt`, says the triple for the `nxt` operation. In b we are allowed to run `dec` but not `inc`. Finally, `fin` can turn the abstract state b back to $1{\hookrightarrow}\_$. Note that by using a and b, the interface specification does not expose the private state of the module to the client. It reveals only *partial information* about the private state of the module; here it is whether the private state is in the first or the second stage. The flexibility afforded by revealing partial information is very useful in applications; see the examples mentioned in the references above.

In these logic-based approaches, proof attempts for clients of a module can succeed only when they are given with respect to the abstract interface specification, without making any further assumptions on assertion variables. For instance, the proof attempt on the bottom left of Figure 1 is successful, whereas the bottom right one is not, because the latter assumes that the assertion variable b entails the allocatedness of cell 1. This is so, even when the entailment holds for an actual definition of b.

Representation Independence In this paper, we analyze the condition for successful client-side proofs, using *representation independence*: given two implementations of a module, *if* there are coupling relations between the definitions of assertion variables in the two modules and, furthermore, the relations are preserved by the corresponding operations of the implementations, *then* a *proved* client will give the *same* result no matter whether it is executed with one or the other module implementation. For instance, Figure 1 describes two implementations of the counter, where cell 1 is used in both to represent their private states, but in different ways — the first stores the current value of the counter, but the second stores the current value or its negative version, depending on whether it is in the first stage or the second. In both cases, the assertion variables a and b are defined to be the same $1{\hookrightarrow}\_$. The operations of the two implementations preserve the coupling relations r_a and r_b:

$$r_a \stackrel{def}{=} \{(h_1,h_2) \mid 1{\in}\,\mathrm{dom}(h_1){\cap}\,\mathrm{dom}(h_2) \wedge h_1(1){=}h_2(1)\}$$
$$r_b \stackrel{def}{=} \{(h_1,h_2) \mid 1{\in}\,\mathrm{dom}(h_1){\cap}\,\mathrm{dom}(h_2) \wedge h_1(1){=}{-}h_2(1)\}$$

Thus, the representation independence result says that all proved clients, such as the one on the left bottom of Figure 1, should behave the same for both implementations.

In earlier work [6] we were able to prove such a representation independence result for a more restricted form of logical data abstraction, namely one given by frame rules rather than general assertion variables. Roughly speaking, frame rules use a restricted form of assertion variables that are not exposed to clients at all, as can be seen from some models of separation logic in which frame rules are modelled via quantification over semantic assertions [5]. This means that the rules do not allow the exposure of even partial information about module internals. (On the other hand, frame rules implement information hiding, because they completely relieve clients of tracking the private state of a module, even in an abstracted form.) Our model in [6] exploited this restricted use of assertion variables, and gave relational meanings to Hoare triple specifications, which led to representation independence.

Removing this restriction and allowing assertion variables in client proofs turned out to be very challenging. The main problem was that it was unclear under which conditions a standard unary implication between assertions in the rule of consequence could be lifted to a *relational* one. "Always" is not an answer, because only some, not all, implications can be lifted. In this paper, we provide a sound and, in a certain sense, complete answer to when the lifting can be done.

For instance, consider the example in Figure 2. Our answer lets us conclude that the client on the left is *good* but the one on the right is *bad*. The client on the left calls `init` and ends with the post-condition $(1{\hookrightarrow}\_ \wedge a * b)$. Since $(1{\hookrightarrow}\_ \wedge a * b) \implies a$ is true in the standard interpretation (in an intuitionistic setup[1]),

Interface Specification

$\{1{\hookrightarrow}\_\}\texttt{init}\{1{\hookrightarrow}\_ \wedge a * b\}$ $\quad$ $\{a\}\texttt{fin}\{1{\hookrightarrow}\_\}$

$\{1{\hookrightarrow}\_ * a \vee 1{\hookrightarrow}\_ * b\}\texttt{badfin}\{1{\hookrightarrow}\_\}$

Two Implementations

$\texttt{init}_1 \stackrel{def}{=} \texttt{skip}$ $\quad$ $\texttt{fin}_1 \stackrel{def}{=} \texttt{skip}$ $\quad$ $\texttt{badfin}_1 \stackrel{def}{=} [1]{:=}1$

$\texttt{init}_2 \stackrel{def}{=} \texttt{skip}$ $\quad$ $\texttt{fin}_2 \stackrel{def}{=} \texttt{skip}$ $\quad$ $\texttt{badfin}_2 \stackrel{def}{=} [1]{:=}2$

Definitions of Abstract Predicates

$a_1 \stackrel{def}{=} 1{\hookrightarrow}\_$, $\quad$ $b_1 \stackrel{def}{=} \mathsf{true}$, $\quad$ $a_2 \stackrel{def}{=} 1{\hookrightarrow}\_$, $\quad$ $b_2 \stackrel{def}{=} \mathsf{true}$

Two Client-side Proofs

$\{1{\hookrightarrow}\_\}$	$\{1{\hookrightarrow}\_\}$
`init`;	`init`;
$\{1{\hookrightarrow}\_ \wedge a * b\}$	$\{1{\hookrightarrow}\_ \wedge a * b\}$
$\{a\}$	$\{1{\hookrightarrow}\_ * a \vee 1{\hookrightarrow}\_ * b\}$
`fin`	`badfin`
$\{1{\hookrightarrow}\_\}$	$\{1{\hookrightarrow}\_\}$

Figure 2. Good or Bad Client-side Proofs

[1] In an intuitionistic setup, $\varphi * \psi \implies \varphi$ holds for all φ, ψ.

the rule of consequence can be applied to yield the precondition of `fin`, which can be called, ending up with the postcondition $(1\hookrightarrow\_)$. The key point here is the implication used in the rule of consequence. Our results imply that this implication can indeed be lifted to an implication between relational meanings of assertions $(1\hookrightarrow\_ \wedge\ a * b)$ and a (Theorem 8 in Section 4). They also entail that this lifting implies the representation independence theorem. We can confirm the claim of the theorem, simply noting that both $\texttt{init}_1;\texttt{fin}_1$ and $\texttt{init}_2;\texttt{fin}_2$ are the same skip command.

The client on the right also first calls `init` and then uses the rule of consequence. But this time our results say that a true implication $(1\hookrightarrow\_ \wedge\ a * b) \implies (1\hookrightarrow\_ * a \vee 1\hookrightarrow\_ * b)$ in the rule of consequence does *not* lift to an implication between relational meanings of the assertions (Theorem 15 in Section 5 or just Example 12.) Because of this failure, the proof of the client does not ensure representation independence. In fact, the client can indeed distinguish between the two module implementations — when the client is executed with the first module implementation, the final heap maps address 1 to 1, but when the client is executed with the second, the final heap maps address 1 to 2.

Note that we phrase the lifting only in terms of semantically true implications, without referring to how they are proved. By doing so, we make our results relevant to automatic tools that use the semantic model of separation logic to prove implications, such as the ones based on shallow embeddings of the assertion logic [12, 9].

To sum up, the question of whether representation independence holds for a client comes down to whether the implications used in the rule of consequence in a client-side proof can be lifted to a relational interpretation. In this paper, we give a sound and complete characterization of when that is the case.

2 Semantic Domain

In the following section we will define the meaning of an assertion to be an n-ary relation on heaps. To formalize this relational meaning, we need a semantic domain IRel_n of relations, which we define and explain in this section.

Let Heap be the set of finite partial functions from positive integers to integers (i.e., $\mathsf{Heap} \stackrel{def}{=} \mathsf{PosInt} \rightarrow_{fin} \mathsf{Int}$), ranged over by f, g, h. This is a commonly used set for modelling heaps in separation logic, and it has a partial commutative monoid structure $([], \cdot)$, where $[]$ is the empty heap and the $\cdot$ operator combines disjoint heaps:

$$[] \stackrel{def}{=} \emptyset, \qquad f \cdot g \stackrel{def}{=} \begin{cases} f \cup g & \text{if } \mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

The operator $\cdot$ induces a partial order $\sqsubseteq$ on Heap, modelling heap extension, by $f \sqsubseteq g$ iff $g = f \cdot h$ for some h.

We also consider the $+$ operator for combining possibly-overlapping but consistent heaps, and the $-$ operator for subtracting one heap from another:

$$f + g \stackrel{def}{=} \begin{cases} f \cup g & \text{if } \forall l \in \mathrm{dom}(f) \cap \mathrm{dom}(g).\ f(l) = g(l) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$(f - g)(l) \stackrel{def}{=} \begin{cases} f(l) & \text{if } l \in \mathrm{dom}(f) \setminus \mathrm{dom}(g) \\ \text{undefined} & \text{otherwise} \end{cases}$$

We call an n-ary relation $r \subseteq \mathsf{Heap}^n$ *upward closed* iff $(f_1, \ldots, f_n) \in r \wedge (\forall i.\ f_i \sqsubseteq g_i) \implies (g_1, \ldots, g_n) \in r$.

Definition 1 IRel_n *is the family of upward closed n-ary relations on heaps.*

Note that IRel_1 consists of upward closed *sets* of heaps, which are frequently used to interpret assertions in separation logic for garbage-collected languages. We call elements of IRel_1 *predicates* and denote them by p, q.

For every $n \geq 1$, domain IRel_n has a complete lattice structure: join and meet are given by union and intersection, bottom is the empty relation, and top is Heap^n. The domain also has a semantic separating conjunction connective defined by

$$(f_1, .., f_n) \in r * s \stackrel{def}{\iff} \exists (g_1, .., g_n) \in r.\ \exists (h_1, .., h_n) \in s.\ (f_1, .., f_n) \cdot (g_1, .., g_n) = (h_1, .., h_n).$$

Here we use the component-wise extension of $\cdot$ for tuples. Intuitively, a tuple is related by $r * s$ when it can be splitted into two disjoint tuples, one related by r and the other by s.

The domain IRel_1 of predicates is related to IRel_n for every n, by the map $\Delta_n \stackrel{def}{=} \lambda p.\{(f, \ldots, f) \mid f \in p\}^{\uparrow}$, where $\uparrow$ is the upward closure on relations. Note that each predicate is turned into an n-ary identity relation on p modulo the upward closure. This map behaves well with respect to the structures discussed on IRel_1 and IRel_n, as expressed by the lemma below:

Lemma 2 *Function Δ_n preserves the complete lattice structure and the $*$ operator.*

3 Assertions and Relational Semantics

Let Var and AVar be disjoint sets of normal variables $x, y, ...$ and assertion variables $a, b, ...$, respectively. Our assertions φ are from higher-order separation logic, and they conform to the following grammar:

$$E ::= x \mid 0 \mid 1 \mid E + E \mid \ldots \qquad P ::= E \hookrightarrow E \mid \ldots$$
$$\varphi ::= P \mid a \mid \varphi * \varphi \mid \mathsf{true} \mid \varphi \wedge \varphi \mid \mathsf{false} \mid \varphi \vee \varphi \mid \forall x.\ \varphi \mid \exists x.\ \varphi$$

$$\begin{array}{ll}
[\![P]\!]^n_{\eta,\rho} \stackrel{def}{=} \Delta_n((\!|P|\!)_\eta) & [\![\varphi * \psi]\!]^n_{\eta,\rho} \stackrel{def}{=} [\![\varphi]\!]^n_{\eta,\rho} * [\![\psi]\!]^n_{\eta,\rho} \\
[\![a]\!]^n_{\eta,\rho} \stackrel{def}{=} \rho(a) & [\![\varphi \wedge \psi]\!]^n_{\eta,\rho} \stackrel{def}{=} [\![\varphi]\!]^n_{\eta,\rho} \cap [\![\psi]\!]^n_{\eta,\rho} \\
[\![\mathsf{true}]\!]^n_{\eta,\rho} \stackrel{def}{=} \mathsf{Heap}^n & [\![\varphi \vee \psi]\!]^n_{\eta,\rho} \stackrel{def}{=} [\![\varphi]\!]^n_{\eta,\rho} \cup [\![\psi]\!]^n_{\eta,\rho} \\
[\![\mathsf{false}]\!]^n_{\eta,\rho} \stackrel{def}{=} \emptyset & [\![\forall x.\varphi]\!]^n_{\eta,\rho} \stackrel{def}{=} \bigcap_{v\in\mathsf{Int}} [\![\varphi]\!]^n_{\eta[x\mapsto v],\rho} \\
 & [\![\exists x.\varphi]\!]^n_{\eta,\rho} \stackrel{def}{=} \bigcup_{v\in\mathsf{Int}} [\![\varphi]\!]^n_{\eta[x\mapsto v],\rho}
\end{array}$$

where $(\!|P|\!)_\eta$ is the standard semantics of P as an upward closed set of heaps, which satisfies:

$$(\!|E{\hookrightarrow}F|\!)_\eta \;=\; \{f \mid [\![E]\!]_\eta \in \mathrm{dom}(f) \;\wedge\; f([\![E]\!]_\eta) = [\![F]\!]_\eta\}.$$

Figure 3. Interpretation of Assertions

In the grammar, E is a heap-independent expression, and P is a primitive predicate, which in the standard interpretation denotes an upward closed set of heaps. For instance, $E{\hookrightarrow}E'$ means heaps containing cell E with contents E'. Formulas φ are assertions in higher-order separation logic that do not include negation nor implication. Note that these assertions can include assertion variables, thus called higher-order. The dots in the grammar indicate possible extensions of cases, such as multiplication for E and inductive predicates for P. We will use the abbreviation $E{\hookrightarrow}\_$ for $\exists y.E{\hookrightarrow}y$.

An assertion φ is given a meaning $[\![\varphi]\!]^n_{\eta,\rho} \in \mathsf{IRel}_n$ as an n-ary relation on heaps, where n is a parameter of the semantics. Here environment η maps normal variables in φ to integers, and ρ maps assertion variables in φ to n-ary relations in IRel_n. When φ does not contain any assertion variables, we often omit ρ and write $[\![\varphi]\!]^n_\eta$, because the meaning of φ does not depend on ρ.

We define the semantics of φ, using the complete lattice structure and the $*$ operator of the domain IRel_n; see Figure 3. Note that the relational semantics of primitive predicates is defined by embedding their standard meanings via Δ_n. In fact, this embedding relationship holds for all assertions, because Δ_n preserves the semantic structures of the domains (Lemma 2).

Lemma 3 *For all φ and η, ρ, ρ', if $\Delta_n(\rho(a)) = \rho'(a)$ for every $a \in \mathsf{AVar}$, we have that $\Delta_n([\![\varphi]\!]^1_{\eta,\rho}) = [\![\varphi]\!]^n_{\eta,\rho'}$.*

We write $\varphi \models^n \psi$ to mean that $[\![\varphi]\!]^n_{\eta,\rho} \subseteq [\![\psi]\!]^n_{\eta,\rho}$ holds for all environments η, ρ. If n=1, this reduces to the standard semantics of assertions in separation logic. We will use the phrase "$\varphi \Longrightarrow \psi$ is n-ary valid" to mean that $\varphi \models^n \psi$ holds. In addition, we write $\varphi \models^n_\eta \psi$ for a fixed η to mean that $[\![\varphi]\!]^n_{\eta,\rho} \subseteq [\![\psi]\!]^n_{\eta,\rho}$ holds for all environments ρ; we say that "$\varphi \Longrightarrow \psi$ is n-ary η-valid" if this is true.

4 Lifting Theorems and Completeness

We call an assertion φ *simple* if it is of the form $(\bigvee_{i=1}^{I} \bigwedge_{j=1}^{J} \varphi_{(i,j)} * \mathbf{a}_{(i,j)})$, where $\mathbf{a}_{(i,j)}$ is a vector of assertion variables and $\varphi_{i,j}$ is an assertion not containing any assertion variables. We will consider the question of lifting an implication between simple assertions φ, ψ to a binary relational interpretation: when does $\varphi \models^1 \psi$ imply that $\varphi \models^2 \psi$?

The starting point is to realize that it is sufficient to study implications of the form:

$$\bigwedge_{i=1}^{M} \varphi_i * a_{i,1} * \cdots * a_{i,M_i} \implies \bigvee_{j=1}^{N} \psi_j * b_{j,1} * \cdots * b_{j,N_j} \quad (1)$$

where φ_i's and ψ_j's do not contain assertion variables, and no assertion variables occur only on the right hand side of the implication.

Lemma 4 *There is an algorithm taking simple assertions φ, ψ and returning finitely many implications $\{\varphi^l \Longrightarrow \psi^l\}_{l\in L}$, such that (a) $\varphi^l \Longrightarrow \psi^l$ has the form (1) and (b) for any $n \in \{1, 2\}$, we have that $\varphi \models^n \psi$ holds iff $\varphi^l \models^n \psi^l$ holds for all $l \in L$.*

The algorithm in the lemma is given in Appendix B.

Thus, in this section, we will focus on lifting implications of the form (1). Specifically, we will give a *complete* answer to the following question: Given one such implication that is η-valid in the unary interpretation for some environment η, can we decide if the implication is η-valid in the binary interpretation *merely* by inspection of the layout of the assertion variables? The answer will come in two parts. The first part, in Section 4.2, provides three lifting theorems, each of which has a criterion on the variable layout that, if met, implies that η-validity may be lifted regardless of the φ_i's and ψ_j's. The second part, in Section 4.3, is a completeness theorem; it states that if the variables fail the criteria of all three lifting theorems then there are choices of φ_i's and ψ_j's with no variables such that we have unary but not binary validity.

This approach has ups and downs. Assume that we have an implication of the aforementioned form that is valid in the unary interpretation, and we would like to know if it is valid in the binary interpretation too. Trying out the layout of the variables against the criteria of the three lifting theorems is an easily decidable and purely syntactical process – and if it succeeds then we have binary validity. If it fails, however, we are at a loss; we know that there are φ_i's and ψ_j's with the same variable layout such that lifting fails but we do not learn anything about our concrete implication. There is, however, an alternate use of the theory below if the

lifting criteria fail: We can sidestep the general lifting theorems and try to verify directly the Parametricity Condition from Definition 6 for all environments η. It is a semantic condition and probably undecidable in general, but it involves no assertion variables and only unary semantics – and if it holds then, by Proposition 7, we have binary validity. Indeed, the lifting theorems with direct verification of the Parametricity Condition as backup can be thought of as an algorithm. It is complete in some cases: Theorem 15 expresses that binary η-validity and the Parametricity Condition imply each other if $M_i \leq 2$ for all $1 \leq i \leq M$.

4.1 Notation

We need some notation that will accompany us throughout this section. Consider an implication of the form (1). Let $V = \bigcup_{i=1}^{M} \{a_{i,1}, \ldots, a_{i,M_i}\}$ be the set of all left hand side assertion variables, these include the right hand side assertion variables too by assumption. Define $\Pi : \{1, \ldots, M\} \to \mathsf{Nat}^V$ and $\Omega : \{1, \ldots, N\} \to \mathsf{Nat}^V$ by the following:

$$\Pi(i)(c) \stackrel{def}{=} |\{k \mid a_{i,k} \equiv c\}|, \quad \Omega(j)(c) \stackrel{def}{=} |\{k \mid b_{j,k} \equiv c\}|.$$

These functions give vectors of assertion variable counts for each conjunct and disjunct. For $1 \leq i \leq M$ and $1 \leq j \leq N$ we write $\Pi(i) \geq \Omega(j)$ if we have $\Pi(i)(c) \geq \Omega(j)(c)$ for each variable $c \in V$, i.e., if conjunct i has the same or a greater number of occurrences of all variables than disjunct j. We write $\Pi(i) \not\geq \Omega(j)$ if this fails, i.e., if there is $c \in V$ such that $\Pi(i)(c) < \Omega(j)(c)$. If a conjunct, say conjunct i, has no variables, i.e., if $\Pi(i)(c) = 0$ holds for all $c \in V$, then we say it *empty*; the same goes for the disjuncts.

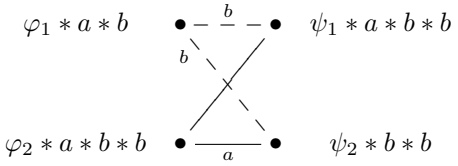
We shall write $-$ to denote $\exists n, m.\ n{\hookrightarrow}m$, meaning heaps with at least one cell. On the semantic side, we write $[m]$ for $m \in \mathsf{PosInt}$ to denote the heap that stores 0 at location m and nothing else. For $m_0, ..., m_n \in \mathsf{PosInt}$ different we write $[m_0, ..., m_n]$ for $[m_0] \cdot ... \cdot [m_n]$.

Finally we introduce a piece of sanity-preserving graphical notation. We depict an implication of the form (1) as a complete bipartite graph with the conjuncts lined up on the left hand side and the disjuncts on the right hand side. For any $1 \leq i \leq M$ and any $1 \leq j \leq N$ we draw a solid line from conjunct i to disjunct j if $\Pi(i) \geq \Omega(j)$. We label that line with all the $c \in V$ such that $\Pi(i)(c) > \Omega(j)(c)$ if indeed there are any such. If, on the other hand, $\Pi(i) \not\geq \Omega(j)$ then we draw a dashed line instead and label it with all the $c \in V$ such that $\Pi(i)(c) < \Omega(j)(c)$. Note that the drawing of edges depend solely on the layout of the variables; the φ_i's and ψ_j's have no say in the matter. As examples we refer the reader to Examples 12 and 13 for graphical representations of $1{\hookrightarrow}\_ \wedge a * b \implies 1{\hookrightarrow}\_ * a \vee 1{\hookrightarrow}\_ * b$ and $- * a * b \wedge a * a \implies - * a * a \vee - * - * b$ respectively.

4.2 Layouts that Lift

The following is a first example of a layout of variables that ensure that for any choice of φ_i's and ψ_j's we get that unary η-validity of the implication yields binary η-validity. That it holds is a consequence of Theorem 8 but we have spelled out a concrete proof that will serve as a guide to the further development.

Example 5 (Shadow-Lift) For any four assertions $\varphi_1, \varphi_2, \psi_1, \psi_2$ with no assertion variables and any appropriate environment η we have that unary η-validity of the following implication implies binary η-validity:



Assume that we have unary η-validity. Before we go on to consider the binary case we derive a simple unary consequence that does not involve assertion variables: For any $h \in \mathsf{Heap}$ with subheaps $h_1 \sqsubseteq h$ and $h_2 \sqsubseteq h$ such that $h_1 \in [\![\varphi_1]\!]^1_\eta$ and $h_2 \in [\![\varphi_2]\!]^1_\eta$ we get that $h_2 \in [\![\psi_1]\!]^1_\eta$ or that $h_2 \in [\![\psi_2]\!]^1_\eta$.

To prove this, let h, h_1 and h_2 be as assumed. We build $\rho : \{a, b\} \to \mathsf{IRel}_1$ by letting $\rho(a) = \mathsf{Heap}$ and letting $\rho(b)$ be the following union of sets of heaps:

$$\{(h - h_1) \cdot [n, n+1]\}^\uparrow \cup \{(h - h_2) \cdot [n]\}^\uparrow \cup \{[n+1]\}^\uparrow$$

where $n = \max(\mathrm{dom}(h) \cup \{0\}) + 1$. It is now immediate that $h \cdot [n, n+1]$ lies in the interpretation of both conjuncts and by our assumption on the original implication it must lie in the interpretation on one of the disjuncts too. Suppose that we have

$$h \cdot [n, n+1] \in [\![\psi_1 * a * b * b]\!]^1_{\eta,\rho} = [\![\psi_1]\!]^1_\eta * \rho(b) * \rho(b),$$

where the equality holds because $\rho(a) = \mathsf{Heap}$ is the unit for $*$. We then write $h \cdot [n, n+1] = g_1 \cdot g_2 \cdot g_3$ for $g_1 \in [\![\psi]\!]^1_\eta$ and $g_2, g_3 \in \rho(b)$. But as g_2 and g_3 have disjoint domains we must have $(h - h_2) \cdot [n] \sqsubseteq g_2$ and $[n+1] \sqsubseteq g_3$ or the version with g_2 and g_3 swapped. In any case we have that

$$\begin{aligned} \mathrm{dom}(g_1) &= \mathrm{dom}(h \cdot [n, n{+}1]) \setminus (\mathrm{dom}(g_2 \cdot g_3)) \\ &\subseteq \mathrm{dom}(h \cdot [n, n{+}1]) \setminus (\mathrm{dom}(h{-}h_2) \cup \{n, n{+}1\}) \\ &= \mathrm{dom}(h_2). \end{aligned}$$

But then we have $g_1 \sqsubseteq h_2$ and since $g_1 \in [\![\psi_1]\!]^1_\eta$ we get $h_2 \in [\![\psi_1]\!]^1_\eta$ too. If we have $h \cdot [n, n+1] \in [\![\psi_2 * b * b]\!]^1_{\eta,\rho}$ we proceed similarly.

The above short proof is the crux of the example. It implies unary η-validity – this we knew already – but also the binary η-validity. To see this, we pick an arbitrary environment $\rho : \{a, b\} \to \mathsf{IRel}_2$, we take arbitrary $(h_1, h_2) \in [\![\varphi_1 * a * b \wedge \varphi_2 * a * b * b]\!]^2_{\eta,\rho}$ and we aim to prove that $(h_1, h_2) \in [\![\psi_1 * a * b * b \vee \psi_2 * b * b]\!]^2_{\eta,\rho}$ too. We split (h_1, h_2) according to the conjuncts. Because of Lemma 3 and the upward closedness condition of IRel_2, we can write

$$(h_1, h_2) = (g^1, g^1) \cdot (g^2_1, g^2_2) \cdot (g^3_1, g^3_2)$$

for $g^1 \in [\![\varphi_1]\!]^1_\eta$, $(g^2_1, g^2_2) \in \rho(a)$ and $(g^3_1, g^3_2) \in \rho(b)$. Also we can write

$$(h_1, h_2) = (f^1, f^1) \cdot (f^2_1, f^2_2) \cdot (f^3_1, f^3_2) \cdot (f^4_1, f^4_2)$$

for $f^1 \in [\![\varphi_2]\!]^1_\eta$, $(f^2_1, f^2_2) \in \rho(a)$ and $(f^3_1, f^3_2), (f^4_1, f^4_2) \in \rho(b)$. But now $g^1 + f^1$ with subheaps g^1 and f^1 fulfills the above properties and so we get $f^1 \in [\![\psi_1]\!]^1_\eta$ or $f^1 \in [\![\psi_2]\!]^1_\eta$ and the second splitting of (h_1, h_2) shows that (h_1, h_2) lie in the binary interpretation of the first or second disjunct, respectively. Notice that neither $g^1 \in [\![\psi_1]\!]^1_\eta$ nor $g^1 \in [\![\psi_2]\!]^1_\eta$ would have worked since the first conjunct has too few variables, i.e., $\Pi(1) \not\geq \Omega(1)$ and $\Pi(1) \not\geq \Omega(2)$ ■

Generalizing the unary consequence that served as the crucial stepping stone in the above example we arrive at the following condition on our implications:

Definition 6 (Parametricity Condition) *Assume that we have an implication of the form* (1) *and an appropriate environment η. For all $h, h_1, \ldots, h_M \in \mathsf{Heap}$ with $h_i \sqsubseteq h$ and $h_i \in [\![\varphi_i]\!]^1_\eta$ for all $1 \leq i \leq M$ we must have one or two of these options:*

1. *There are $1 \leq i \leq M$ and $1 \leq j \leq N$ such that $h_i \in [\![\psi_j]\!]^1_\eta$, $\Pi(i) \geq \Omega(j)$ and the j-th disjunct is not empty.*
2. *There is $1 \leq j \leq N$ such that $h \in [\![\psi_j]\!]^1_\eta$ and the j-th disjunct is empty.*

Note that specializing the Parametricity Condition, henceforth just the PC, to an implication of the form treated in the above example yields the stated unary consequence because no disjuncts are empty. The second option in the PC will be motivated later.

We emphasize that the PC may hold or may fail for any given combination of an implication and environment η. But if it holds then we have binary η-validity; the proof in case of the first option of the PC is an easy generalization of the latter half of the above example:

Proposition 7 *The PC implies binary η-validity.*

We arrive now at the first lifting theorem. It is a generalization of the former half of Example 5; the proof of the theorem has a lot more details to it than the example but the overall idea is the same. The theorem states a criterion on the layout of the variables that, if met, means that unary η-validity implies the PC and hence also binary η-validity. The criterion is, loosely, that we can remove all variables that occur as labels of solid lines without introducing new solid lines and without emptying any disjuncts:

Theorem 8 (Shadow-Lift) *Unary η-validity of an implication implies the PC if each dashed line has a label that is not a label on a solid line and each disjunct has an occurrence of a variable that is not a label on a solid line. Spelling it out in symbols, we require, with $L = \{(i, j) \mid 1 \leq i \leq M \wedge 1 \leq j \leq N\}$, that*

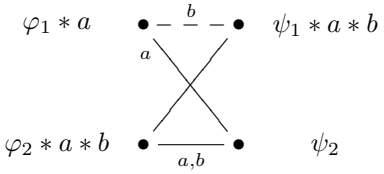
$$\begin{array}{l}\forall (i,j) \in L.\ \Pi(i) \not\geq \Omega(j) \implies \\ \quad \exists c \in V.\ \Pi(i)(c) < \Omega(j)(c) \wedge \\ \qquad \big(\forall (k,l) \in L.\ \Pi(k) \geq \Omega(l) \implies \Pi(k)(c) = \Omega(l)(c)\big)\end{array}$$

and

$$\begin{array}{l}\forall 1 \leq j \leq N.\ \exists c \in V.\ \Omega(j)(c) > 0 \wedge \\ \quad \big(\forall (k,l) \in L.\ \Pi(k) \geq \Omega(l) \implies \Pi(k)(c) = \Omega(l)(c)\big).\end{array}$$

As motivation for the next lifting theorem, we note that the variable layout criterion of the above theorem fails if one or more disjuncts are empty. Correspondingly, we never touch upon the second option of the PC. But there are variable layouts with empty disjuncts that ensure lifting:

Example 9 (Balloon-Lift) For any four assertions $\varphi_1, \varphi_2, \psi_1, \psi_2$ with no assertion variables and any appropriate environment η we have that unary η-validity of the following implication implies binary η-validity:



Assume unary η-validity. As in Example 5 we derive a unary consequence as intermediate result: For any $h \in \mathsf{Heap}$ with subheaps $h_1 \sqsubseteq h$ and $h_2 \sqsubseteq h$ such that $h_1 \in [\![\varphi_1]\!]^1_\eta$ and $h_2 \in [\![\varphi_2]\!]^1_\eta$ we have that either $h_2 \in [\![\psi_1]\!]^1_\eta$ or $h \in [\![\psi_2]\!]^1_\eta$.

To prove this, let h, h_1 and h_2 be as assumed. We construct $\rho : \{a, b\} \to \mathsf{IRel}_1$ by letting $\rho(a) = \mathsf{Heap}$ and

$\rho(b) = \{h - h_2\}^\uparrow$. We get that

$$h \sqsupseteq h_1 \in [\![\varphi_1]\!]^1_\eta = [\![\varphi_1]\!]^1_\eta * \mathsf{Heap} = [\![\varphi_1 * a]\!]^1_{\eta,\rho},$$

and

$$h = h_2 \cdot (h - h_2) \in [\![\varphi_2]\!]^1_\eta * \rho(b) = [\![\varphi_2]\!]^1_\eta * \mathsf{Heap} * \rho(b) = [\![\varphi_2 * a * b]\!]^1_{\eta,\rho}.$$

This means that h must lie in the interpretation of one of the disjuncts. If it is the first, we inspect the interpretation and get that

$$h = g_1 \cdot g_2 \cdot g_3$$

for $g_1 \in [\![\psi_1]\!]^1_\eta$, $g_2 \in \mathsf{Heap}$ and $g_3 \sqsupseteq h - h_2$. It means that

$$\mathrm{dom}(g_1) = \mathrm{dom}(h) \setminus \mathrm{dom}(g_2 \cdot g_3) \subseteq \mathrm{dom}(h) \setminus \mathrm{dom}(g_2) \subseteq \mathrm{dom}(h) \setminus \mathrm{dom}(h - h_2) = \mathrm{dom}(h_2)$$

which implies that $g_1 \sqsubseteq h_2$ and so $h_2 \in [\![\psi_1]\!]^1_\eta$. If, on the other hand, h lies in the interpretation of the second disjunct then we are done immediately.

Now we prove the claim of binary η-validity. We pick an arbitrary environment $\rho : \{a, b\} \to \mathsf{IRel}_2$, we take arbitrary $(h_1, h_2) \in [\![\varphi_1 * a \wedge \varphi_2 * a * b]\!]^2_{\eta,\rho}$ and we must prove that $(h_1, h_2) \in [\![\psi_1 * a * b \vee \psi_2]\!]^2_{\eta,\rho}$ too. We write

$$(h_1, h_2) = (g^1, g^1) \cdot (g^2_1, g^2_2)$$

for $g^1 \in [\![\varphi_1]\!]^1_\eta$ and $(g^2_1, g^2_2) \in \rho(a)$, and

$$(h_1, h_2) = (f^1, f^1) \cdot (f^2_1, f^2_2) \cdot (f^3_1, f^3_2)$$

for $f^1 \in [\![\varphi_2]\!]^1_\eta$, $(f^2_1, f^2_2) \in \rho(a)$ and $(f^3_1, f^3_2) \in \rho(b)$. But now $g^1 + f^1$ with subheaps g^1 and f^1 satisfies the above properties and so we get $f^1 \in [\![\psi_1]\!]^1_\eta$ or $g^1 + f^1 \in [\![\psi_2]\!]^1_\eta$. If $f^1 \in [\![\psi_1]\!]^1_\eta$ holds then the second splitting shows that (h_1, h_2) is in the interpretation of the first disjunct. If $g^1 + f^1 \in [\![\psi_2]\!]^1_\eta$, we are done too, since we may write $(h_1, h_2) = (g^1 + f^1, g^1 + f^1) \cdot (e_1, e_2)$ for some $(e_1, e_2) \in \mathsf{Heap}^2$ and so (h_1, h_2) lies in the interpretation $[\![\psi_2]\!]^2_\eta = \Delta([\![\psi_2]\!]^1_\eta)$ of the second conjunct. ■

Just as we did for Example 5 we may generalize the former half of this example, this yields Theorem 10 below. The latter half of the example, on the other hand, constitutes an example of the approach of the proof of Proposition 7 in case we run into the second option of the PC.

Theorem 10 (Balloon-Lift) *Unary η-validity of an implication implies the PC if there is a subset $B \subseteq V$ with the following three properties. First, each conjunct has at most one occurrence of a variable from B, i.e.,*

$$\forall 1 \le i \le M.\ \sum_{c \in B} \Pi(i)(c) \le 1.$$

Second, each disjunct is empty or has exactly one occurrence of a variable from B, i.e.,

$$\forall 1 \le j \le N.\ \sum_{c \in V} \Omega(j)(c) = 0 \ \vee \ \sum_{c \in B} \Omega(j)(c) = 1.$$

Third, each dashed line must have a label from B. That is, when $L = \{(i, j) \mid 1 \le i \le M \wedge 1 \le j \le N\}$,

$$\forall (i, j) \in L.\ \Pi(i) \not\ge \Omega(j) \implies \exists c \in B.\ \Pi(i)(c) < \Omega(j)(c).$$

One thing to note about the theorem is that if we have no empty disjuncts, none of the variables in the subset $B \subseteq V$ can be labels of a solid line. In particular, the conditions of Theorem 8 are met, so the above theorem is really only useful if one or more disjuncts are empty.

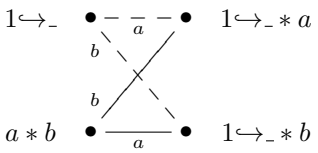
The final lifting theorem captures the oddities of the special case of just one conjunct:

Theorem 11 (Lonely-Lift) *Unary η-validity of an implication implies the PC if there is just one conjunct, i.e., M=1, and all lines are solid, i.e., $\Pi(1) \ge \Omega(j)$ for all $1 \le j \le N$.*

4.3 Completeness

It is now time for examples of implications that do not lift, i.e., that are valid in the unary interpretation but not in the binary. The first is based on the following observation: If $h \in [\![1 \hookrightarrow \_]\!]^1_\eta$ and $h \in p * q$ for $h \in \mathsf{Heap}$ and $p, q \in \mathsf{IRel}_1$ then we have $h \in [\![1 \hookrightarrow \_]\!]^1_\eta * p$ or $h \in [\![1 \hookrightarrow \_]\!]^1_\eta * q$. This is because we must have $[1 \mapsto n] \sqsubseteq h$ for some $n \in \mathsf{Int}$ and so writing $h = h_1 \cdot h_2$ with $h_1 \in p$ and $h_2 \in q$ gives us $[1 \mapsto n] \sqsubseteq h_1$ or $[1 \mapsto n] \sqsubseteq h_2$. But this line of argument breaks down if we change to binary reading. We have, e.g., $([1], [1]) \in [\![1 \hookrightarrow \_]\!]^2_\eta$ and $([1], [1]) \in \{([1], [])\}^\uparrow * \{([], [1])\}^\uparrow$ but both $[\![1 \hookrightarrow \_]\!]^2_\eta * \{([1], [])\}^\uparrow$ and $[\![1 \hookrightarrow \_]\!]^2_\eta * \{([], [1])\}^\uparrow$ are empty. We can recast this as an implication that cannot be lifted:

Example 12 (Fan-Counter) This implication is valid on the unary but not on the binary level:



First we argue that the implication holds on the unary level. Let $\rho : \{a, b\} \to \mathsf{IRel}_1$ be an arbitrary environment of upwards closed sets of heaps to a and b. Let $h \in \mathsf{Heap}$ be arbitrary and assume that

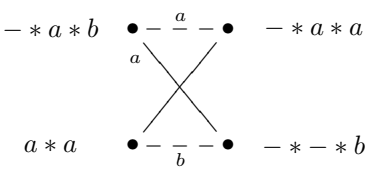
$$h \in [\![1 \hookrightarrow \_ \wedge (a * b)]\!]^1_{\eta,\rho} = [\![1 \hookrightarrow \_]\!]^1_\rho \cap (\rho(a) * \rho(b)).$$

By the above observation we get either $h \in [\![1{\hookrightarrow}\_]\!]^1_\eta *$ $\rho(a)$ or $h \in [\![1{\hookrightarrow}\_]\!]^1_\eta * \rho(b)$ which matches the right hand side of the implication.

Now we move on to prove that the implication fails on the binary level. Define an environment $\rho : \{a, b\} \to \mathsf{IRel}_2$ by $\rho(a) = \{([1], [])\}^\uparrow$ and $\rho(b) = \{([], [1])\}^\uparrow$. Then, $[\![1{\hookrightarrow}\_ \wedge a * b]\!]^2_{\eta,\rho} = [\![1{\hookrightarrow}\_]\!]^2_{\eta,\rho} \cap (\rho(a) * \rho(b))$, which contains the pair $([1], [1])$. But, as observed, both disjuncts have empty binary interpretations. ■

An observation of similar nature is that for $p \in \mathsf{IRel}_1$ we have either $p = \mathsf{Heap}$ or $p \subseteq [\![-]\!]^1_\eta = \{[m{\mapsto}n] \mid m \in \mathsf{PosInt}, n \in \mathsf{Int}\}^\uparrow$ because if $h \neq \mathsf{Heap}$ then it cannot contain the empty heap. On the binary level, however, we have $\mathsf{Heap}^2 \neq \{([1], [])\}^\uparrow \not\subseteq [\![-]\!]^2_\eta = \{([m{\mapsto}n], [m{\mapsto}n]) \mid m \in \mathsf{PosInt}, n \in \mathsf{Int}\}^\uparrow$. One consequence is this:

Example 13 (Bridge-Counter) This implication is valid on the unary but not on the binary level:



First we argue that the implication holds on the unary level. Let $\rho : \{a, b\} \to \mathsf{IRel}_1$ be an arbitrary environment that assigns upwards closed sets of heaps to each of the two variables. We branch on the value of $\rho(a)$. If $\rho(a) \neq \mathsf{Heap}$ then we have $\rho(a) \subseteq [\![-]\!]^1_\eta$ which again means that the first conjunct directly implies the second disjunct. If $\rho(a) = \mathsf{Heap}$ holds, we get that

$$[\![-*a*b]\!]^1_{\eta,\rho} = [\![-]\!]^1_{\eta,\rho} * \mathsf{Heap} * \rho(b) = [\![-]\!]^1_{\eta,\rho} * \rho(b)$$
$$\subseteq [\![-]\!]^1_{\eta,\rho} = [\![-]\!]^1_{\eta,\rho} * \mathsf{Heap} * \mathsf{Heap} = [\![-*a*a]\!]^1_{\eta,\rho}$$

because Heap is the unit for $*$. Hence we get that the first conjunct implies the first disjunct and we have proved that the implication holds unarily.

Now we prove that the implication fails on the binary level. Define an environment $\rho : \{a, b\} \to \mathsf{IRel}_2$ by $\rho(a) = \{([1], [])\}^\uparrow \cup \{([2], [2])\}^\uparrow$ and $\rho(b) = \mathsf{Heap}^2$. Observe now that $([1, 2], [2]) = ([2], [2]) \cdot ([1], []) \cdot ([], [])$, which implies that $([1, 2], [2]) \in [\![-*a*b]\!]^2_{\eta,\rho}$. From the rewriting $([1, 2], [2]) = ([1], []) \cdot ([2], [2])$, we get $([1, 2], [2]) \in [\![a*a]\!]^2_{\eta,\rho}$ too and so this pair of heaps lies in the interpretation of the left hand side. But it does not belong to the interpretation of either disjunct. An easy – if somewhat indirect – way of realizing this is to note that any pair of heaps in either $[\![-]\!]^2_{\eta,\rho}$ or in $[\![a*a]\!]^2_{\eta,\rho}$ must have a second component with nonempty domain. But then any pair of heaps in the interpretation of either disjunct must have a second component with a domain of at least two elements. In particular, neither can contain the pair $([1, 2], [2])$. ■

In principle, the above two observations are all that we need to prove completeness. Or, phrased differently, assume that we have a layout of variables that fail the criteria of all three lifting theorems; by applying one of the two observations, we can then build a concrete implication with that variable layout and with unary but not binary validity.

Having said that, the territory to cover is huge; the full completeness proof is a lengthy and rather technical journey, the details of which do not provide much insight. We supply it as a series of lemmas in Appendix D; these include generalizations of Example 12 and Example 13 above. If one verifies the lemmas in the order listed and apply them as sketched then it is feasible, if not exactly easy, to prove the following:

Theorem 14 (Completeness) *If a variable layout meets none of the criteria in Theorems 8, 10 and 11, then there are choices of φ_i's and ψ_j's with no variables such we have unary but not binary validity.*

5 Higher Arities and Parametricity

We saw in Proposition 7 that the PC implies binary η-validity of an implication. It is easy to show that the PC also implies unary η-validity, either directly or by observing that binary implies unary. A natural question to ask is whether we can reverse this? Example 12 shows that unary validity does not entail the PC, because the latter fails for that concrete implication. But as binary validity fails too, we could hope that binary validity would enforce the PC. Unfortunately, this is not the case, as demonstrated by the implication

$$1{\hookrightarrow}\_ \wedge a * a * b \implies 1{\hookrightarrow}\_ * a \vee 1{\hookrightarrow}\_ * b.$$

Here the PC is the same as for Example 12 and hence still is not true, but we do have binary validity. We do not, however, have ternary validity but the example could easily be scaled: having n occurrences of a in the second conjunct means n-ary but not $n+1$-ary validity for any $n \geq 1$. In summary, we have seen that for any $n \geq 1$ we can have n-ary validity whilst the PC fails.

What does hold, however, is the following:

Theorem 15 *For an implication of the form* (1) *and an appropriate environment η we have that n-ary η-validity implies the PC if $n \geq \max\{M_1, \ldots, M_M\}$.*

Notice how this fits nicely with the above example: with n occurrences of a we have n-ary validity but we

need $(n{+}1)$-ary validity to prove the PC since there is also a single b. The proof is in Appendix E, and reuses techniques from the proofs of Theorems 8 and 10.

By an easy generalization of Proposition 7 we have the following corollary to the above theorem:

Corollary 16 *The PC holds iff we have n-ary η-validity for all $n \geq 1$.*

This corollary can be read, loosely, as a coincidence between parametric polymorphism as introduced by Strachey [19] and relational parametricity as proposed by Reynolds [18]: The PC corresponds to Strachey parametricity in the loose sense that if it holds, then there is an approach, parametric in the assertion variables, that produce right hand side proofs of heap membership from the left hand side ones: Take a heap, split it along the conjuncts, apply the PC to the parts in the interpretations of the φ's and you are done, possibly after discarding some variables. This involves no branching or other intrinsic operations on the assertion variables, which we are free to discard by our intuitionistic setup. If, on the other hand, the implication is η-valid for arbitrary arity, then it is fair to call it relationally parametric. Note also that the Examples 12 and 13 branch on assertion variable values.

This result is analogous to the conjecture of coincidence between Strachey parametricity and n-ary relational parametricity for traditional type-based parametricity [17, Page 2].

Finally we note that as a consequence of the above corollary we have that the lifting theorems in the previous really show that unary validity can be lifted to validity of arbitrary arity. In some sense, they are stronger than required for representation independence, for which binary validity suffices. The authors are unaware of any practical applications of this fact.

6 Representation Independence

In this section, we relate our lifting theorems to representation independence. We consider separation logic with assertion variables where the rule of consequence is restricted according to our lifting theorems, and we define a relational semantics of the logic, which gives a representation independence theorem: all proved clients cannot distinguish between appropriately related module implementations. For space reasons, our presentation will be minimalistic, covering only a part of the logic; the missing parts, in particular allocation of new cells, we believe, can be handled using FM sets, following [6].

We consider commands C given by the grammar:

$$C ::= k \mid [E]{:=}E \mid \mathbf{let}\, y{=}[E]\, \mathbf{in}\, C \mid C; C \mid \mathbf{if}\, B\, C\, C$$

$$\frac{\textsc{Chk}(\varphi', \varphi) \quad \varphi' \models^1 \varphi \quad \{\varphi\}C\{\psi\} \quad \psi \models^1 \psi' \quad \textsc{Chk}(\psi, \psi')}{\{\varphi'\}C\{\psi'\}}$$

$$\frac{\{\varphi\}C\{\varphi'\}}{\{\varphi * \psi\}C\{\varphi' * \psi\}} \qquad \frac{\{\varphi\}C\{\psi\}}{\{\exists x.\, \varphi\}C\{\exists x.\, \psi\}}\ x \notin \mathsf{FV}(C)$$

$$\frac{}{\Gamma, \{\varphi\}k\{\psi\} \vdash \{\varphi\}k\{\psi\}} \qquad \frac{}{\{E{\hookrightarrow}\_\}[E]{:=}F\{E{\hookrightarrow}F\}}$$

$$\frac{\{\varphi * E{\hookrightarrow}x\}\mathbf{let}\, x{=}[E]\, \mathbf{in}\, C\{\psi\}}{\{\exists x.\varphi * E{\hookrightarrow}x\}\mathbf{let}\, x{=}[E]\, \mathbf{in}\, C\{\psi\}}\ x \notin \mathsf{FV}(\psi)$$

$$\frac{\{\varphi\}C\{\varphi'\} \quad \{\varphi'\}C'\{\psi\}}{\{\varphi\}C; C'\{\psi\}} \qquad \frac{\{\varphi \wedge B\}C\{\psi\} \quad \{\varphi \wedge \neg B\}C'\{\psi\}}{\{\varphi\}\mathbf{if}\, B\, C\, C'\{\psi\}}$$

Figure 4. Proof Rules

Here B is a heap-independent boolean expression, such as $x{=}0$. Commands C are from the loop-free simple imperative language. They can call module operations k, and manipulate heap cells; command $[x]{:=}E$ assigns E to the heap cell x, and this assigned value is read by $\mathbf{let}\, y{=}[x]\, \mathbf{in}\, C$, which also binds y to the read value and runs C under this binding.

Properties of commands C are specified using Hoare triples $\Gamma \vdash \{\varphi\}C\{\psi\}$, where the context Γ is a set of triples for module operations. Figure 4 shows rules for proving these properties. In the figure, we omit contexts, if the same context Γ is used for all the triples.

The rule of consequence deserves attention. Note that the rule uses semantic implications $\models^1$ in the standard unary interpretation, thus allowing the use of existing theorem provers for (higher-order) separation logic. The rule does not allow all semantic implications, but only those that pass our algorithm Chk, so as to ensure that the implications can lift to the relational level. Our algorithm $\textsc{Chk}(\varphi, \psi)$ performs two checks, and returns true only when both succeed. The first check is whether φ and ψ can be transformed to simple assertions φ' and ψ', using only the distribution of $*$ over $\exists x$ and $\vee$ and distributive lattice laws for $\vee$ and $\wedge$. If this check succeeds and gives φ' and ψ', the algorithm transforms $\varphi' \models^1 \psi'$ to a set of implications of the form (1) in Section 4 (Lemma 4). Then, it uses the method outlined just before Section 4.1, and does the second check on whether all the implications in the resulting set lift to the relational level.

Commands C are interpreted in a standard way, as functions of the type: $[\![C]\!]_{\eta,u} \in \mathsf{Heap} \to (\mathsf{Heap} \cup \{err\})$. Here err denotes a memory error, and η and u are environments that provide the meanings of, respectively, free ordinary variables and module operations. For instance, $[\![k]\!]_{\eta,u}$ is $u(k)$.

Our semantics of triples, on the other hand, is not standard, and uses the binary interpretation of asser-

tions: $(\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}C\{\psi\}$ iff

$$\forall r \in \mathsf{IRel}_2.\ \forall f, g \in \mathsf{Heap}.\ (f,g) \in [\![\varphi]\!]^2_{\eta,\rho} * r \implies ([\![C]\!]_{\eta,u_1}(f), [\![C]\!]_{\eta,u_2}(g)) \in [\![\psi]\!]^2_{\eta,\rho} * r.$$

The environment ρ provides the meanings of assertion variables, and the 2-dimensional vector $\mathbf{u}$ gives the two meanings for module operations; intuitively, each u_i corresponds to the i-th module implementation. The interpretation means that if two module implementations $\mathbf{u}$ are used by the same client C, then these combinations should result in the same computation, in the sense that they map φ-related input heaps to ψ-related outputs. The satisfaction of triples can be extended to $(\eta, \rho, \mathbf{u}) \models^2 \Gamma$, by asking that all triples in Γ should hold wrt. $(\eta, \rho, \mathbf{u})$. Using these satisfaction relations on triples and contexts, we define the notion of 2-validity of judgements: $\Gamma \vdash \{\varphi\}C\{\psi\}$ is 2-valid iff

$$\forall (\eta, \rho, \mathbf{u}).\ (\eta, \rho, \mathbf{u}) \models^2 \Gamma \implies (\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}C\{\psi\}.$$

Theorem 17 *Every derivable $\Gamma \vdash \{\varphi\}C\{\psi\}$ is 2-valid.*

It is this theorem that we use to derive the representation independence results mentioned in the introduction. Consider again the example in Figure 1. Since the proof of the left hand side client C is derivable using the above rules, with Γ being the interface specification for the operations of the counter, we get 2-validity of $\Gamma \vdash \{1 \hookrightarrow \_\}C\{1 \hookrightarrow \_\}$. Therefore, when we run the client C with the related module implementations in the introduction, we find that C maps $[\![1 \hookrightarrow \_]\!]^2$-related heaps (i.e, heaps with the same value at cell 1) to $[\![1 \hookrightarrow \_]\!]^2$-related heaps again.

7 Conclusion and Discussion

In this paper, we have given a sound and complete characterization of when *semantic* implications between assertions in higher-order separation logic can be lifted to a relational interpretation. This characterization has, then, been used to identify proofs of clients that respect the abstraction of module internals, specified by means of assertion variables, and to show representation independence for clients with such proofs. We hope that our results provide a solid semantic basis for recent logic-based approaches to data abstraction.

In earlier work, Banerjee and Naumann [2] studied relational parametricity for dynamically allocated heap objects in a Java-like language. Banerjee and Naumann made use of a non-trivial semantic notion of confinement to describe internal resources of a module; here instead we use higher-order separation logic to describe which resources are internal to the module.

Relational interpretations have also been used to give models of programming languages with local state, which can validate representation independence results [13, 16, 3, 1]. These results typically rely on the module allocating the private state, whereas we use the power of separation logic and allow the ownership transfer of states from client to module. For instance, in the two-stage counter in the introduction, the ownership of the cell 1 is transferred from the client to the module upon calling `init`. Even with this ownership transfer, representation independence is guaranteed, because we consider only those clients having (good) proofs in separation logic. This contrasts with representation independence results in local state models, which consider not some but all well-typed clients.

References

[1] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *POPL*, pages 340–353, 2009.

[2] A. Banerjee and D. Naumann. Ownership confinement ensures representation independence for object-oriented programs. *JACM*, 52(6), 2005.

[3] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *TLCA*, pages 86–101, 2005.

[4] B. Biering, L. Birkedal, and N. Torp-Smith. BI hyperdoctrines and higher-order separation logic. In *ESOP*, 2005.

[5] L. Birkedal, N. Torp-Smith, and H. Yang. Semantics of separation-logic typing and higher-order frame rules. In *LICS*, pages 260–269, 2005.

[6] L. Birkedal and H. Yang. Relational parametricity and separation logic. In *FOSSACS*, pages 93–107, 2007.

[7] W.-N. Chin, C. David, H. H. Nguyen, and S. Qin. Enhancing modular oo verification with separation logic. In *POPL*, 2008.

[8] D. Distefano and M. Parkinson. jStar: towards practical verification for java. In *OOPSLA*, pages 213–226, 2008.

[9] R. Dockins, A. Hobor, and A. Appel. A fresh look at separation algebras and share accounting. In *APLAS*, pages 161–177, 2009.

[10] B. Jacobs and F. Piessens. The VeriFast program verifier. Technical Report CW-520, Katholieke Universiteit Leuven, 2008.

[11] A. Nanevski, A. Ahmed, G. Morrisett, and L. Birkedal. Abstract predicates and mutable ADTs in Hoare type theory. In *ESOP*, 2007.

[12] A. Nanevski, G. Morrisett, A. Shinnar, P. Govereau, and L. Birkedal. Ynot: dependent types for imperative programs. In *ICFP*, pages 229–240, 2008.

[13] P. O'Hearn and R. Tennent. Parametricity and local variables. *JACM*, 42(3):658–709, May 1995.

[14] M. Parkinson and G. Bierman. Separation logic and abstraction. In *POPL*, pages 247–258, 2005.

[15] M. Parkinson and G. Bierman. Separation logic, abstraction and inheritance. In *POPL*, pages 75–86, 2008.

[16] A. Pitts and I. Stark. Observable properties of higher order functions that dynamically create local names, or: What's new? In *MFCS*, pages 122–141, 1993.

[17] G. Plotkin and M. Abadi. A logic for parametric polymorphism. In *TLCA*, pages 361–375, 1993.

[18] J. C. Reynolds. Types, abstraction, and parametric polymorphism. In *Information Processing 83*, pages 513–523, 1983.

[19] C. Strachey. Fundamental concepts in programming languages. In *Proceedings of the 1967 International Summer School in Computer Programming, Copenhagen, Denmark.*, 1967.

A Proofs of Lemma 2 and Lemma 3

Lemma 2 *Function Δ_n preserves the complete lattice structure and the $*$ operator.*

Proof: From the definition, it is immediate that $\Delta_n(\mathsf{Heap}) = \mathsf{Heap}^n$ and $\Delta_n(\emptyset) = \emptyset$. Now consider a non-empty family $\{p_i\}_{i\in I}$ of predicates in IRel_1. In order to show the preservation of the complete lattice structure, we need to prove that

$$\Delta_n(\bigcap_{i\in I} p_i) = \bigcap_{i\in I} \Delta_n(p_i) \ \wedge \ \bigcup_{i\in I} \Delta_n(p_i) = \Delta_n(\bigcup_{i\in I} p_i).$$

The $\subseteq$ direction in both cases is easy; it follows from the monotonicity of Δ_n.

We start with the $\supseteq$ direction for the meet operator. Pick $(h_1, \ldots, h_n)$ from $\bigcap_{i\in I} \Delta_n(p_i)$. Then,

$$\forall i \in I.\ (h_1, \ldots, h_n) \in \Delta_n(p_i).$$

By the definition of Δ_n, this means that

$$\forall i \in I.\ \exists f_i \in p_i.\ f_i \sqsubseteq h_1 \ \wedge \ \ldots \ \wedge \ f_i \sqsubseteq h_n. \qquad (2)$$

Let $f = \sum_{i\in I} f_i$. The sum here is well-defined, because (a) there are only finitely many f's such that $f \sqsubseteq h_k$ for all $1 \le k \le n$, and (b) any two such f and g should have the same value for every location in $\mathrm{dom}(f) \cap \mathrm{dom}(g)$. Since all f_i's satisfy (2), their sum f also satisfies

$$f \sqsubseteq h_1 \ \wedge \ \ldots \ \wedge \ f \sqsubseteq h_n.$$

Furthermore, $f \in \bigcap_{i\in I} p_i$, because p_i's are upward closed and f is an extension of f_i in p_i. Hence, $\Delta_n(\bigcap_{i\in I} p_i) \subseteq \bigcap_{i\in I} \Delta_n(p_i)$.

Next we prove the $\supseteq$ direction for the join operator. Pick $(h_1, \ldots, h_n)$ from $\Delta_n(\bigcup_{i\in I} p_i)$. Then,

$$\exists i \in I.\ \exists f \in p_i.\ f \sqsubseteq h_1 \ \wedge \ \ldots \ \wedge \ f \sqsubseteq h_n.$$

Hence, by the definition of Δ_n,

$$(h_1, \ldots, h_n) \ \in \ \Delta_n(p_i) \ \subseteq \ \bigcup_{i\in I} \Delta_n(p_i),$$

as desired.

Finally, it remains to show that Δ_n preserves the $*$ operator. Consider predicates $p, q \in \mathsf{IRel}_1$. We need to prove that

$$\Delta_n(p * q) \ = \ \Delta_n(p) * \Delta_n(q).$$

Choose an arbitrary $(h_1, \ldots, h_1)$ from $\Delta_n(p * q)$. By the definition of $\Delta_n(p * q)$, it follows that

$$\exists f \in p.\ \exists g \in q.\ (\mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset) \ \wedge \\ f \sqsubseteq h_1 \ \wedge \ \ldots \ \wedge \ f \sqsubseteq h_n \ \wedge \ g \sqsubseteq h_1 \ \wedge \ \ldots \ \wedge \ g \sqsubseteq h_n.$$

Now, define $f_i = f$ and $g_i = h_i - f$ for $i \in \{1, \ldots, n\}$. Then,

$$(\forall i \in \{1, \ldots, n\}.\ f_i \cdot g_i = h_i) \\ \wedge \ (f_1, \ldots, f_n) \in \Delta_n(p) \ \wedge \ (g_1, \ldots, g_n) \in \Delta_n(q).$$

Hence, $(h_1, \ldots, h_n) \in \Delta_n(p) * \Delta_n(q)$. This shows that $\Delta_n(p * q) \subseteq \Delta_n(p) * \Delta_n(q)$. For the other inclusion, suppose that

$$(h_1, \ldots, h_1) \ \in \ \Delta_n(p) * \Delta_n(q).$$

Then, by the definition of $*$,

$$\exists (f_1, \ldots, f_n) \in \Delta_n(p).\ \exists (g_1, \ldots, g_n) \in \Delta_n(q). \\ (\forall i \in \{1, \ldots, n\}.\ f_i \cdot g_i = h_i).$$

Since $(f_1, \ldots, f_n) \in \Delta_n(p)$ and $(g_1, \ldots, g_n) \in \Delta_n(q)$, there are $f \in p$ and $g \in q$ such that

$$f \sqsubseteq f_1 \ \wedge \ \ldots \ \wedge \ f \sqsubseteq f_n \ \wedge \ g \sqsubseteq g_1 \ \wedge \ \ldots \ \wedge \ g \sqsubseteq g_n.$$

Furthermore, since f_1 and g_1 have disjoint domains, their subheaps f and g must have disjoint domains as well. Consequently, $f \cdot g$ is well defined, and it satisfies

$$f \cdot g \in p * q \ \wedge \ (\forall i \in \{1, \ldots, n\}.\ f \cdot g \ \sqsubseteq \ f_i \cdot g_i \ \sqsubseteq \ h_i).$$

This implies that $(h_1, \ldots, h_n) \in \Delta_n(p * q)$, as desired. ■

Lemma 3 *For all φ and η, ρ, ρ', if $\Delta_n(\rho(a)) = \rho'(a)$ for every $a \in \mathsf{AVar}$, we have that $\Delta_n([\![\varphi]\!]^1_{\eta,\rho}) = [\![\varphi]\!]^n_{\eta,\rho'}$.*

Proof: We prove by induction on the structure of φ. All the inductive cases and the cases of true and false follow from the preservation result of Lemma 2. Thus, it is sufficient to show the lemma when $\varphi \equiv a$ or $\varphi \equiv P$. When $\varphi \equiv a$, the assumption of the lemma implies that

$$\Delta_n([\![a]\!]^1_{\eta,\rho}) \ = \ \Delta_n(\rho(a)) \ = \ \rho'(a) \ = \ [\![a]\!]^n_{\eta,\rho'}.$$

When $\varphi \equiv P$, we note that $\Delta_n \circ \Delta_1 = \Delta_n$, and conclude that

$$\Delta_n([\![P]\!]^1_{\eta,\rho}) \ = \ \Delta_n(\Delta_1((\!|P|\!)_\eta)) \ = \ \Delta_n((\!|P|\!)_\eta) \ = \ [\![P]\!]^n_{\eta,\rho'}.$$

■

B Proof of Lemma 4

Lemma 4 *There is an algorithm taking simple assertions φ, ψ and returning finitely many implications $\{\varphi^l \Longrightarrow \psi^l\}_{l\in L}$, such that (a) $\varphi^l \Longrightarrow \psi^l$ has the form (1) and (b) for any $n \in \{1, 2\}$, we have that $\varphi \models^n \psi$ holds iff $\varphi^l \models^n \psi^l$ holds for all $l \in L$.*

Proof: The algorithm first transforms ψ in the conjunctive normal form, using proof rules in classical logic, which hold in all the n-ary semantics. This gives an implication of the form:

$$\bigvee_{i=1}^{I}\bigwedge_{j=1}^{J}\varphi_{(i,j)} * \mathbf{a}_{(i,j)} \implies \bigwedge_{k=1}^{K}\bigvee_{l=1}^{L}\psi_{(k,l)} * \mathbf{b}_{(k,l)}.$$

Then, the algorithm constructs the below set:

$$\left\{\bigwedge_{j=1}^{J}\varphi_{(i,j)} * \mathbf{a}_{(i,j)} \implies \bigvee_{l=1}^{L}\psi_{(k,l)} * \mathbf{b}_{(k,l)}\right\}_{1\leq i\leq I, 1\leq k\leq K}.$$

Finally, it removes, in each implication, all the disjuncts that include assertion variables not appearing on the LHS of the implication. The outcome of this removal becomes the result of the algorithm. ■

C Layouts that Lift

Lemma 18 (Segregation) *For any $I, J \geq 1$ there are non-empty, finite* segregating *subsets $\mathbf{S}^{I,J}_{i,j} \subseteq \mathsf{PosInt}$ for all $1 \leq i \leq I$ and $1 \leq j \leq J$ with these properties:*

1. $\forall 1 \leq i_1, i_2 \leq I.\ \bigcup_{1\leq j\leq J}\mathbf{S}^{I,J}_{i_1,j} = \bigcup_{1\leq j\leq J}\mathbf{S}^{I,J}_{i_2,j}$.
2. $\forall 1 \leq i \leq I.\ \forall 1 \leq j_1 \neq j_2 \leq J.\ \mathbf{S}^{I,J}_{i,j_1} \cap \mathbf{S}^{I,J}_{i,j_2} = \emptyset$.
3. $\forall 1 \leq i_1 \neq i_2 \leq I.\ \forall 1 \leq j_1, j_2 \leq J.\ \mathbf{S}^{I,J}_{i_1,j_1} \cap \mathbf{S}^{I,J}_{i_2,j_2} \neq \emptyset$.

By 1 we define $\mathbf{S}^{I,J} = \bigcup_{1\leq j\leq J}\mathbf{S}^{I,J}_{i,j}$ for any $1 \leq i \leq I$.

Theorem 8 (Shadow-Lift) *Unary η-validity of an implication implies the PC if each dashed line has a label that is not a label on a solid line and each disjunct has an occurrence of a variable that is not a label on a solid line. Spelling it out in symbols, we require, with $L = \{(i,j) \mid 1 \leq i \leq M \wedge 1 \leq j \leq N\}$, that*

$$\begin{array}{l}\forall (i,j) \in L.\ \Pi(i) \not\geq \Omega(j) \implies \\ \quad \exists c \in V.\ \Pi(i)(c) < \Omega(j)(c) \wedge \\ \quad\quad \big(\forall (k,l) \in L.\ \Pi(k) \geq \Omega(l) \implies \Pi(k)(c) = \Omega(l)(c)\big)\end{array}$$

and

$$\begin{array}{l}\forall 1 \leq j \leq N.\ \exists c \in V.\ \Omega(j)(c) > 0 \wedge \\ \quad \big(\forall (k,l) \in L.\ \Pi(k) \geq \Omega(l) \implies \Pi(k)(c) = \Omega(l)(c)\big).\end{array}$$

Proof: Assume that we have an implication of the form (1) in Section 4 and an appropriate environment η, that the stated criterion on the variable layout holds and that we have unary η-validity. We must show that the PC holds.

According to Definition 6 we assume that we have heaps $h, h_1, \ldots, h_M \in \mathsf{Heap}$ with $h_i \sqsubseteq h$ and $h_i \in [\![\varphi_i]\!]^1_\eta$ for all $1 \leq i \leq M$. The core of the proof is the construction of a particular environment $\rho : V \to \mathsf{IRel}_1$. For that purpose we need some notation. For a subset $M \subseteq \mathsf{PosInt}$ we denote by $[M]$ the heap that has domain M and stores some fixed value, say 0, at all these locations. Let $C \subseteq V$ be the set of assertion variables that do not occur as labels on solid edges, i.e., for a $c \in V$ we have that $c \in C$ iff

$$\begin{array}{l}\forall 1 \leq i \leq M.\ \forall 1 \leq j \leq N. \\ \quad \Pi(i) \geq \Omega(j) \implies \Pi(i)(c) = \Omega(j)(c).\end{array}$$

For each $1 \leq i \leq M$ we let K_i be the set of second indices of all variables in conjunct i that lie in C, i.e., we set $K_i = \{1 \leq k \leq M_i \mid a_{i,k} \in C\}$. If non-empty, we let $k_i = \min(K_i)$.

We now define $\rho(c) = \mathsf{Heap}$ for $c \in V \setminus C$. For a variable $c \in C$ we let $\rho(c)$ be the union of

$$\bigcup_{\substack{1\leq i\leq M,\\ K_i\neq\emptyset,\\ a_{i,k_i}\equiv c}} (h - h_i)\cdot[\mathbf{S}^{M,K}_{i,k_i} + L]\cdot \prod_{\substack{1\leq k\leq K,\\ k\notin K_i}} [\mathbf{S}^{M,K}_{i,k} + L]$$

and

$$\bigcup_{1\leq i\leq M, K_i\neq\emptyset, k\in K_i\setminus\{k_i\}, a_{i,k}\equiv c} [\mathbf{S}^{M,K}_{i,k} + L],$$

where we have used $K = \max\{M_1, \ldots, M_M\}$ and $L = \max(\mathrm{dom}(h) \cup \{0\})$. For each $1 \leq i \leq M$ we can write $h \cdot [\mathbf{S}^{M,K} + L]$ as the following product

$$h_i \cdot (h - h_i) \cdot \prod_{k\in K_i} [\mathbf{S}^{M,K}_{i,k} + L] \cdot \prod_{1\leq k\leq K, k\notin K_i} [\mathbf{S}^{M,K}_{i,k} + L],$$

which implies that we have $h \cdot [\mathbf{S}^{M,K} + L]$ a member of $[\![\varphi_i * a_{i,1} * \cdots * a_{i,M_i}]\!]^1_{\eta,\rho}$. In summary, we have shown that $h \cdot [\mathbf{S}^{M,K} + L]$ lies in the unary interpretation of the left hand side in the environments η and ρ. By assumption, the same must hold for the right hand side and from this we aim to derive the PC.

We now know that $h \cdot [\mathbf{S}^{M,K} + L]$ lies in the interpretation of some disjunct, say disjunct j. This means that

$$\begin{array}{rl} h \cdot [\mathbf{S}^{M,K} + L] &\in [\![\psi_j * b_{j,1} * \cdots * b_{j,N_j}]\!]^1_{\eta,\rho} \\ &= [\![\psi_j]\!]^1_\eta * \prod_{k\in J}\rho(b_{j,k}),\end{array}$$

where $J = \{1 \leq k \leq N_j \mid b_{j,k} \in C\}$ is the set of second indices of variables of disjunct j that are in C. By the

second assumption of the theorem we know that $J \neq \emptyset$. We write

$$h \cdot [\mathbf{S}^{M,K} + L] = g \cdot \prod_{k \in J} g_k$$

for $g \in [\![\psi_j]\!]^1_\eta$ and $g_k \in \rho(b_{j,k})$ for each $k \in J$. By the properties of segregating sets we get that there must be a common $1 \leq i \leq M$ such that for all $k \in J$ there is $l_k \in K_i$ with

$$[\mathbf{S}^{M,K}_{i,l_k} + L] \sqsubseteq g_k,$$

i.e., the g_k's are all 'from the same conjunct'. But this implies $\Pi(i)(c) \geq \Omega(j)(c)$ for all $c \in C$ as the segregating sets are non-empty. But then $\Pi(i)(c) \geq \Omega(j)(c)$ must hold for $c \in V \setminus C$ too by the first assumption of the lemma and so $\Pi(i) \geq \Omega(j)$. Also we must have $\Pi(i)(c) = \Omega(j)(c)$ for each $c \in C$ by definition of C. By construction we have

$$\mathrm{dom}\left(\prod_{k \in J} g_k\right) \supseteq \mathrm{dom}(h - h_i) \cup (\mathbf{S}^{M,K} + L)$$

But then $\mathrm{dom}(g) \subseteq h_i$ and so we have $h_i \in [\![\psi_j]\!]^1_\eta$ too and we have proved the first option of the PC. ■

Theorem 10 (Balloon-Lift) *Unary η-validity of an implication implies the PC if there is a subset $B \subseteq V$ with the following three properties. First, each conjunct has at most one occurrence of a variable from B, i.e.,*

$$\forall 1 \leq i \leq M.\ \sum_{c \in B} \Pi(i)(c) \leq 1.$$

Second, each disjunct is empty or has exactly one occurrence of a variable from B, i.e.,

$$\forall 1 \leq j \leq N.\ \sum_{c \in V} \Omega(j)(c) = 0 \ \vee \ \sum_{c \in B} \Omega(j)(c) = 1.$$

Third, each dashed line must have a label from B. That is, when $L = \{(i,j) \mid 1 \leq i \leq M \wedge 1 \leq j \leq N\}$,

$$\forall (i,j) \in L.\ \Pi(i) \not\geq \Omega(j) \implies \exists c \in B.\ \Pi(i)(c) < \Omega(j)(c).$$

Proof: Assume that we have an implication of the form (1) in Section 4 and an appropriate environment η, that the stated criterion on the variable layout holds and that we have unary η-validity. We must show that the PC holds.

According to Definition 6 we assume that we have heaps $h, h_1, \ldots, h_M \in \mathsf{Heap}$ with $h_i \sqsubseteq h$ and $h_i \in [\![\varphi_i]\!]^1_\eta$ for all $1 \leq i \leq M$. The core of the proof is the construction of a particular environment $\rho : V \to \mathsf{IRel}_1$. We define $\rho(c) = \mathsf{Heap}$ for $c \in V \setminus B$. For a variable $c \in B$ we let $\rho(c)$ be the following union

$$\bigcup_{1 \leq i \leq M, 1 \leq k \leq M_i, a_{i,k} \equiv c} h - h_i.$$

For each $1 \leq i \leq M$ we can write $h = h_i \cdot (h - h_i)$ and so we have h in $[\![\varphi_i * a_{i,1} * \cdots * a_{i,M_i}]\!]^1_{\eta,\rho}$ by the first of the original assumption on the set B. In summary, we have shown that h lies in the unary interpretation of the left hand side in the environments η and ρ. By assumption, the same must hold for the right hand side and from this we aim to derive the PC.

We now know that h lies in the interpretation of some disjunct, say disjunct j. If this disjunct is empty we have proved the second option of the PC. Otherwise we know that there is exactly one $1 \leq k \leq N_j$ such that $b_{j,k} \in B$. But then we have

$$h \in [\![\psi_j * b_{j,1} * \cdots * b_{j,N_j}]\!]^1_{\eta,\rho} = [\![\psi_j]\!]^1_\eta * \rho(b_{j,k}).$$

We write

$$h = g \cdot g_k$$

for $g \in [\![\psi_j]\!]^1_\eta$ and $g_k \in \rho(b_{j,k})$. There must be an $1 \leq i \leq M$ such that $g_k \sqsupseteq h - h_i$ and such that $\Pi(i)(b_{j,k}) = \Omega(j)(b_{j,k}) = 1$. The first gives $h_i \in [\![\psi_j]\!]^1_\eta$ and the second implies $\Pi(i) \geq \Omega(j)$ by the third assumption on B. And we have arrived at the first option of the PC. ■

D Completeness

Lemma 19 (Fan-Counter) *Suppose that the layout of variables is as follows. There are at least two conjuncts, i.e., $M \geq 2$, and one conjunct has the property that each variable occurring in the conjunct also occurs as a label of a solid line leaving the conjunct and ending in a non-empty disjunct. In symbols the latter is*

$$\begin{array}{l} \exists 1 \leq i \leq M.\ \forall c \in V.\ \Pi(i)(c) > 0 \implies \\ \quad \exists 1 \leq j \leq N.\ \Pi(i) \geq \Omega(j) \wedge \Pi(i)(c) > \Omega(j)(c) \ \wedge \\ \quad\quad \exists d \in V.\ \Omega(j)(d) > 0. \end{array}$$

Then there are choices of φ_i's and ψ_j's with no variables such that the implication holds on the unary level but not on the binary level.

In the search for counterexamples we may without loss of generality assume the negation of the conditions of the above lemma. This means, provided at least two conjuncts, that for any non-empty set of solid lines leaving one common conjunct and ending in non-empty disjuncts there is a variable that occurs in the conjunct

but is not a label of either of the lines. If, loosely phrased, we invalidate that variable, then all the solid lines break down, i.e., become dashed.

Lemma 20 (X-Counter) *Suppose that the layout of variables is as follows. There are two distinct conjuncts i_0 and i_1 and two distinct non-empty disjuncts j_0 and j_1 such that $\Pi(i_0) \not\geq \Omega(j_0)$ while $\Pi(i_0) \geq \Omega(j_1)$, $\Pi(i_1) \geq \Omega(j_0)$ and $\Pi(i_1) \geq \Omega(j_1)$. Then there are choices of φ_i's and ψ_j's that the implication holds on the unary level but not on the binary level.*

Again we may without loss of generality assume that the negation of this lemma holds when building counterexamples. Picture the graph of the implication without empty disjuncts and without dashed lines. The negation of the above means that we may arrive at all vertices in the connected component containing some vertex by paths from that vertex of length 2 or less. Also all connected components are complete, in particular no two vertices with a dashed line between them can belong to the same component.

Lemma 21 (Bridge-Counter) *Suppose that the layout of variables is as follows. There are at least two conjuncts, i.e., $M \geq 2$, all disjuncts are non-empty and there is a dashed line with labels that all occur as labels on solid lines too. In symbols the last demand is*

$$\begin{array}{l}\exists 1 \leq i \leq M.\ \exists 1 \leq j \leq N.\ \Pi(i) \not\geq \Omega(j) \wedge \\ \quad \forall c \in V.\ \Pi(i)(c) < \Omega(j)(c) \implies \\ \quad\quad \exists 1 \leq k \leq M.\ \exists 1 \leq l \leq N. \\ \quad\quad\quad \Pi(k) \geq \Omega(l) \wedge \Pi(k)(c) > \Omega(l)(c).\end{array}$$

Then there are choices of φ_i's and ψ_j's with no variables such that the implication holds on the unary level but not on the binary level.

This lemma deals with the case of a variable layout with at least two conjuncts and no empty disjuncts but where the first condition of Theorem 8 fails.

Lemma 22 (All-Out-Counter) *Suppose that the layout of variables is as follows. There are at least two conjuncts, i.e., $M \geq 2$, at least one non-empty disjunct and for each variable one of the following two holds: Either the variable occurs as a label on a solid line ending in a non-empty disjunct. Or it occurs at least twice in a conjunct and we have an empty disjunct. In symbols the variable condition is*

$$\begin{array}{l}\forall c \in V.\ \big(\exists 1 \leq i \leq M.\ \exists 1 \leq j \leq N.\ \Pi(i) \geq \Omega(j) \wedge \\ \quad \Pi(i)(c) > \Omega(j)(c) \wedge \exists d \in V.\ \Omega(j)(d) > 0\big) \vee \\ \quad \big(\exists 1 \leq i \leq M.\ \Pi(i)(c) \geq 2 \wedge \\ \quad\quad \exists 1 \leq j \leq N.\ \forall d \in V.\ \Omega(j)(d) = 0\big).\end{array}$$

Then there are choices of φ_i's and ψ_j's φ_i's and ψ_j's with no variables such that the implication holds on the unary level but not on the binary level.

This lemma deals with two cases. The first is the case of a variable layout with at least two conjuncts and no empty disjuncts but where the second condition of Theorem 8 fails while the first holds. The second is the case of a variable layout with at least two conjuncts, at least one empty disjunct and no dashed lines for which Theorem 10 fails.

E Higher Arities and Parametricity

Theorem 15 *For an implication of the form (1) and an appropriate environment η we have that n-ary η-validity implies the PC if $n \geq \max\{M_1, \ldots, M_M\}$.*

Proof: Assume that we have an implication of the form (1) in Section 4 and an appropriate environment η, that $n \geq \max\{M_1, \ldots, M_M\}$ and that we have n-ary η-validity. We must show that the PC holds. We consider only the case $n \geq 2$, the case $n = 1$ proceeds along the lines of the proof of Theorem 10.

According to Definition 6 we assume that we have heaps $h, h_1, \ldots, h_M \in \mathsf{Heap}$ with $h_i \sqsubseteq h$ and $h_i \in [\![\varphi_i]\!]^1_\eta$ for all $1 \leq i \leq M$. The core of the proof is the construction of a particular environment $\rho : V \to \mathsf{IRel}_n$. For that purpose we need some notation. Define, for each $1 \leq k \leq n$, a map $\gamma_k : \mathsf{Heap} \to \mathsf{Heap}^n$ by letting

$$\gamma_k(h) = (\overbrace{[], \ldots, []}^{k-1}, h, \overbrace{[], \ldots, []}^{n-k})$$

for any $h \in \mathsf{Heap}$, i.e., it returns the n-tuple that has h as the k-th entry and the empty heap everywhere else. Similarly, we define $\delta : \mathsf{Heap} \to \mathsf{Heap}^n$ by setting

$$\delta(h) = (\overbrace{h, \ldots, h}^{n})$$

for any $h \in \mathsf{Heap}$, i.e., it returns the n-tuple that has h as all entries. For a subset $M \subseteq \mathsf{PosInt}$ we denote by $[M]$ the heap that has domain M and stores some fixed value, say 0, at all these locations.

For a variable $c \in V$ we now define $\rho(c)$ to be the following union of relations in IRel_n:

$$\bigcup_{1 \leq i \leq M, 1 \leq k \leq M_i, a_{(i,k)} \equiv c} \{\gamma_k(h - h_i) \cdot \gamma_1([\mathbf{S}^{M,K}_{i,k} + L])\}^\uparrow$$

where $K = \max\{M_1, \ldots, M_M\}$, $L = \max(\mathrm{dom}(h))$. This is well-defined because of our assumption that $n \geq \max\{M_1, \ldots, M_M\}$. For each $1 \leq i \leq M$ we have that

$$\begin{array}{rl}\delta(h) &= \delta(h_i) \cdot \gamma_1(h - h_i) \cdot \cdots \cdot \gamma_n(h - h_i) \\ &\sqsupseteq \delta(h_i) \cdot \gamma_1(h - h_i) \cdot \cdots \cdot \gamma_{M_i}(h - h_i)\end{array}$$

where we use the extension order for heap tuples defined by pointwise extension in all entries. Also, we have that

$$[\mathbf{S}^{M,K} + L] = [\mathbf{S}^{M,K}_{i,1} + L] \cdot \cdots \cdot [\mathbf{S}^{M,K}_{i,K} + L] \\ \sqsupseteq [\mathbf{S}^{M,K}_{i,1} + L] \cdot \cdots \cdot [\mathbf{S}^{M,K}_{i,M_i} + L].$$

This gives us that $\delta(h) \cdot \gamma_1([\mathbf{S}^{M,K} + L])$ extends the following n-tuple of heaps:

$$\delta(h_i) \cdot \prod_{1 \le k \le M_i} \gamma_k(h - h_i) \cdot \gamma_1([\mathbf{S}^{M,K}_{i,k} + L])$$

which again means that $\delta(h) \cdot \gamma_1([\mathbf{S}^{M,K} + L])$ lies in

$$[\![\varphi_i * a_{i,1} * \cdots * a_{i,M_i}]\!]^n_{\eta,\rho}.$$

In summary, we have shown that $\delta(h) \cdot \gamma_1([\mathbf{S}^{M,K} + L])$ lies in the n-ary interpretation of the left hand side in the environments η and ρ. By assumption, the same must hold for the right hand side and from this we aim to derive the PC.

There is $1 \le j \le N$ such that we have

$$\delta(h) \cdot \gamma_1([\mathbf{S}^{M,K} + L]) \in [\![\psi_j * b_{j,1} * \cdots * b_{j,N_j}]\!]^n_{\eta,\rho}.$$

Consider first the case of a non-empty disjunct, i.e., the case $N_j > 0$. We split along the disjunct and get

$$\delta(h) \cdot \gamma_1([\mathbf{S}^{M,K} + L]) = \delta(g) \cdot \mathbf{g}_1 \cdot \cdots \cdot \mathbf{g}_{N_j}$$

for $g \in [\![\psi_j]\!]^1_\eta$ and $\mathbf{g}_k \in \rho(b_{j,k})$ for all $1 \le k \le N_j$. By the properties of segregating sets we get that there must be a common $1 \le i \le M$ such that for all $1 \le k \le N_k$ there is $1 \le k_k \le M_i$ with

$$\gamma_{k_k}(h - h_i) \cdot \gamma_1([\mathbf{S}^{M,K}_{i,k_k} + L]) \sqsubseteq \mathbf{g}_k,$$

i.e., the $\mathbf{g}_k$'s are all 'from the same conjunct'. But this implies $\Pi(i) \ge \Omega(j)$ as the segregating sets are non-empty. Also the above equality enforces $\mathrm{dom}(g) \subseteq \mathrm{dom}(h)$ by the definition of γ_1. Indeed we must have $\mathrm{dom}(g) \subseteq \mathrm{dom}(h_i)$ since in particular we have

$$\gamma_{1_k}(h - h_i) \cdot \gamma_1([\mathbf{S}^{M,K}_{i,1_k} + L]) \sqsubseteq \mathbf{g}_1.$$

But then $g \sqsubseteq h_i$ so we have $h_i \in [\![\psi_j]\!]^1_\eta$ too and the first option of the PC holds.

We consider now the case of an empty disjunct, i.e., the case $N_j = 0$. As above we split along the disjunct and get

$$\delta(h) \cdot \gamma_1([\mathbf{S}^{M,K} + L]) = \delta(g) \cdot \mathbf{g}$$

for $g \in [\![\psi_j]\!]^1_\eta$ and $\mathbf{g} \in \mathsf{Heap}^n$. Again we must have $\mathrm{dom}(g) \subseteq \mathrm{dom}(h)$ which implies $g \sqsubseteq h$ and the second option of the PC holds. ■

F Proof of Theorem 17

Theorem 17 *Every derivable* $\Gamma \vdash \{\varphi\}C\{\psi\}$ *is 2-valid.*

Proof: We will show that all the rules in Figure 4 are sound. This lets us prove the theorem by induction on the height of the derivation of a judgment, because using the soundness of the rules, we can handle all the base and inductive cases.

Let's start with the rule for the module operation k. Suppose that $(\eta, \rho, \mathbf{u}) \models^2 (\Gamma, \{\varphi\}k\{\psi\})$. Then, by the definition of $\models^2$, we should have $(\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}k\{\psi\}$ as well. From this follows the soundness of the rule.

Next, consider four rules: (a) the frame rule for adding $- * \varphi$ to the pre and post-conditions, (b) the rule for adding $\exists x$ to the pre and post-conditions, (c) the rule for sequencing, and (d) the rule for the conditional statement. All these rules are sound, because of the following four facts:

$$(\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}C\{\varphi'\} \\ \quad \Longrightarrow (\eta, \rho, \mathbf{u}) \models^2 \{\varphi * \psi\}C\{\varphi' * \psi\}$$

$$(x \notin \mathsf{FV}(C)) \ \wedge \ (\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}C\{\psi\} \\ \quad \Longrightarrow (\eta, \rho, \mathbf{u}) \models^2 \{\exists x.\varphi\}C\{\exists x.\psi\}$$

$$(\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}C\{\varphi'\} \ \wedge \ (\eta, \rho, \mathbf{u}) \models^2 \{\varphi'\}C'\{\psi\} \\ \quad \Longrightarrow (\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}C; C'\{\psi\}$$

$$(\eta, \rho, \mathbf{u}) \models^2 \{\varphi \wedge B\}C\{\psi\} \ \wedge \\ (\eta, \rho, \mathbf{u}) \models^2 \{\varphi \wedge \neg B\}C'\{\psi\} \\ \quad \Longrightarrow (\eta, \rho, \mathbf{u}) \models^2 \{\varphi\}\mathbf{if}\ B\ C\ C'\{\psi\}$$

The first fact is an easy consequence of using the quantification over IRel_2 in the semantics of triples. The second also follows easily from the semantics of triples and the fact that $[\![C]\!]_{\eta,u_i} = [\![C]\!]_{\eta[x \mapsto v],u_i}$ for all $v \in \mathsf{Int}$, as long as one remembers that the $*$ operator distributes over union. The third and fourth are not different, and they follow from the semantics of triples and commands. Here we will go through the details of proving the fourth fact. Consider $(\eta, \rho, \mathbf{u})$ satisfying the assumption of the fact. Pick $r \in \mathsf{IRel}_2$ and heaps f, g such that

$$(f, g) \ \in \ [\![\varphi]\!]^2_{\eta,\rho} * r.$$

Now we do the case analysis on whether $[\![B]\!]_\eta$ is true or not. If it is true, then

$$(f, g) \ \in \ [\![\varphi \wedge B]\!]^2_{\eta,\rho} * r \\ \ \wedge \ \ [\![\mathbf{if}\ B\ C\ C']\!]_{\eta,u_1}(f) \ = \ [\![C]\!]_{\eta,u_1}(f) \\ \ \wedge \ \ [\![\mathbf{if}\ B\ C\ C']\!]_{\eta,u_2}(g) \ = \ [\![C]\!]_{\eta,u_2}(g).$$

Hence, by assumption, we get that

$$\begin{aligned}&\big(\,[\![\mathbf{if}\,B\,C\,C']\!]_{\eta,u_1}(f),\ [\![\mathbf{if}\,B\,C\,C']\!]_{\eta,u_2}(g)\,\big)\\ &\qquad=\ \big(\,[\![C]\!]_{\eta,u_1}(f),\ [\![C]\!]_{\eta,u_2}(g)\,\big)\ \in\ [\![\psi]\!]^2_{\eta,\rho}*r.\end{aligned}$$

If $[\![B]\!]_\eta$ is not true, we reason similarly, but with C' instead of C, and get that

$$\big(\,[\![\mathbf{if}\,B\,C\,C']\!]_{\eta,u_1}(f),\ [\![\mathbf{if}\,B\,C\,C']\!]_{\eta,u_2}(g)\,\big)\ \in\ [\![\psi]\!]^2_{\eta,\rho}*r.$$

We have just shown that in both cases, the outcomes of the conditional statements are related by $[\![\psi]\!]^2_{\eta,\rho}*r$, as claimed by the fourth fact.

We move on to the rules for heap update and dereference. They are sound because of the below two facts:

$$(\eta,\rho,\mathbf{u})\models^2\{E\hookrightarrow\_\}[E]{:=}F\{E\hookrightarrow F\}$$

$$\begin{aligned}&x\notin\mathsf{FV}(\psi)\ \wedge\ (\eta,\rho,\mathbf{u})\models^2\{\varphi*E\hookrightarrow x\}\mathbf{let}\,x{=}[E]\,\mathbf{in}\,C\{\psi\}\\ &\quad\Longrightarrow\ (\eta,\rho,\mathbf{u})\models^2\{\exists x.\varphi*E\hookrightarrow x\}\mathbf{let}\,x{=}[E]\,\mathbf{in}\,C\{\psi\}\end{aligned}$$

To prove the first, we pick $(\eta,\rho,\mathbf{u})$, a relation $r\in\mathsf{IRel}_2$ and heaps f,g such that

$$(f,g)\ \in\ [\![E\hookrightarrow\_]\!]^2_{\eta,\rho}*r.$$

Then, there exist heaps h,f_1,g_1 such that

$$(f,g)=(h,h)\cdot(f_1,g_1)\ \wedge\ [\![E]\!]_\eta\in\mathrm{dom}(h)\ \wedge\ (f_1,g_1)\in r.$$

Thus,

$$\begin{aligned}&\big(\,[\![[E]{:=}F]\!]_{\eta,u_1}(f),\ [\![[E]{:=}F]\!]_{\eta,u_2}(g)\,\big)\\ &\qquad=\ \big(\,h[[\![E]\!]_\eta{\mapsto}[\![F]\!]_\eta]\cdot f_1,\ h[[\![E]\!]_\eta{\mapsto}[\![F]\!]_\eta]\cdot g_1\,\big)\\ &\qquad\in\ [\![E{\hookrightarrow}F]\!]^2_{\eta,\rho}\ *\ r,\end{aligned}$$

as desired by the first fact. For the proof of the second fact, suppose that the assumption of the second fact holds, and pick $r\in\mathsf{IRel}_2$ and heaps f,g such that

$$(f,g)\ \in\ [\![\exists x.\,\varphi*E{\hookrightarrow}x]\!]^2_{\eta,\rho}*r.$$

Then, there exists an integer v and heaps h,f_1,g_1 such that

$$\begin{aligned}(f,g)=(h,h)\cdot(f_1,g_1)\ &\wedge\ (f_1,g_1)\in r\\ \wedge\ h\in[\![\varphi*E{\hookrightarrow}x]\!]^1_{\eta[x\mapsto v],\rho}&.\end{aligned}$$

Thus, $(f,g)\in[\![\varphi*E{\hookrightarrow}x]\!]^1_{\eta[x\mapsto v],\rho}\ *\ r$, and $f([\![E]\!]_\eta)=g([\![E]\!]_\eta)=v$. Using these and the assumed triple of the second fact, we derive the below:

$$\begin{aligned}&\big(\,[\![\mathbf{let}\,x{=}[E]\,\mathbf{in}\,C]\!]_{\eta,u_1}(f),\ [\![\mathbf{let}\,x{=}[E]\,\mathbf{in}\,C]\!]_{\eta,u_2}(g)\,\big)\\ &\qquad=\ \big(\,[\![C]\!]_{\eta[x\mapsto v],u_1}(f),\ [\![C]\!]_{\eta[x\mapsto v],u_2}(g)\,\big)\\ &\qquad\in\ [\![\psi]\!]^2_{\eta[x\mapsto v],\rho}*r\\ &\qquad=\ [\![\psi]\!]^2_{\eta,\rho}*r.\end{aligned}$$

The last equality holds, because x does not appear in φ. We have just proved that the output states of two dereferencing commands are $([\![\psi]\!]^2_{\eta,\rho}*r)$-related, as claimed by the second fact.

Finally, we prove that the rule of consequence is sound. It is sufficient to show that

$$\begin{aligned}&\textsc{Chk}(\varphi',\varphi)\ \wedge\ \varphi'\models^1\varphi\ \wedge\\ &\textsc{Chk}(\psi,\psi')\ \wedge\ \psi\models^1\psi'\ \wedge\ (\eta,\rho,\mathbf{u})\models^2\{\varphi\}C\{\psi\}\\ &\quad\Longrightarrow\ (\eta,\rho,\mathbf{u})\models^2\{\varphi'\}C\{\psi'\}.\end{aligned}$$

From the first four conjuncts of the assumption, it follows that

$$\varphi'\models^2\varphi\quad\wedge\quad\psi\models^2\psi'.$$

This is due to the correctness of Chk, which holds because all the transformations used in the first check of Chk are based on semantic equivalences holding in IRel_2 and the second lifting check is sound because of our lifting theorems. In order to prove the conclusion of the above implication, pick $r\in\mathsf{IRel}_2$ and heaps f,g such that

$$(f,g)\ \in\ [\![\varphi']\!]^2_{\eta,\rho}*r.$$

Since the $*$ operator is monotone and $\varphi'\models^2\varphi$, we get that

$$(f,g)\ \in\ [\![\varphi]\!]^2_{\eta,\rho}*r.$$

This relationship and the assumed triple $\{\varphi\}C\{\psi\}$, then, imply the below:

$$\big(\,[\![C]\!]_{\eta,u_1}(f),\,[\![C]\!]_{\eta,u_2}(g)\,\big)\ \in\ [\![\psi]\!]^2_{\eta,\rho}*r.$$

Again, since $\psi\models^2\psi'$, the monotonicity of the $*$ operator implies that

$$\big(\,[\![C]\!]_{\eta,u_1}(f),\,[\![C]\!]_{\eta,u_2}(g)\,\big)\ \in\ [\![\psi']\!]^2_{\eta,\rho}*r.$$

Note that this is the conclusion that we are looking for. ■

Chapter 8

A Tale of Two Recursive Predicates (note)

A Tale of Two Recursive Predicates

Jacob Thamsborg, thamsborg@itu.dk, May 9, 2010

Flat Locations and Recursion

Consider – without further ado – some language with integers, recursive types and general reference types, i.e.,

$$\tau ::= \alpha \mid \texttt{int} \mid \texttt{ref}\ \tau \mid \mu\alpha.\,\tau \mid \cdots$$

and a set V with maps $in_{\texttt{int}} : \mathbb{Z} \to V$, $in_{\texttt{ref}} : \omega \to V$, and $in_\mu : V \to V$ with disjoint images; we have, e.g, $in_{\texttt{int}}(\mathbb{Z}) \cap in_{\texttt{ref}}(V) = \emptyset$.

We should like to have a world-indexed interpretation of the closed types as subsets of V such that

$$[\![\texttt{int}]\!](\Delta) = in_{\texttt{int}}(\mathbb{Z}),$$

such that

$$\begin{aligned}&[\![\texttt{ref}\ \tau]\!](\Delta)\\ &\quad = in_{\texttt{ref}}\big(\{l \in \mathrm{dom}(\Delta) \mid [\![\Delta(l)]\!](\Delta) = [\![\tau]\!](\Delta)\}\big)\end{aligned}$$

and finally such that

$$[\![\mu\alpha.\,\tau]\!](\Delta) = in_\mu\big([\![\tau[\mu\alpha.\,\tau/\alpha]]\!](\Delta)\big).$$

Here the Δ ranges over worlds, i.e., partial, finite maps from ω to closed syntactic types. We stick, as in [3], to syntactic worlds for simplicity.

Looking for trouble, we assume the existence of an interpretation with the stated properties and focus on the type τ given by

$$\mu\alpha.\,\texttt{ref}\ \alpha$$

in the context of the world Δ given by

$$0 \mapsto \texttt{ref int}.$$

For an easy start, we take a shot at interpreting the type stored at location zero:

$$\begin{aligned}&[\![\Delta(0)]\!](\Delta)\\ &\quad = [\![\texttt{ref int}]\!](\Delta)\\ &\quad = in_{\texttt{ref}}(\{l \in \{0\} \mid [\![\Delta(l)]\!](\Delta) = [\![\texttt{int}]\!](\Delta)\})\\ &\quad = \begin{cases} in_{\texttt{ref}}(\{0\}) & \text{if } [\![\Delta(0)]\!](\Delta) = [\![\texttt{int}]\!](\Delta)\\ \emptyset & \text{if } [\![\Delta(0)]\!](\Delta) \neq [\![\texttt{int}]\!](\Delta)\end{cases}\end{aligned}$$

We have, unsurprisingly, hit a circularity and direct calculations will get us nowhere. But we must have $[\![\Delta(0)]\!](\Delta) \subseteq in_{\texttt{ref}}(\omega)$ and hence the former cannot equal $[\![\texttt{int}]\!](\Delta) = in_{\texttt{int}}(\mathbb{Z})$, since, e.g., $in_{\texttt{int}}(\mathbb{Z}) \ni in_{\texttt{int}}(0) \notin in_{\texttt{ref}}(\omega)$. Hence we conclude that $[\![\Delta(0)]\!](\Delta) = \emptyset$.

But how, then, about interpreting τ in the world Δ? Being a reference type in the context of a world with singleton domain leaves little room for diversity: either it must be empty

$$[\![\tau]\!](\Delta) = \emptyset$$

or it must contain the single allocated location

$$[\![\tau]\!](\Delta) = in_\mu(in_{\texttt{ref}}(\{0\})).$$

The first guess is arguably the most tempting, but it is easily seen from the following that either possibility implies the other:

$$\begin{aligned}&[\![\tau]\!](\Delta)\\ &\quad = [\![\mu\alpha.\,\texttt{ref}\ \alpha]\!](\Delta)\\ &\quad = in_\mu([\![\texttt{ref}\ \tau]\!](\Delta))\\ &\quad = in_\mu(in_{\texttt{ref}}(\{l \in \{0\} \mid [\![\Delta(l)]\!](\Delta) = [\![\tau]\!](\Delta)\}))\\ &\quad = \begin{cases} in_\mu(in_{\texttt{ref}}(\{0\})) & \text{if } [\![\Delta(0)]\!](\Delta) = [\![\tau]\!](\Delta)\\ \emptyset & \text{if } [\![\Delta(0)]\!](\Delta) \neq [\![\tau]\!](\Delta).\end{cases}\end{aligned}$$

We arrive at the contradiction $\emptyset = in_\mu(in_{\texttt{ref}}(\{0\}))$ and are forced to conclude that there is no such interpretation.

A natural work-around would be to consider syntactic rather than semantic equality when interpreting reference types, i.e., to demand

$$[\![\texttt{ref}\ \tau]\!](\Delta) = in_{\texttt{ref}}\big(\{l \in \mathrm{dom}(\Delta) \mid \Delta(l) = \tau\}\big)$$

of the interpretation of reference types, where the last equality is syntactical. But proceeding along those lines we get into trouble when extending to open types since we cannot deal with non-syntactic types – we get no proper space of semantic types. In particular, we obstruct relational parametric reasoning in an envisioned binary interpretation.

The problem appears to persist in the presence of semantic worlds as in [2]. Also, it can be replayed with a more realistic version of τ such as linked lists of integers if we equip the language with sum, product and unit types. A serious shortcoming – so to speak – of the counter-example is, however, that it does not scale to a more realistic interpretation of reference types that quantify over future worlds as used in [3, 2]. In other words, if we change our demand on reference types to be

$$[\![\texttt{ref}\ \tau]\!](\Delta) = in_{\texttt{ref}}\big(\{l \in \mathrm{dom}(\Delta) \mid \forall \Delta' \sqsupseteq \Delta.\, [\![\Delta(l)]\!](\Delta') = [\![\tau]\!](\Delta')\}\big)$$

then the possibility $[\![\tau]\!](\Delta) = \emptyset$ is no longer (immediately) inconsistent. Here we write $\Delta' \sqsupseteq \Delta$ for two worlds Δ and Δ' to indicate that $\mathrm{dom}(\Delta) \subseteq \mathrm{dom}(\Delta')$ and that that the worlds agree on the domain of Δ.

Let us take our game to the next level. Assume the above interpretation of reference types with quantification over future worlds, the original interpretation of recursive types, and an additional type constructor `nozero` interpreted as follows:

$$[\![\texttt{nozero}\ \tau]\!](\Delta) = [\![\tau]\!](\Delta) \setminus \{in_{\texttt{loc}}(0)\}.$$

We reuse τ from above, but redefine Δ to be the world

$$0 \mapsto \texttt{nozero}\ \tau.$$

By calculations as above, it is not hard to see that for any $\Delta' \sqsupseteq \Delta$ we have that $in_\mu(in_{\texttt{ref}}(0)) \in [\![\tau]\!](\Delta')$ implies $[\![\texttt{nozero}\ \tau]\!](\Delta') = [\![\tau]\!](\Delta')$, an unfortunate combination. On the other hand, $[\![\tau]\!](\Delta) = \emptyset$ implies, sadly, the existence of a world $\Delta' \sqsupseteq \Delta$ such that $in_\mu(in_{\texttt{ref}}(0)) \in [\![\tau]\!](\Delta')$.

How much did we cheat in adding the novel, to say the least, type operator? Some, at least; it is of a very semantic nature, and its applicability to a specific, alleged model cannot be determined from the (intentionally) vague requirements we make on models here. Having said that, it works in practice and shows that if we go with flat locations in either [3] or [2] then the logical relations do not exist.

The second counter-example was discovered independently by the author and Kristian Støvring based on a related, but rather more complex and semantic, counter-example by Kristian Støvring.

A Quest for Purity

The following question was posed by Benton, Kennedy, Beringer and Hofmann [1, Section 6.2]: Let $i : A \to A_\perp \cong A$ be the minimal invariant of the predomain functor $(-) \to (-)_\perp$; is it possible to find a (necessarily chain-complete) subset $P \subseteq A$ of A with

$$a \in P \iff \forall b \in P.\, i^{-1}(a)(p) \in \{\perp, \lfloor b \rfloor\}\,?$$

Intuitively, A models commands that work on stores with a single global cell and P – if it exists – singles out the *hereditarily read-only* commands: they may freely read the contents of the cell and decide to terminate or not on that basis, but they may not change the contents. Provided, in all fairness, that the contents of the cell is itself a pure command; hence the recursive requirement and hence the name.

To the best of the knowledge of the author, this question remains unanswered. It is, however, tempting to ask if the technique of approximated (or semantic) locations coined by Birkedal, Støvring and Thamsborg [3, 2] could be adapted to this setting. After all, the existence of a similarly recursive predicate is their *raison d'etre*. And indeed, as we shall see, it is possible

to proceed along those lines to obtain something that approximates P above. This, unfortunately, does not answer the original question and whether an approximated solution suffices for the larger purpose of the cited paper is not at all clear.

Let $\omega = \{0, 1, 2, \ldots\}$ and $\bar{\omega}_1 = \{1, 2, 3, \ldots, \infty\}$. Define $\mathcal{T} \subseteq \mathcal{P}(\bar{\omega}_1 \times A)$ to be the set of subsets $P \subseteq \bar{\omega}_1 \times A$ that are *downwards closed*, i.e.,

$$\forall (n, a) \in P. \forall 1 \leq m < n. (m, a) \in P$$

as well as *complete*, i.e.,

$$\forall a \in A. [\forall 1 \leq n < \infty. (n, a) \in P] \implies (\infty, a) \in P.$$

If, for any $P, Q \in \mathcal{T}$ and $n \in \omega$, it is the case that $(a, m) \in P$ iff $(a, m) \in Q$ for all $a \in A$ and all $1 \leq m \leq n$ we write $P =_n Q$; note that $P =_0 Q$ always holds. It is easy to show the following:

Proposition 1. *There is a (unique) complete ultrametric $d : \mathcal{T} \times \mathcal{T} \to \{0\} \cup \{2^{-n} \mid n \in \omega\}$ such that*

$$P =_n Q \iff d(P, Q) \leq 2^{-n}$$

for all $P, Q \in \mathcal{T}$ and $n \in \omega$.

Recall that, by the standard construction [4], A comes equipped with an increasing chain of continuous *projections* $(\pi_n)_{n \in \omega}$ from A to $A_\perp$ such that

$$\pi_0 = \lambda a \in A. \perp, \quad \bigsqcup_{n \in \omega} \pi_n = \lambda a \in A. \lfloor a \rfloor$$

and such that

$$\bar{\pi}_n \circ \pi_m = \bar{\pi}_m \circ \pi_n = \pi_{n \wedge m}$$

for any $n, m \in \omega$; for any continuous map $\varphi : A \to A_\perp$, we denote by $\bar{\varphi} : A_\perp \to A_\perp$ the strict extension. For ease of notation we furthermore set $\pi_\infty = \lambda a \in A. \lfloor a \rfloor$, this extension does not invalidate the last property of the projections.

Using these projections we define $\Phi : \mathcal{T} \to \mathcal{T}$ by

$$\Phi(P) = \big\{(n, a) \mid \forall (m, b) \in P. \\ \bar{\pi}_{(n-1) \wedge m}(i^{-1}(a)(b)) \in \{\perp, \pi_{(n-1) \wedge m}(b)\}\big\}$$

for any $P \in \mathcal{T}$; it is not hard to prove Φ well-defined using the properties of the projections.

We now proceed to the main result:

Proposition 2. *$\Phi : \mathcal{T} \to \mathcal{T}$ is contractive.*

Proof. Take any two $P, Q \in \mathcal{T}$ and any $n \in \omega$, it shall suffice to prove that $P =_n Q$ implies $\Phi(P) =_{n+1} \Phi(Q)$. So we assume $P =_n Q$, take an arbitrary $(m, a) \in \Phi(P)$ with $m \leq n + 1$ and must prove that we have $(m, a) \in \Phi(Q)$ too; symmetry makes short work of the other direction.

By definition, we pick $(l, b) \in Q$ and must show that

$$\bar{\pi}_{(m-1) \wedge l}(i^{-1}(a)(b)) \in \{\perp, \pi_{(m-1) \wedge l}(b)\}.$$

If $n = 0$ it must be the case that $m = 1$ and we get $\bar{\pi}_{(m-1) \wedge l}(i^{-1}(a)(b)) = \bar{\pi}_0(i^{-1}(a)(b)) = \perp$. If $l \leq n$ we have $(l, b) \in P$ and we are done since $(m, d) \in \Phi(P)$. Otherwise we have $l > n \geq 1$. By downwards closure of Q we must have $(n, b) \in Q$ too and so $(n, b) \in P$ and we have

$$\bar{\pi}_{(m-1) \wedge n}(i^{-1}(a)(b)) \in \{\perp, \pi_{(m-1) \wedge n}(b)\}.$$

Luckily, we also have $(m - 1) \wedge l = m - 1 = (m - 1) \wedge n$ and we are done. □

By Banach's fixed-point theorem we have the following corollary:

Corollary 3. *There is $P' \in \mathcal{T}$ such that*

$$(n, a) \in P' \iff \forall (m, b) \in P'. \\ \bar{\pi}_{(n-1) \wedge m}(i^{-1}(a)(b)) \in \{\perp, \pi_{(n-1) \wedge m}(b)\}.$$

Consider now elements of P' that have infinite index; these are the closest thing we get to hereditary read-only commands. Take one such, say $(\infty, a) \in P'$; for any other $(\infty, b) \in P'$ we do have $i^{-1}(a)(b) \in \{\perp, \lfloor b \rfloor\}$ as desired. The converse, however, does not hold: to prove $(\infty, a) \in P'$ it does not suffice to test against all $(\infty, b) \in P'$, we have to consider elements with finite index too. We rationalize: elements with finite index $1 \leq n < \infty$ are intuitively read-only up to equality under π_{n-1}; it is, therefore, fair that proper read-only commands behave up to equality under π_n when applied to such. The index-increase is due to our 'folding' of the argument as member of A rather than a function.

By loose analogy with approximated locations, a function of type $\mathit{ref}\ \tau \to \sigma$ must function for all values in the interpretation of $\mathit{ref}\ \tau$, be they proper or approximated locations. Hence such functions must have some approximated behavior in the latter case, e.g., applying the appropriate projection to whatever it looks up in the memory.

References

[1] Nick Benton, Andrew Kennedy, Lennart Beringer, and Martin Hofmann. Relational semantics for effect-based program transformations: higher-order store. In António Porto and Francisco Javier López-Fraguas, editors, *PPDP*, pages 301–312. ACM, 2009.

[2] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In Luca de Alfaro, editor, *FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 456–470. Springer, 2009.

[3] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. Relational parametricity for references and recursive types. In Andrew Kennedy and Amal Ahmed, editors, *TLDI*, pages 91–104. ACM, 2009.

[4] Andrew M. Pitts. Relational properties of domains. *Inf. Comput.*, 127(2):66–90, 1996.

Chapter 9

From M-categories to O-categories (note)

From M-categories to O-categories

Jacob Thamsborg, thamsborg@itu.dk, May 9, 2010

1 Banach from Kleene

In the context of an ultrametric space, say (X, d), we write $x =_n y$ to denote that $d(x, y) \leq 2^{-n}$ for any two elements x and y of X and any $n \in \mathbb{N}$. This is an equivalence relation by the ultrametric inequality. For convenience we shall furthermore denote equality on-the-nose by $=_\infty$. Indeed, it simplifies notation in general to allow indices ro range over $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ and not just over $\mathbb{N}$. We consider $\mathbb{N}_\infty$ partially ordered in the obvious way. Addition is extended too, to cover $n + \infty = \infty + n = \infty$ for any $n \in \mathbb{N}_\infty$; we still have that $n \leq m$ implies $n + 1 \leq m + 1$.

Definition 1. The *qualification* of a 1-bounded ultrametric space (X, d) is the partial order

$$\overline{(X, d)} = \left(\sum_{n \in \mathbb{N}_\infty} X/ =_n, \sqsubseteq \right)$$

where

$$(n, [x]_n) \sqsubseteq (m, [y]_m) \iff n \leq m \wedge [x]_n \supseteq [y]_m \iff n \leq m \wedge x =_n y.$$

Elements with index ∞ are called *ideal*, the remaining are *shadow* elements.

Notice that if (X, d) is non-empty then $\overline{(X, d)}$ is non-empty too, indeed it is pointed since for any $x \in X$ we have that $(0, [x]_0)$ is the least element.

Proposition 2. The qualification of a *complete*, 1-bounded ultrametric space (X, d) is a *complete* partial order. Indeed, for any increasing chain of elements

$$(n_0, [x_0]_{n_0}) \sqsubseteq (n_1, [x_1]_{n_1}) \sqsubseteq (n_2, [x_2]_{n_2}) \sqsubseteq \cdots$$

we either have that the set of indices is *finitely bounded*, i.e., $\exists N \in \mathbb{N} \forall m \in \mathbb{N}.\ n_m < N$, in which case the chain is constant from a certain point, or we have have that

$$\bigsqcup_m (n_m, [x_m]_{n_m}) = (\infty, [\lim_m x_m]_\infty).$$

Proof. Assume that we have an increasing chain of elements

$$(n_0, [x_0]_{n_0}) \sqsubseteq (n_1, [x_1]_{n_1}) \sqsubseteq (n_2, [x_2]_{n_2}) \sqsubseteq \cdots .$$

This means that we have

$$n_0 \leq n_1 \leq n_2 \leq \cdots \quad \text{and} \quad x_0 =_{n_0} x_1,\ x_1 =_{n_1} x_2,\ x_2 =_{n_2} x_3,\ \cdots .$$

We may without loss of generality assume that the set of indices is not finitely bounded because otherwise the chain is constant from some point. Hence, for any $N \in \mathbb{N}$ there is an $M \in \mathbb{N}$ such that $n_M \geq N$. But then for any $m \geq M$ we have that $x_M =_N x_m$ and hence $(x_m)_m$ is a Cauchy sequence which by completeness of (X, d) has a limit. This means that $(\infty, [\lim_m x_m]_\infty)$ is well-defined, it now remains to prove that it is the least upper bound.

Let $M \in \mathbb{N}$ be arbitrary. As $x_M =_{n_M} x_m$ for any $m \geq M$ we must have $x_M =_{n_M} \lim_m x_m$ and hence $(n_M, [x_M]_{n_M}) \sqsubseteq (\infty, [\lim_m x_m]_\infty)$, i.e., the candidate is an upper bound. Now take any other upper bound, it must necessarily be of the form $(\infty, [x]_\infty)$. For any $N \in \mathbb{N}$ there is an $M \in \mathbb{N}$ such that $n_M \geq N$ and hence $\lim_m x_m =_N x_M =_N x$ and so we must have $\lim_m x_m = x$, hence $(\infty, [\lim_m x_m]_\infty)$ is the unique upper bound, in particular it is the least. □

Proposition 3. Let (X, d) be a non-empty, complete, 1-bounded ultrametric space and let $f : X \to X$ be contractive with factor $\frac{1}{2}$, i.e., $d(f(x), f(y)) \leq \frac{1}{2} \cdot d(x, y)$ for all $x, y \in X$. Then f has a fixed point, i.e., there is an $x \in X$ such that $f(x) = x$.

Proof (using Kleenes' fixed point theorem). We form the qualification

$$\overline{(X, d)} = \left(\sum_{n \in \mathbb{N}_\infty} X/ =_n, \sqsubseteq \right)$$

according to Definition 1. Consider now the map $\overline{f} : \overline{(X, d)} \to \overline{(X, d)}$ defined by

$$\overline{f}(n, [x]_n) = (n + 1, [f(x)]_{n+1}),$$

which is well-defined by the assumption of contractiveness. We have monotonicity since $n \leq m$ and $x =_n y$ implies $n + 1 \leq m + 1$ and $f(x) =_{n+1} f(y)$. To prove continuity, assume that we have an increasing chain of elements

$$(n_0, [x_0]_{n_0}) \sqsubseteq (n_1, [x_1]_{n_1}) \sqsubseteq (n_2, [x_2]_{n_2}) \sqsubseteq \cdots,$$

we may without loss of generality assume that the set of indices is not finitely bounded for the same reasons as in the proof of Proposition 2. But then we have

$$\begin{aligned}
\overline{f}(\bigsqcup_m (n_m, [x_m]_{n_m})) &= \overline{f}(\infty, [\lim_m x_m]_\infty) \\
&= (\infty, [f(lim_m x_m)]_\infty) \\
&= (\infty, [lim_m f(x_m)]_\infty) \\
&= \bigsqcup_m (n_m + 1, [f(x_m)]_{n_m}) \\
&= \bigsqcup_m \overline{f}((n_m, [x_m]_{n_m})),
\end{aligned}$$

and continuity is done. By Kleene's fixed point theorem, the continuous function $\overline{f}$ on the complete pointed partial order $\overline{(X, d)}$ has a (least) fixed point, i.e., there is $(n, [x]_n) \in \overline{(X, d)}$ such that

$$(n, [x]_n) = \overline{f}((n, [x]_n)) = (n + 1, [f(x)]_{n+1}).$$

This implies that $n = \infty$, i.e., the fixed point is an ideal element, and we get $f(x) = x$. □

Kleene's fixed point theorem is also sometimes referred to as Scott's fixed point theorem. Indeed, according to John Reynolds, the latter may be the most correct in terms of giving due credit for discovery.

2 O-categories from M-categories

Definition 4. The *qualification* of an M-category $\mathcal{D}$ is the category in which the objects are pairs (n, A) of an index $n \in \mathbb{N}_\infty$ and an object A of $\mathcal{D}$ and the morphisms from (n, A) to (m, B) are the elements $(i, [f]_i)$ of the qualification of $\mathcal{D}(A, B)$ with $i \leq n \wedge m$. Given a morphism $(i, [f]_i)$ from (n, A) to (m, B) and a morphism $(j, [g]_j)$ from (m, B) to (l, C) we define composition by

$$(j, [g]_j) \circ (i, [f]_i) = (i \wedge j, [g \circ f]_{i \wedge j}).$$

Notice how we may apply qualification to the hom-sets of $\mathcal{D}$ since they come equipped with metrics turning them into (non-empty, complete,) 1-bounded ultrametric spaces.

Proposition 5. The qualification an M-category is a well-defined O-category.

Proof. Let $\mathcal{D}$ be an M-category. Let us initially verify that the qualification $\mathcal{C}$ of $\mathcal{D}$ is a well-defined category. Composition is well defined, because for objects A, B and C and morphisms $f, f' : A \to B$ and $g, g' : B \to C$ of $\mathcal{D}$ with $f =_i f'$ and $g =_j g'$ for any $i, j \in \mathbb{N}_\infty$ we have $g \circ f =_{i \wedge j} g' \circ f'$ by non-expansiveness of composition in $\mathcal{D}$. Also composition in $\mathcal{C}$ is associative because both taking the minimum in $\mathbb{N}_\infty$ and composition in $\mathcal{D}$ are associative. The identity on an object (n, X) is $(n, [1_X]_n)$ where 1_X is the identity on X in $\mathcal{D}$.

Consider now the hom-set of morphisms from an object (n, A) to another object (m, B). If we have $n = m = \infty$ then the hom-set is just the qualification of $\mathcal{D}(A, B)$ which is complete partial order by Proposition 2. If either or both of m and n are natural numbers, we consider only the subset of elements of the qualification with indices less than our equal to $n \wedge m$. But any ascending chain in this subset must have finitely bounded indices and thus be constant from some point, i.e., the subset is itself a complete partial order. It remains to prove that for any three objects (n, A), (m, B) and (l, C) of $\mathcal{C}$ we have that the composition function $\circ : \mathcal{C}((m, B), (l, C)) \times \mathcal{C}((n, A), (m, B)) \to \mathcal{C}((n, A), (l, C))$ is continuous. Monotonicity goes first: let $(i, [f]_i), (i', [f']_{i'}) \in \mathcal{C}((n, A), (m, B))$ and $(j, [g]_j), (j', [g']_{j'}) \in \mathcal{C}((m, B), (l, C))$ with

$$(i, [f]_i) \sqsubseteq (i', [f']_{i'}) \wedge (j, [g]_j) \sqsubseteq (j', [g']_{j'})$$

be given. But $i \wedge j \leq i' \wedge j'$ and also $f =_{i \wedge j} f'$ and $g =_{i \wedge j} g'$ are immediate from the definition and non-expansiveness of composition in $\mathcal{D}$ yields $g \circ f =_{i \wedge j} g' \circ f'$ and so we have

$$\begin{aligned}(j, [g]_j) \circ (i, [f]_i) &= (i \wedge j, [g \circ f]_{i \wedge j}) \\ &\sqsubseteq (i' \wedge j', [g' \circ f']_{i' \wedge j'}) \\ &= (j', [g']_{j'}) \circ (i', [f']_{i'}).\end{aligned}$$

Finally, assume that we have ascending chains

$$(i_0, [f_0]_{i_0}) \sqsubseteq (i_1, [f_1]_{i_1}) \sqsubseteq (i_2, [f_2]_{i_2}) \sqsubseteq \cdots$$

and

$$(j_0, [g_0]_{g_0}) \sqsubseteq (j_0, [g_0]_{g_0}) \sqsubseteq (j_0, [g_0]_{g_0}) \sqsubseteq \cdots$$

in $\mathcal{C}((n, A), (m, B))$ respectively $\mathcal{C}((m, B), (l, C))$. We have to show that the composition of the least upper bound equals the least upper bound of the compositions. If neither of the chains have finitely bounded indices we calculate as follows:

$$\begin{aligned}
\bigsqcup_n (j_n, [g_n]_n) \circ \bigsqcup_n (i_n, [f_n]_n) &= (\infty, [\lim_n g_n]_\infty) \circ (\infty, [\lim_n f_n]_\infty) \\
&= (\infty, [\lim_n g_n \circ \lim_n f_n]_\infty) \\
&= (\infty, [\lim_n g_n \circ f_n]_\infty) \\
&= \bigsqcup_n (i_n \wedge j_n, [g_n \circ f_n]_{i_n \wedge j_n}) \\
&= \bigsqcup_n (j_n, [g_n]_{j_n}) \circ (i_n, [f_n]_{i_n}).
\end{aligned}$$

Suppose now that one chain, say the one in $\mathcal{C}((n, A), (m, B))$, has finitely bounded indices but that the other has not. Pick $N \in \mathbb{N}$ such that for any $n \geq N$ we have $(i_N, [f_N]_{i_N}) = (i_{N+n}, [f_{N+n}]_{i_{N+n}})$ and such that $j_N \geq i_N$. We get the following:

$$\begin{aligned}
\bigsqcup_n (j_n, [g_n]_n) \circ \bigsqcup_n (i_n, [f_n]_n) &= (\infty, [\lim_n g_n]_\infty) \circ (i_N, [f_N]_{i_N}) \\
&= (i_N, [\lim_n g_n \circ f_N]_{i_N}) \\
&= (i_N, [g_N \circ f_N]_{i_N}) \\
&= \bigsqcup_n (i_n \wedge j_n, [g_n \circ f_n]_{i_n \wedge j_n}) \\
&= \bigsqcup_n (j_n, [g_n]_{j_n}) \circ (i_n, [f_n]_{i_n}).
\end{aligned}$$

The final case of both chains having finitely bounded indices holds trivially because it is the case that the chain of pairs is constant from some point. □

It is worth noticing that the construction yields not only an O-category, but one in which all hom-sets are pointed and all composition functions are bi-strict. These additional properties are not formal requirements of an O-category, but are necessary for the standard construction of solutions to domain equations.

Next up is building locally continuous functors on the qualification of an M-categories from locally contractive ones on the M-category itself:

Definition 6. Let $\mathcal{D}$ be an M-category and let $G : \mathcal{D}^{op} \times \mathcal{D} \to \mathcal{D}$ be a mixed-variance functor that is locally contractive with factor $\frac{1}{2}$. The *qualification* of G is the mixed variance functor $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$ on the qualification $\mathcal{C}$ of $\mathcal{D}$ that maps objects $((n, A), (m, B))$ to $((n \wedge m) + 1, G(A, B))$ and morphisms $((i, [f]_i), (j, [g]_j))$ to $((i \wedge j) + 1, [G(f, g)]_{(i \wedge j)+1})$.

Proposition 7. The qualification of a locally contractive functor is a well-defined locally continuous functor.

Proof. With the nomenclature of Definition 6 we initially prove that $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$ is well defined. For any two objects $((n, A), (m, B))$ and $((n', A'), (m', B'))$ of $\mathcal{C}^{op} \times \mathcal{C}$ and any morphism $((i, [f]_i), (j, [g]_i))$ between them we know that $i \leq n \wedge n'$ and $j \leq m \wedge m'$ and so we get $(i \wedge j) + 1 \leq (n \wedge m) + 1$ and $(i \wedge j) + 1 \leq (n' \wedge m') + 1$. Also we know that $G(f, g) : G(A, B) \to G(A', B')$ so $((i \wedge j) + 1, [G(f, g)]_{(i \wedge j)+1})$ really is a morphism from $((n \wedge m) + 1, G(A, B))$ to $((n' \wedge m') + 1, G(A', B'))$. To verify representation independence we pick $f' : A' \to B'$ and $g' : B \to B'$ with $f =_i f'$ and $g =_j g'$ and note that since G is locally contractive with factor $\frac{1}{2}$ we get that $G(f, g) =_{(i \wedge j)+1} G(f', g')$. It remains to show that F respects identity and composition. The former is immediate by definition, to prove the latter we take two composable morphisms $((i, [f]_i), (j, [g]_j))$ and $((i', [f']_{i'}), (j', [g']_{j'}))$ and get

$$
\begin{aligned}
&F(((i', [f']_{i'}), (j', [g']_{j'})) \circ ((i, [f]_i), (j, [g]_i))) \\
&\quad = F((i' \wedge i, [f \circ f']_{i \wedge i'}), (j \wedge j', [g' \circ g]_{j \wedge j'})) \\
&\quad = (((i' \wedge i) \wedge (j \wedge j')) + 1, [G(f \circ f', g' \circ g)]_{((i' \wedge i) \wedge (j \wedge j'))+1}) \\
&\quad = (((i \wedge j) + 1) \wedge ((i' \wedge j') + 1), [G(f', g') \circ G(f, g)]_{((i \wedge j)+1) \wedge ((i' \wedge j')+1)}) \\
&\quad = ((i' \wedge j') + 1, [G(f', g')]_{(i' \wedge j')+1}) \circ ((i \wedge j) + 1, [G(f, g)]_{(i \wedge j)+1}) \\
&\quad = F((i', [f']_{i'}), (j', [g']_{j'})) \circ F((i, [f]_i), (j, [g]_i)).
\end{aligned}
$$

To prove that F is locally monotone we take two morphisms $((i, [f]_i), (j, [g]_j))$ and $((i', [f']_{i'}), (j', [g']_{j'}))$ that agree on both domain and codomain and we assume that the former is less than or equal to the latter in the qualification ordering on the hom-set. This implies that $i \leq i'$, $j \leq j'$, $f =_i f'$ and $g =_j g'$ which again means that $(i \wedge j) + 1 \leq (i' \wedge j') + 1$ and that $G(f, g) =_{(i \wedge j)+1} G(f', g')$. But then

$$
\begin{aligned}
F((i, [f]_i), (j, [g]_j)) &= ((i \wedge j) + 1, [G(f, g)]_{(i \wedge j)+1}) \\
&\sqsubseteq ((i' \wedge j') + 1, [G(f', g')]_{(i' \wedge j')+1}) \\
&= F((i', [f']'_i), (j', [g']'_j)).
\end{aligned}
$$

Finally, to prove F locally continuous we take an increasing chain of morphisms

$$((i_0, [f_0]_{i_0}), (j_0, [g_0]_{j_0})) \sqsubseteq ((i_1, [f_1]_{i_1}), (j_1, [g_1]_{j_1})) \sqsubseteq \cdots$$

in a hom-set and we need to prove that the image under F of the least upper bound equals the least upper bound of the image of the chain under F. Consider initially the case where neither of the component chains have finitely bounded

indices, this yields

$$\begin{aligned}
F(\bigsqcup_n((i_n,[f_n]_{i_n}),(j_n,[g_n]_{j_n}))) &= F(\bigsqcup_n(i_n,[f_n]_{i_n}),\bigsqcup_n(j_n,[g_n]_{j_n})) \\
&= F((\infty,[\lim_n f_n]_\infty),(\infty,[\lim_n g_n]_\infty)) \\
&= (\infty,[G(\lim_n f_n,\lim_n g_n)]_\infty) \\
&= (\infty,[\lim_n G(f_n,g_n)]_\infty) \\
&= \bigsqcup_n((i_n \wedge j_n)+1,[G(f_n,g_n)]_{(i_n\wedge j_n)+1}) \\
&= \bigsqcup_n G((i_n,[f_n]_{i_n}),(j_n,[g_n]_{j_n})).
\end{aligned}$$

Suppose now that one of the component chains, say the first, has finitely bounded indices but that the other does not. Much as in the proof of Proposition 5 we pick $N \in \mathbb{N}$ such that for any $n \geq N$ we have $(i_N,[f_N]_{i_N}) = (i_{N+n},[f_{N+n}]_{i_{N+n}})$ and such that $j_N \geq i_N$. This gives us the following:

$$\begin{aligned}
F(\bigsqcup_n((i_n,[f_n]_{i_n}),(j_n,[g_n]_{j_n}))) &= F(\bigsqcup_n(i_n,[f_n]_{i_n}),\bigsqcup_n(j_n,[g_n]_{j_n})) \\
&= F((i_N,[f_N]_{i_N}),(\infty,[\lim_n g_n]_\infty)) \\
&= (i_N+1,[G(f_N,\lim_n g_n)]_{i_N+1}) \\
&= (i_N+1,[G(f_N,g_N)]_{i_N+1}) \\
&= \bigsqcup_n((i_n \wedge j_n)+1,[G(f_n,g_n)]_{(i_n\wedge j_n)+1}) \\
&= \bigsqcup_n G((i_n,[f_n]_{i_n}),(j_n,[g_n]_{j_n})).
\end{aligned}$$

Finally we observe that if both component chains have finitely bounded indices then the chain is constant from some point and the desired equality is trivially true. □

Proposition 8. A locally contractive functor inherits fixed points from its qualification. Indeed, let $\mathcal{D}$ be an M-category and $\mathcal{C}$ its qualification. Let $G : \mathcal{D}^{op} \times \mathcal{D} \to \mathcal{D}$ be a mixed-variance functor that is locally contractive with factor $\frac{1}{2}$ and let $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$ be its qualification. If now $F((n,A),(n,A))$ is isomorphic to (n,A) in $\mathcal{C}$ then we have that $G(A,A)$ is isomorphic to A in $\mathcal{D}$.

Proof. We have a morphism $(i,[f]_i)$ from $F((n,A),(n,A)) = (n+1,G(A,A))$ to (n,A) and a morphism $(j,[g]_j)$ the other way such that

$$(j,[g]_j) \circ (i,[f]_i) = (i \wedge j,[g \circ f]_{i\wedge j}) = (n+1,[1_{G(A,A)}]_{n+1})$$

and such that

$$(i,[f]_i) \circ (j,[g]_j) = (j \wedge i,[f \circ g]_{j\wedge i}) = (n,[1_A]_n).$$

In particular we have $n+1 = n$ which leaves $n = \infty$ as the only option. But then $g \circ f = 1_{G(A,A)}$ and $f \circ g = 1_A$ and we are done. □

Finally we observe (we omit the proof) that if an M-category has the completeness properties that permits the (symmetric) construction of solutions to recursive metric equations then its qualification has completeness properties that similarly permit the construction of solutions to recursive domain equations:

Proposition 9. Let $\mathcal{D}$ be an M-category and let $\mathcal{C}$ be its qualification. If $\mathcal{D}$ is nonempty and has inverse limits of Cauchy towers then $\mathcal{C}$ has a terminal object and limits of ω^{op}-chains of split epis.