

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

Lars Birkedal, Kristian Støvring, and Jacob Thamsborg

IT University of Copenhagen*

December 12, 2009

Abstract

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 (as needed for parametricity reasoning) 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

*Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark.

with effects, here general references. We discussed this point of view extensively earlier [11].

- 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., [18, 24]), 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 [21] 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 stages. 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 [4] 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 [11]. These pairs, called semantic locations, are needed for modeling reference types in stage 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 [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 need to solve recursive equations of approximately the following form

$$\begin{aligned} \mathcal{W} &= \mathbb{N}_0 \multimap_{fn} \mathcal{T} \\ \mathcal{T} &= \mathcal{W} \rightarrow CURel(V) \end{aligned}$$

even in order to define the space in which types will be modeled. 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 [1, 4, 5, 13, 19]; 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} .¹ This involves interpreting the various type constructors of the language as non-expansive 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).

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 wrt. the untyped semantics of terms.

Limitations. In this article we do not consider operational semantics but focus on presenting the model outlined above. We have earlier shown a computational-adequacy result for a semantics similar to the untyped semantics defined in stage one [11]: we expect that result to carry over to the present setup.

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.” Furthermore, a certain technical requirement on the relations we consider (“uniformity”) seems to be too restrictive. In recent work we have shown that both these problems can be overcome: one can use the techniques presented here to construct a more advanced (and more complicated) model that validates sophisticated equivalences in the style of Ahmed et al. [3]. This work will be described elsewhere. 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

¹We remark that the fixed point could also be found using the technique of Pitts [23]; 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 and negative arguments in recursive definitions of relations, and hence we define the meaning of recursive types via Banach’s fixed point theorem [4, 5].

we present the untyped semantics, corresponding to stage one in the outline above. In Section 4 we present the typed semantics, corresponding to the last two stages. In Section 5 we present a few examples of reasoning using the model. Related work is discussed in Section 6.

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 [22]. 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.

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 $Loc = \mathbb{N}_0 \times \bar{\omega}$ where $\bar{\omega}$ is the “vertical natural numbers” cpo $Loc = \mathbb{N}_0 \times \bar{\omega}$ where $\bar{\omega}$ is the “vertical natural numbers” cpo: $1 \sqsubset 2 \sqsubset \dots \sqsubset n \sqsubset \dots \sqsubset \infty$. (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. It is essential for the construction of the typed semantics (in the next section) that these “approximate locations” (l, n) are included.

3.1 Domain-theoretic preliminaries

We assume that the reader is familiar with basic denotational semantics, as presented for example in Winskel [31], and with semantics in monadic style [20]. 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 [23] should be useful, but is not assumed.

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

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

Typing rules:

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

$$\frac{\Xi \mid \Gamma \vdash t_0 : \text{int} \quad \Xi \mid \Gamma \vdash t_1 : \tau \quad \Xi \mid \Gamma \vdash t_2 : \tau}{\Xi \mid \Gamma \vdash \text{ifz } t_0 t_1 t_2 : \tau}$$

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

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

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

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

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

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

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

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

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

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

Figure 1: Programming language

Let \mathbf{Cpo} be the category of ω -cpo's and ω -continuous functions. We use the standard notation for products, sums, and function spaces in \mathbf{Cpo} . Injections into binary sums are written ι_1 and ι_2 . For any set M and any cpo A , the cpo $M \rightarrow_{fn} A$ 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 (cpo) 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 $f : D \rightarrow D$ from a cppo D to itself is written $fix f$. 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 \mathbf{pCpo} be the Kleisli category for the lifting monad: objects are cpo's, while morphisms from A to B are continuous functions from A to B_\perp . The identity maps in \mathbf{pCpo} are written \bar{id} ; they are given by lifting: $\bar{id} = \lambda a. \lfloor a \rfloor$. Composition in \mathbf{pCpo} is written $\bar{\circ}$:

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

The semantics below is presented in monadic style [20], 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 cpo S and every cppo Ans , the continuation-and-state monad $T_{S,Ans} : \mathbf{Cpo} \rightarrow \mathbf{Cpo}$ over S and Ans is given by

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

where $\eta_A : A \rightarrow T_{S,Ans} A$ and $\star_{A,B} : T_{S,Ans} A \rightarrow (A \rightarrow T_{S,Ans} B) \rightarrow T_{S,Ans} B$. 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:

$$\eta a \star f = f a \tag{1}$$

$$c \star \eta = c \tag{2}$$

$$(c \star f) \star g = c \star (\lambda a. f a \star g). \tag{3}$$

(Continuations are included for a technical reason, namely to ensure chain-completeness of the relations that will be used to model computations.)

3.2 Uniform cpos

The standard methods for solving recursive domain equations give solutions that satisfy certain induction principles [23, 29]. 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

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \dots \sqsubseteq \varpi_n \sqsubseteq \dots \quad (4)$$

$$\bigsqcup_{n \in \omega} \varpi_n = \overline{id_A} = \lambda a. [a] \quad (5)$$

$$\varpi_m \bar{\circ} \varpi_n = \varpi_n \bar{\circ} \varpi_m = \varpi_{\min(m,n)} \quad (6)$$

$$\varpi_0 = \lambda e. \perp. \quad (7)$$

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

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \dots \sqsubseteq \varpi_n \sqsubseteq \dots \quad (8)$$

$$\bigsqcup_{n \in \omega} \varpi_n = id_D \quad (9)$$

$$\varpi_m \circ \varpi_n = \varpi_n \circ \varpi_m = \varpi_{\min(m,n)} \quad (10)$$

$$\varpi_0 = \lambda e. \perp. \quad (11)$$

Remark.

1. The projection functions ϖ_n will not be required to have finite range, cf. Abadi and Plotkin [1]. With this requirement, a uniform cppo would just be an SFP-domain together with a particular choice of projection functions. In future work we plan to investigate practical consequences of enforcing the finite-range requirement; this leads to working with metric spaces that are compact (and not merely complete) [1].
2. Uniform cpos are exactly the algebras for a certain monad on the category of cpos and strict, continuous functions. The monad is given by an obvious monoid structure on $\bar{\omega}$: $T_u D = \bar{\omega} \otimes D$, $\eta_u e = (\infty, e)$, $\mu_u(m, (n, e)) = (\min(m, n), e)$. Our locations are modeled using a free algebra for this monad: $Loc_\perp \cong \bar{\omega} \otimes (\mathbb{N}_0)_\perp$.

Uniform cpos are called *rank-ordered cpos* in earlier work by Baier and Majster-Cederbaum [7]. Uniform cpos and uniform cpos are two instances of a general construction on O -categories: see Birkedal et al. [9, Section 8].

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 functions π_n will be 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.

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.

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 \mathbf{Cpo} :

$$V \cong \mathbb{Z} + Loc + 1 + (V \times V) + (V + V) + V + T_{S,Ans}V + (V \rightarrow T_{S,Ans}V) \quad (12)$$

where

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

and

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

2. Abbreviate $TV = T_{S,Ans}V$ and $K = V \rightarrow S \rightarrow Ans$. Define the following injection functions corresponding to the summands on the right-hand side of the isomorphism (12):

$$\begin{array}{ll} in_{\mathbb{Z}} : \mathbb{Z} \rightarrow V & in_+ : V + V \rightarrow V \\ in_{Loc} : Loc \rightarrow V & in_{\rightarrow} : (V \rightarrow TV) \rightarrow V \\ in_1 : 1 \rightarrow V & in_{\mu} : V \rightarrow V \\ in_{\times} : V \times V \rightarrow V & in_{\forall} : TV \rightarrow V \end{array}$$

With that notation, the functions $\pi_n : V \rightarrow V_{\perp}$ satisfy (and are determined by) the equations shown in Figure 2.

$$\pi_0 = \lambda v. \perp \quad (13)$$

$$\pi_{n+1}(in_{\mathbb{Z}}(m)) = [in_{\mathbb{Z}}(m)] \quad (14)$$

$$\pi_{n+1}(in_1(*)) = [in_1(*)] \quad (15)$$

$$\pi_{n+1}(in_{Loc}(l, \infty)) = [in_{Loc}(l, n+1)] \quad (16)$$

$$\pi_{n+1}(in_{Loc}(l, m)) = [in_{Loc}(l, \min(n+1, m))] \quad (17)$$

$$\pi_{n+1}(in_{\times}(v_1, v_2)) = \begin{cases} [in_{\times}(v'_1, v'_2)] & \text{if } \pi_n v_1 = [v'_1] \text{ and } \pi_n v_2 = [v'_2] \\ \perp & \text{otherwise} \end{cases} \quad (18)$$

$$\pi_{n+1}(in_{+}(l_i v)) = \begin{cases} [in_{+}(l_i v')] & \text{if } \pi_n v = [v'] \\ \perp & \text{otherwise} \end{cases} \quad (i = 1, 2) \quad (19)$$

$$\pi_{n+1}(in_{\mu} v) = \begin{cases} [in_{\mu} v'] & \text{if } \pi_n v = [v'] \\ \perp & \text{otherwise} \end{cases} \quad (20)$$

$$\pi_{n+1}(in_{\forall} c) = [in_{\forall}(\pi_{n+1}^T c)] \quad (21)$$

$$\pi_{n+1}(in_{\rightarrow} f) = \left[in_{\rightarrow} \left(\lambda v. \begin{cases} \pi_{n+1}^T(f v') & \text{if } \pi_n v = [v'] \\ \perp & \text{otherwise} \end{cases} \right) \right] \quad (22)$$

Here the functions $\pi_n^S : S \rightarrow S_{\perp}$ and $\pi_n^K : K \rightarrow K$ and $\pi_n^T : TV \rightarrow TV$ are defined as follows:

$$\pi_0^S = \lambda s. \perp \quad \pi_0^K = \lambda k. \perp \quad \pi_0^T = \lambda c. \perp \quad (23)$$

$$\pi_{n+1}^S(s) = \begin{cases} [s'] & \text{if } \pi_n \circ s = \lambda l. [s'(l)] \\ \perp & \text{otherwise} \end{cases} \quad (24)$$

$$\pi_{n+1}^K(k) = \lambda v. \lambda s. \begin{cases} k v' s' & \text{if } \pi_n v = [v'] \text{ and } \pi_{n+1}^S s = [s'] \\ \perp & \text{otherwise} \end{cases} \quad (25)$$

$$\pi_{n+1}^T(c) = \lambda k. \lambda s. \begin{cases} c(\pi_{n+1}^K k) s' & \text{if } \pi_{n+1}^S s = s' \\ \perp & \text{otherwise.} \end{cases} \quad (26)$$

Figure 2: Characterization of the projection functions $\pi_n : V \rightarrow V_{\perp}$.

These two properties determine V uniquely, up to isomorphism in \mathbf{Cpo} .

Proof (sketch). One solves the predomain equation (12) as usual [29]; 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 $\sqcup_n \pi_n = \bar{id}$ are easy to show. To show that $\sqcup_n \pi_n = \bar{id}$, one first shows by induction on m that $\pi_n \circ \varpi_m = \varpi_m \circ \pi_n$ for all n , and that

$$(\sqcup_n \pi_n) \circ \varpi_m = \varpi_m.$$

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

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, $TV = (V \rightarrow S \rightarrow Ans) \rightarrow S \rightarrow Ans$. 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”:

$$\begin{aligned} error_{Ans} &= [\iota_2^*] \\ error &= \lambda k. \lambda s. error_{Ans}. \end{aligned}$$

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 cpos.

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 3.

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

Notice that it would not suffice to define, e.g., $lookup(\lambda_l^{n+1})(k)(s) = \perp$ for $l \in \text{dom}(s)$, 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 $\text{FV}(t) \subseteq X$. The *untyped semantics of t with respect to X* is the continuous function $\llbracket t \rrbracket_X : V^X \rightarrow TV$ defined by induction on t in Figure 4.

$alloc : V \rightarrow TV, \quad lookup : V \rightarrow TV, \quad assign : V \rightarrow V \rightarrow TV.$

$$\begin{aligned}
alloc\ v &= \lambda k\ \lambda s. k\ (\lambda_{free(s)})\ (s[free(s) \mapsto v]) \\
&\quad (\text{where } free(s) = \min\{n \in \mathbb{N}_0 \mid n \notin \text{dom}(s)\}) \\
lookup\ v &= \lambda k\ \lambda s. \left\{ \begin{array}{ll} k\ s(l)\ s & \text{if } v = \lambda_l \text{ and } l \in \text{dom}(s) \\ k\ v'\ s & \text{if } v = \lambda_l^{n+1}, l \in \text{dom}(s), \\ & \text{and } \pi_n(s(l)) = [v'] \\ \perp_{Ans} & \text{if } v = \lambda_l^{n+1}, l \in \text{dom}(s), \\ & \text{and } \pi_n(s(l)) = \perp \\ error_{Ans} & \text{otherwise} \end{array} \right. \\
assign\ v_1\ v_2 &= \lambda k\ \lambda s. \left\{ \begin{array}{ll} k\ (in_1^*)\ (s[l \mapsto v_2]) & \text{if } v_1 = \lambda_l \text{ and } l \in \text{dom}(s) \\ k\ (in_1^*)\ (s[l \mapsto v_2']) & \text{if } v_1 = \lambda_l^{n+1}, l \in \text{dom}(s), \\ & \text{and } \pi_n(v_2) = [v_2'] \\ \perp_{Ans} & \text{if } v_1 = \lambda_l^{n+1}, l \in \text{dom}(s), \\ & \text{and } \pi_n(v_2) = \perp \\ error_{Ans} & \text{otherwise} \end{array} \right.
\end{aligned}$$

Figure 3: Functions used for interpreting reference operations.

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 $\llbracket t \rrbracket^P$ of Ans defined by

$$\llbracket t \rrbracket^P = \llbracket t \rrbracket_{\emptyset} \emptyset k_{init} s_{init}$$

where

$$k_{init} = \lambda v. \lambda s. \left\{ \begin{array}{ll} [v_1\ m] & \text{if } v = in_Z(m) \\ error_{Ans} & \text{otherwise} \end{array} \right.$$

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

We emphasize that even though the above semantics is slightly non-standard because of the use of the projection functions in lookup and assignment, we can still use it to reason about operational behaviour: as mentioned in the introduction an earlier adequacy proof [11] should carry over to the present setting.

For every t with $\text{FV}(t) \subseteq X$, define the continuous $\llbracket t \rrbracket_X : V^X \rightarrow TV$ by induction on t :

$$\begin{aligned}
\llbracket x \rrbracket_X \rho &= \eta(\rho(x)) \\
\llbracket \bar{m} \rrbracket_X \rho &= \eta(\text{in}_{\mathbb{Z}} m) \\
\llbracket \text{ifz } t_0 \ t_1 \ t_2 \rrbracket_X \rho &= \llbracket t_0 \rrbracket_X \rho \star \lambda v_0. \begin{cases} \llbracket t_1 \rrbracket_X \rho & \text{if } v_0 = \text{in}_{\mathbb{Z}} 0 \\ \llbracket t_2 \rrbracket_X \rho & \text{if } v_0 = \text{in}_{\mathbb{Z}} m \text{ where } m \neq 0 \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket t_1 \pm t_2 \rrbracket_X \rho &= \llbracket t_1 \rrbracket_X \rho \star \lambda v_1. \llbracket t_2 \rrbracket_X \rho \star \lambda v_2. \begin{cases} \eta(\text{in}_{\mathbb{Z}}(m_1 \pm m_2)) & \\ \text{if } v_1 = \text{in}_{\mathbb{Z}} m_1 & \\ \text{and } v_2 = \text{in}_{\mathbb{Z}} m_2 & \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket () \rrbracket_X \rho &= \eta(\text{in}_1 *) \\
\llbracket (t_1, t_2) \rrbracket_X \rho &= \llbracket t_1 \rrbracket_X \rho \star \lambda v_1. \llbracket t_2 \rrbracket_X \rho \star \lambda v_2. \eta(\text{in}_{\times}(v_1, v_2)) \\
\llbracket \text{fst } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \begin{cases} \eta(v_1) & \text{if } v = \text{in}_{\times}(v_1, v_2) \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket \text{snd } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \begin{cases} \eta(v_2) & \text{if } v = \text{in}_{\times}(v_1, v_2) \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket \text{void } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \text{error} \\
\llbracket \text{inl } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \eta(\text{in}_+(l_1 v)) \\
\llbracket \text{inr } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \eta(\text{in}_+(l_2 v)) \\
\llbracket \text{case } t_0 \ x_1.t_1 \ x_2.t_2 \rrbracket_X \rho &= \llbracket t_0 \rrbracket_X \rho \star \lambda v_0. \begin{cases} \llbracket t_1 \rrbracket_{X,x_1}(\rho[x_1 \mapsto v]) & \text{if } v_0 = \text{in}_+(l_1 v) \\ \llbracket t_2 \rrbracket_{X,x_2}(\rho[x_2 \mapsto v]) & \text{if } v_0 = \text{in}_+(l_2 v) \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket \lambda x. t \rrbracket_X \rho &= \eta(\text{in}_{\rightarrow}(\lambda v. \llbracket t \rrbracket_{X,x}(\rho[x \mapsto v]))) \\
\llbracket t_1 \ t_2 \rrbracket_X \rho &= \llbracket t_1 \rrbracket_X \rho \star \lambda v_1. \llbracket t_2 \rrbracket_X \rho \star \lambda v_2. \begin{cases} g \ v_2 & \text{if } v_1 = \text{in}_{\rightarrow} g \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket \text{fix } f. \lambda x. t \rrbracket_X \rho &= \eta(\text{in}_{\rightarrow}(\text{fix}(\lambda g^{V \rightarrow TV}. \lambda v. \llbracket t \rrbracket_{X,f,x}(\rho[f \mapsto \text{in}_{\rightarrow} g, x \mapsto v]))) \\
\llbracket \text{fold } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \eta(\text{in}_{\mu} v) \\
\llbracket \text{unfold } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \begin{cases} \eta(v_0) & \text{if } v = \text{in}_{\mu} v_0 \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket \Lambda \alpha. t \rrbracket_X \rho &= \eta(\text{in}_{\forall}(\llbracket t \rrbracket_X \rho)) \\
\llbracket t[\tau] \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \begin{cases} c & \text{if } v = \text{in}_{\forall} c \\ \text{error} & \text{otherwise} \end{cases} \\
\llbracket \text{ref } t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \text{alloc } v \\
\llbracket !t \rrbracket_X \rho &= \llbracket t \rrbracket_X \rho \star \lambda v. \text{lookup } v \\
\llbracket t_1 := t_2 \rrbracket_X \rho &= \llbracket t_1 \rrbracket_X \rho \star \lambda v_1. \llbracket t_2 \rrbracket_X \rho \star \lambda v_2. \text{assign } v_1 \ v_2
\end{aligned}$$

Figure 4: Untyped semantics of terms.

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 [28], 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 $d : X \times X \rightarrow \mathbb{R}^+$ satisfying the following three conditions:

- (i) $d(x, y) = 0 \iff x = y$
- (ii) $d(x, y) = d(y, x)$
- (iii) $d(x, z) \leq d(x, y) + d(y, z)$.

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}^ω 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}^ω *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}^ω 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 $f : X_1 \rightarrow X_2$ 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 $f : X_1 \rightarrow X_2$ 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 [30]. 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 $(X_1, d_1) \rightarrow (X_2, d_2)$ 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 $d_{X_1 \rightarrow X_2}$ as distance function:

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

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 [30].

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 [4]:

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, $d(fix f, fix g) \leq d(f, g)$.

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 $x = \text{fix } f$ and $y = \text{fix } g$. 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 $d(x, y) \leq d(f, g)$. \square

4.1.2 Solving recursive metric-space equations

The inverse-limit method for solving recursive domain equations can be adapted from Cpo to CBUlt [6, 25]. For a unified account, see Wagner [30]; here we sketch a less general variant which 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 $F : \text{CBUlt}^{\text{op}} \times \text{CBUlt} \rightarrow \text{CBUlt}$ 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 $F : \text{CBUlt}^{\text{op}} \times \text{CBUlt} \rightarrow \text{CBUlt}$ 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 [6]:

Proposition 4.9.

Let $0 < c < 1$.

1. Let $(X, d) \in \text{CBUlt}$, and define $c \cdot (X, d) = (X, c \cdot d)$ where $c \cdot d : X \times X \rightarrow \mathbb{R}^+$ is given by $(c \cdot d)(x, y) = c \cdot d(x, y)$. We have $c \cdot (X, d) \in \text{CBUlt}$.
2. Let $F : \text{CBUlt}^{\text{op}} \times \text{CBUlt} \rightarrow \text{CBUlt}$ 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 \mathbf{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.

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

Proof (sketch). By a well-known adaptation of the inverse-limit method [6, 25, 30]. For a detailed proof of a more general theorem, see Birkedal et al. [9]. \square

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 $\text{Rel}(A)$ be the set of binary relations $R \subseteq A \times A$ on A .

1. A relation $R \in \text{Rel}(A)$ 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 $\text{CRel}(A)$ be the set of complete relations on A .
2. A relation $R \in \text{Rel}(D)$ on a cppo D is *pointed* if $(\perp, \perp) \in R$ and *admissible* if it is pointed and complete. Let $\text{ARel}(D)$ be the set of admissible relations on D .
3. For every cpo A and every relation $R \in \text{Rel}(A)$, define the relation $R_{\perp} \in \text{Rel}(A_{\perp})$ by $R_{\perp} = \{(\perp, \perp)\} \cup \{([a], [a']) \mid (a, a') \in R\}$.
4. For $R \in \text{Rel}(A)$ and $S \in \text{Rel}(B)$, let $R \rightarrow S$ 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 [1, 4]. 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 $R \in \text{Rel}(A)$ is *uniform with respect to* $(\varpi_n)_{n \in \omega}$ if $\varpi_n \in R \rightarrow R_\perp$ for all n . Let $\text{CUREl}(A, (\varpi_n)_{n \in \omega})$ 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 $R \in \text{Rel}(D)$ is *uniform with respect to* $(\varpi_n)_{n \in \omega}$ if $\varpi_n \in R \rightarrow R$ for all n . Let $\text{AUREl}(D, (\varpi_n)_{n \in \omega})$ 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 let $R, S \in \text{AUREl}(D, (\varpi_n)_{n \in \omega})$.

1. If $\varpi_n \in R \rightarrow S$, then $\varpi_{n'} \in R \rightarrow S$ for all $n' \leq n$.
2. If $\varpi_n \in R \rightarrow S$ 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 [4], we obtain:

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

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

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 $\varpi_n \in R \rightarrow S$ and $\varpi_n \in S \rightarrow R$. Assume that $R \neq S$. By (11) we always have $\varpi_0 \in R \rightarrow S$ and $\varpi_0 \in S \rightarrow R$, 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:

$$d(R, S) \leq 2^{-n} \text{ if and only if } \varpi_n \in R \rightarrow S \text{ and } \varpi_n \in S \rightarrow R. \quad (27)$$

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 $m, m' \geq M_n$, (27) then implies that $\varpi_n \in R_m \rightarrow R_{m'}$. Therefore, for all $e, e' \in D$,

$$\begin{aligned} (\varpi_n e, \varpi_n e') \in R_m &\implies ((\varpi_n \circ \varpi_n) e, (\varpi_n \circ \varpi_n) e') \in R_{m'} \quad (\text{by definition of } d) \\ &\implies (\varpi_n e, \varpi_n e') \in R_{m'}, \quad (\text{by (10)}) \end{aligned}$$

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 (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 (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 $m \geq M_n$,

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

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 (10) and the definition of R that $(\varpi_n e, \varpi_n e') \in R$. \square

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

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

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

Definition 4.16. For every $(X, d) \in \text{CBUlt}$, define an “extension” ordering \leq on $\mathbb{N}_0 \rightarrow_{fn} X$ by

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

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

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

be the set of functions ν from $\mathbb{N}_0 \rightarrow_{fin} X$ to $\mathbf{AURel}(D, (\varpi_n)_{n \in \omega})$ 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

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

Proof. The set $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} \mathbf{AURel}(D, (\varpi_n)_{n \in \omega})$ is a subset of the underlying set of the exponential $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow \mathbf{AURel}(D, (\varpi_n)_{n \in \omega})$ in \mathbf{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 $(\mathbb{N}_0 \rightarrow_{fin} X)$ to $\mathbf{AURel}(D, (\varpi_n)_{n \in \omega})$ 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 $\mathbb{N}_0 \rightarrow_{fin} X$ such that $\Delta \leq \Delta'$; we must show that $\nu(\Delta) \subseteq \nu(\Delta')$. By Proposition 4.13(2) it suffices to show that $\varpi_n \in \nu(\Delta) \rightarrow \nu(\Delta')$ 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 $\varpi_n \in \nu(\Delta) \rightarrow \nu_m(\Delta)$ by Proposition 4.13(1). But ν_m is assumed to be monotone, so $\nu_m(\Delta) \subseteq \nu_m(\Delta')$ and therefore $\varpi_n \in \nu(\Delta) \rightarrow \nu_m(\Delta')$. Since also $d(\nu(\Delta'), \nu_m(\Delta')) \leq 2^{-n}$, we have $\varpi_n \in \nu_m(\Delta') \rightarrow \nu(\Delta')$, and conclude by (10) that $\varpi_n \in \nu(\Delta) \rightarrow \nu(\Delta')$. \square

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 $\mathbf{CUREl}(A) = \mathbf{CUREl}(A, (\varpi_n)_{n \in \omega})$.

1. The set $\mathbf{CUREl}(A)$ is a complete, 1-bounded ultrametric space with the distance function given by

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

2. Let $(X, d) \in \mathbf{CBUlt}$, and let $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} \mathbf{CUREl}(A)$ be the set of functions ν from $\mathbb{N}_0 \rightarrow_{fin} X$ to $\mathbf{CUREl}(A)$ 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

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

Proof.

1: It is easy to see that the family of strict extensions $\varpi_n^\dagger : A_\perp \rightarrow A_\perp$ of the projection functions $\varpi_n : A \rightarrow A_\perp$ turns $(A_\perp, (\varpi_n^\dagger)_{n \in \omega})$ into a uniform cppo. Abbreviate $AURel(A_\perp) = AURel(A_\perp, (\varpi_n^\dagger)_{n \in \omega})$. By definition of uniform relations,

$$R \in CURel(A) \quad \text{if and only if} \quad R_\perp \in AURel(A_\perp) \quad (28)$$

for all R in $Rel(A)$. Furthermore, Proposition 4.14 gives a metric on the set $AURel(A_\perp)$, and it is easy to see that the distance function on $CURel(A)$ 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 $CURel(A)$ into a 1-bounded ultrametric space.

However, not every S in $AURel(A_\perp)$ has the form R_\perp for some R in $CURel(A)$: unless A is empty, some relations in $AURel(A_\perp)$ relate \perp to elements different from \perp . In other words, the lifting operator from $CURel(A)$ to $AURel(A_\perp)$ is not surjective. Therefore, completeness of $AURel(A_\perp)$ does not immediately imply completeness of $CURel(A)$. What we need to show is that the subset of $AURel(A_\perp)$ 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 $AURel(A_\perp)$. Proposition 4.5 then implies that the subset of strict relations is a complete metric space, and (28) implies that it is isomorphic to $CURel(A)$, which is therefore also complete.

More generally, let $(D, (\varpi'_n)_{n \in \omega})$ be a uniform cppo, and abbreviate $AURel(D) = AURel(D, (\varpi'_n)_{n \in \omega})$; we show that the subset $SAURel(D) \subseteq AURel(D)$ of strict relations is closed. So let $(R_m)_{m \in \omega}$ be a sequence of strict relations (elements of $SAURel(D)$) with limit R for some $R \in AURel(D)$. 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 (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 $\varpi'_n \in R \rightarrow R_m$ 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 $CURel(A)$ is isomorphic to the complete, 1-bounded metric space $SAURel(A_\perp)$ 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 $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} SAURel(A_\perp)$ of non-expansive and monotone functions from $\mathbb{N}_0 \rightarrow_{fin} X$ to $SAURel(A_\perp)$ is a complete metric space with the “sup” metric on functions:

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

By Proposition 4.5 it is enough to show that $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} SAURel(A_\perp)$ is a closed subset of $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} AURel(A_\perp)$. 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. \square

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 $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} CURel(V)$ (as given by the previous proposition) can be extended to a locally non-expansive functor $F : CBUlt^{op} \rightarrow CBUlt$ in the natural way:

$$\begin{aligned} F(X, d) &= (\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} CURel(V) \\ F(f) &= \lambda\nu. \lambda\Delta. \nu(f \circ \Delta) \end{aligned}$$

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 $(\mathbb{N}_0 \rightarrow_{fin} X_1) \rightarrow_{mon} CURel(V)$ to the set of functions from $(\mathbb{N}_0 \rightarrow_{fin} X_2)$ 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 $\Delta, \Delta' \in (\mathbb{N}_0 \rightarrow_{fin} X_2)$ 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 $(\mathbb{N}_0 \rightarrow_{fin} X_1) \rightarrow_{mon} CURel(V)$, and all Δ and Δ' in $(\mathbb{N}_0 \rightarrow_{fin} X_2)$,

$$d(F(f)(\nu)(\Delta), F(f')(\nu')(\Delta')) \leq \max(d(f, f'), d(\nu, \nu'), d(\Delta, \Delta')). \quad (29)$$

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 $(\mathbb{N}_0 \rightarrow_{fin} X_2)$ 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 (29).

Now, for all f and ν , taking $f' = f$ and $\nu' = \nu$ in (29) shows that $F(f)(\nu)$ is non-expansive. Similarly, taking $f' = f$ and $\Delta' = \Delta$ in (29) 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:

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

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

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

for all parallel morphisms (non-expansive functions) f and f' . But that follows from (29) by taking $\nu' = \nu$ and $\Delta' = \Delta$. \square

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

$$\widehat{\mathcal{T}} \cong \frac{1}{2}((\mathbb{N}_0 \rightarrow_{fin} \widehat{\mathcal{T}}) \rightarrow_{mon} CURel(V)) \quad (30)$$

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

$$\widehat{\mathcal{T}} \rightleftarrows ((\mathbb{N}_0 \rightarrow_{fin} \widehat{\mathcal{T}}) \rightarrow_{mon} CURel(V)).$$

We implicitly use that bijection below. Notice that the function space $(\mathbb{N}_0 \rightarrow_{fin} \widehat{\mathcal{T}}) \rightarrow_{mon} CURel(V)$ 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 (30) 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 (30), and let $App : \widehat{\mathcal{T}} \rightarrow \frac{1}{2}((\mathbb{N}_0 \rightarrow_{fin} \widehat{\mathcal{T}}) \rightarrow_{mon} CURel(V))$ be an isomorphism with inverse $Lam : \frac{1}{2}((\mathbb{N}_0 \rightarrow_{fin} \widehat{\mathcal{T}}) \rightarrow_{mon} CURel(V)) \rightarrow \widehat{\mathcal{T}}$. For convenience, we use the following abbreviations (where the names \mathcal{W} and \mathcal{T} are intended to indicate “worlds” and “types”, respectively):

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

With that notation, (30) 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 $K = V \rightarrow S \rightarrow Ans$), 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

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

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

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

be the complete, uniform 1-bounded ultrametric space obtained from Propositions 4.15 and 4.18 and the exponential in \mathbf{CBUit} . (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 $x \stackrel{n}{=}_d y$ means that $d(x, y) \leq 2^{-n}$. When the distance function d is clear from the context, we shall just write $x \stackrel{n}{=} y$ for $x \stackrel{n}{=}_d y$.

(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 $\stackrel{n}{=}_d$ 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 $f, f' : (D_1, d_1) \rightarrow (D_2, d_2)$ and elements $x, x' \in D_1$,

$$f \stackrel{n}{=} f' \wedge x \stackrel{n}{=} x' \implies f(x) \stackrel{n}{=} f'(x'). \quad (31)$$

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 5. Notice that the operator ref is defined in terms of n -approximated equality $\stackrel{n}{=}$ on $CURel(V)$, 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 (30) implies that worlds that are “ $(n + 1)$ -equal” only contain “ n -equal” semantic types.

Lemma 4.25. The function $states$ from \mathcal{W} to $Rel(S)$ defined in the lower part of Figure 5 is an element of \mathcal{T}_S .

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

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

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

For every $\Xi \vdash \tau$, define the non-expansive $\llbracket \tau \rrbracket_{\Xi} : \mathcal{T}^{\Xi} \rightarrow \mathcal{T}$ by induction on τ :

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

The following operators and elements are used above:

$$\begin{array}{ll}
\times : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T} & \text{comp} : \mathcal{T} \rightarrow \mathcal{T}_T \\
+ : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T} & \text{cont} : \mathcal{T} \rightarrow \mathcal{T}_K \\
\text{ref} : \mathcal{T} \rightarrow \mathcal{T} & \text{states} \in \mathcal{T}_S \\
\rightarrow : \mathcal{T} \times \mathcal{T}_T \rightarrow \mathcal{T} & R_{Ans} \in CRel(Ans)
\end{array}$$

$$(\nu_1 \times \nu_2)(\Delta) = \{ (in_{\times}(v_1, v_2), in_{\times}(v'_1, v'_2)) \mid (v_1, v'_1) \in \nu_1(\Delta) \wedge (v_2, v'_2) \in \nu_2(\Delta) \}$$

$$\begin{aligned}
(\nu_1 + \nu_2)(\Delta) &= \{ (in_{+}(\iota_1 v_1), in_{+}(\iota_1 v'_1)) \mid (v_1, v'_1) \in \nu_1(\Delta) \} \cup \\
&\quad \{ (in_{+}(\iota_2 v_2), in_{+}(\iota_2 v'_2)) \mid (v_2, v'_2) \in \nu_2(\Delta) \}
\end{aligned}$$

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

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

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

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

$$\begin{aligned}
\text{states}(\Delta) &= \{ (s, s') \mid \text{dom}(s) = \text{dom}(s') = \text{dom}(\Delta) \\
&\quad \wedge \forall l \in \text{dom}(\Delta). (s(l), s'(l)) \in \text{App}(\Delta(l))(\Delta) \}
\end{aligned}$$

$$R_{Ans} = \{ (\perp, \perp) \} \cup \{ ([\iota_1 m], [\iota_1 m]) \mid m \in \mathbb{Z} \}$$

Figure 5: Interpretation of types.

Lemma A.2(1) and symmetry. So, let $\Delta \stackrel{n}{=} \Delta'$ and let $(s, s') \in \text{states}(\Delta)$; 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 $(s_0, s'_0) \in \text{states}(\Delta')$. If $n = 0$ we are done by (23); assume therefore that $n > 0$. Then $\text{dom}(\Delta) = \text{dom}(\Delta')$ by the definition of the metric on \mathcal{W} , and furthermore, for every $l \in \text{dom}(\Delta)$,

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

By transitivity (Proposition 4.24),

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

and therefore Lemma A.2(1) gives that

$$\pi_{n-1} \in \text{App}(\Delta(l))(\Delta) \rightarrow (\text{App}(\Delta'(l))(\Delta'))_{\perp}.$$

Since the above holds for every $l \in \text{dom}(\Delta)$, Equation (24) gives that either $\pi_n^S(s) = \pi_n^S(s') = \perp$, 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 $(s_0(l), s'_0(l)) \in \text{App}(\Delta'(l))(\Delta')$ for all $l \in \text{dom}(\Delta')$. But the latter means exactly that $(s_0, s'_0) \in \text{states}(\Delta')$. \square

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 $\text{dom}(\Delta) = \text{dom}(\Delta')$ by definition of the metric on \mathcal{W} . Now define $\Delta'_1 \in \mathcal{W}$ with $\text{dom}(\Delta'_1) = \text{dom}(\Delta_1)$ by

$$\Delta'_1(l) = \begin{cases} \Delta'(l) & \text{if } l \in \text{dom}(\Delta) \\ \Delta_1(l) & \text{if } l \in \text{dom}(\Delta_1) \setminus \text{dom}(\Delta). \end{cases}$$

Clearly $\Delta' \leq \Delta'_1$ since $\text{dom}(\Delta) = \text{dom}(\Delta')$. Also, by definition of the metric on \mathcal{W} (as a maximum of the distances for each “ l ”), $d(\Delta_1, \Delta'_1) = d(\Delta, \Delta') \leq 2^{-n}$. \square

Lemma 4.27. The operators \times , $+$, *ref*, \rightarrow , *cont*, and *comp* defined in the lower part of Figure 5 are non-expansive.

Proof. We show that each operator maps into the appropriate codomain and that it is non-expansive.

$\times : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$:

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 $\mathbb{N}_0 \rightarrow_{fn} \widehat{\mathcal{T}}$,

$$\nu_1 \stackrel{n}{=} \nu'_1 \wedge \nu_2 \stackrel{n}{=} \nu'_2 \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in (\nu_1 \times \nu_2)(\Delta) \rightarrow (\nu'_1 \times \nu'_2)(\Delta)_\perp.$$

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

$$(in_\times(v_1, v_2), in_\times(v'_1, v'_2)) \in (\nu_1 \times \nu_2)(\Delta).$$

We must show that either (1) $\pi_n(in_\times(v_1, v_2)) = \pi_n(in_\times(v'_1, v'_2)) = \perp$ or (2) $\pi_n(in_\times(v_1, v_2)) = [w]$ and $\pi_n(in_\times(v'_1, v'_2)) = [w']$ for some w and w' such that $(w, w') \in (\nu'_1 \times \nu'_2)(\Delta')$. If $n = 0$ we are done by Equation (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, (31) 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 $\pi_{n-1} \in \nu_1(\Delta) \rightarrow \nu'_1(\Delta')_\perp$ and $\pi_{n-1} \in \nu_2(\Delta) \rightarrow \nu'_2(\Delta')_\perp$ by Lemma A.2(1). By definition of $\nu_i(\Delta) \rightarrow \nu'_i(\Delta')_\perp$ (for $i = 1, 2$) there are now two cases:

1. $\pi_{n-1}(v_1) = \pi_{n-1}(v'_1) = \perp$ or $\pi_{n-1}(v_2) = \pi_{n-1}(v'_2) = \perp$.
2. There exist $(w_1, w'_1) \in \nu'_1(\Delta')$ and $(w_2, w'_2) \in \nu'_2(\Delta')$ where $\pi_{n-1}(v_1) = [w_1]$ and $\pi_{n-1}(v'_1) = [w'_1]$ and $\pi_{n-1}(v_2) = [w_2]$ and $\pi_{n-1}(v'_2) = [w'_2]$.

In case (1), (18) gives that $\pi_n(in_\times(v_1, v_2)) = \pi_n(in_\times(v'_1, v'_2)) = \perp$ and we are done. In case (2), (18) gives that $\pi_n(in_\times(v_1, v_2)) = [in_\times(w_1, w_2)]$ and $\pi_n(in_\times(v'_1, v'_2)) = [in_\times(w'_1, w'_2)]$. 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 : $\mathcal{T} \rightarrow \mathcal{T}$:

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

$$\nu \stackrel{n}{=} \nu' \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in ref(\nu)(\Delta) \rightarrow ref(\nu')(\Delta')_\perp$$

for all ν and ν' in \mathcal{T} and all Δ and Δ' in $\mathbb{N}_0 \rightarrow_{fn} \widehat{\mathcal{T}}$.

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 completely similar, but slightly easier.) If

$n = 0$ we are done by Equation (4). If $n > 0$, (17) gives that $\pi_n(\lambda_l^m) = \lfloor \lambda_l^{\min(n,m)} \rfloor$, and it therefore remains to show that $(\lambda_l^{\min(n,m)}, \lambda_l^{\min(n,m)}) \in \text{ref}(\nu')(\Delta')$. To that end, let $l \in \text{dom}(\Delta')$ and $\Delta'_1 \geq \Delta'$; we must show that $\text{App}(\Delta'(l))(\Delta'_1) \stackrel{\min(n,m)-1}{=} \nu'(\Delta'_1)$. Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta'_1$. Then:

$$\begin{aligned}
\text{App}(\Delta'(l))(\Delta'_1) &\stackrel{n}{=} \text{App}(\Delta'(l))(\Delta_1) && (\text{App}(\Delta'(l)) \text{ non-expansive}) \\
&\stackrel{n-1}{=} \text{App}(\Delta(l))(\Delta_1) && (\text{Lemma A.1}) \\
&\stackrel{m-1}{=} \nu(\Delta_1) && (\text{since } (\lambda_l^m, \lambda_l^m) \in \text{ref}(\nu)(\Delta)) \\
&\stackrel{n}{=} \nu'(\Delta_1) && (\text{Lemma A.2(2)}) \\
&\stackrel{n}{=} \nu'(\Delta'_1). && (\nu' \text{ non-expansive})
\end{aligned}$$

Hence by transitivity $\text{App}(\Delta'(l))(\Delta'_1) \stackrel{\min(n,m)-1}{=} \nu'(\Delta'_1)$.

$+$: $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$:

It is easy to see that $(\nu_1 + \nu_2)(\Delta)$ is complete for all $\Delta \in \mathcal{W}$, and that $\nu_1 + \nu_2$ is monotone. It then suffices to show that

$$\nu_1 \stackrel{n}{=} \nu'_1 \wedge \nu_2 \stackrel{n}{=} \nu'_2 \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in (\nu_1 + \nu_2)(\Delta) \rightarrow (\nu'_1 + \nu'_2)(\Delta')_{\perp}$$

for all ν_1, ν_2, ν'_1 , and ν'_2 in \mathcal{T} and all Δ and Δ' in $\mathbb{N}_0 \rightarrow_{\text{fin}} \widehat{\mathcal{T}}$.

So, assume that $\nu_1 \stackrel{n}{=} \nu'_1$ and $\nu_2 \stackrel{n}{=} \nu'_2$ and $\Delta \stackrel{n}{=} \Delta'$, and let

$$(\text{in}_+(\iota_1 v), \text{in}_+(\iota_1 v')) \in (\nu_1 + \nu_2)(\Delta).$$

(The case with ι_2 instead of ι_1 is completely symmetric.) If $n = 0$ we are done by Equation (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 $\pi_{n-1}(v) = \pi_{n-1}(v') = \perp$, 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 (19) gives that $\pi_n(\text{in}_+(\iota_1 v)) = \lfloor \text{in}_+(\iota_1 w) \rfloor$ and $\pi_n(\text{in}_+(\iota_1 v')) = \lfloor \text{in}_+(\iota_1 w') \rfloor$ with $(\text{in}_+(\iota_1 w), \text{in}_+(\iota_1 w')) \in (\nu'_1 + \nu'_2)(\Delta')$.

$\text{cont} : \mathcal{T} \rightarrow \mathcal{T}_K$:

First, $\text{cont}(\nu)(\Delta)$ is admissible for each $\Delta \in \mathcal{W}$ since R_{Ans} is admissible. Also, $\text{cont}(\nu)$ is monotone. By Lemma A.3(3), it then suffices to show that

$$\nu \stackrel{n}{=} \nu' \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n^K \in \text{cont}(\nu)(\Delta) \rightarrow \text{cont}(\nu')(\Delta')$$

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 $(k, k') \in \text{cont}(\nu)(\Delta)$; we must show that $(\pi_n^K(k), \pi_n^K(k')) \in \text{cont}(\nu')(\Delta')$. If $n = 0$ this follows from (23) and the fact that $\text{cont}(\nu')(\Delta')$

is pointed. Otherwise, let $\Delta'_1 \geq \Delta'$ and $(v, v') \in \nu'(\Delta'_1)$ and $(s, s') \in \text{states}(\Delta'_1)$. 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 (31), $\nu(\Delta_1) \stackrel{n}{=} \nu'(\Delta'_1)$. Furthermore, the fact that states belongs to \mathcal{T}_S , shown above, implies that $\text{states}(\Delta_1) \stackrel{n}{=} \text{states}(\Delta'_1)$. Therefore, by (25), either $\pi_n^K(k) v s = \pi_n^K(k') v' s' = \perp$, 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 $\pi_n(s) = \lfloor s_0 \rfloor$ and $\pi_n(s') = \lfloor s'_0 \rfloor$ with $(w, w') \in \nu(\Delta_1)$ and $(s_0, s'_0) \in \text{states}(\Delta_1)$. In the latter case, $(k w s_0, k' w' s'_0) \in R_{Ans}$ since $(k, k') \in \text{cont}(\nu)(\Delta)$ and $\Delta \leq \Delta_1$.

comp : $\mathcal{T} \rightarrow \mathcal{T}_T$:

Completely similar to *cont*. First, *comp*(ν)(Δ) is admissible for each $\Delta \in \mathcal{W}$ since R_{Ans} is admissible. Also, *comp*(ν) is monotone. By Lemma A.3(3), it then suffices to show that

$$\nu \stackrel{n}{=} \nu' \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n^T \in \text{comp}(\nu)(\Delta) \rightarrow \text{comp}(\nu')(\Delta')$$

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 $(c, c') \in \text{comp}(\nu)(\Delta)$; we must show that $(\pi_n^T(c), \pi_n^T(c')) \in \text{comp}(\nu')(\Delta')$. If $n = 0$ this follows from (23) and the fact that *comp*(ν')(Δ') is pointed. Otherwise, let $\Delta'_1 \geq \Delta'$ and $(k, k') \in \text{cont}(\nu')(\Delta'_1)$ and $(s, s') \in \text{states}(\Delta'_1)$. 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,

$$\text{cont}(\nu)(\Delta_1) \stackrel{n}{=} \text{cont}(\nu')(\Delta_1) \stackrel{n}{=} \text{cont}(\nu')(\Delta'_1).$$

Furthermore, the fact that states belongs to \mathcal{T}_S implies that $\text{states}(\Delta_1) \stackrel{n}{=} \text{states}(\Delta'_1)$. Therefore, by (26), either $\pi_n^T(c) k s = \pi_n^T(c') k' s' = \perp$, 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 $(\pi_n^K(k), \pi_n^K(k')) \in \text{cont}(\nu)(\Delta_1)$ and $(s_0, s'_0) \in \text{states}(\Delta_1)$. In the latter case, $(c (\pi_n^K(k)) s_0, c' (\pi_n^K(k')) s'_0) \in R_{Ans}$ since $(c, c') \in \text{comp}(\nu)(\Delta)$ and $\Delta \leq \Delta_1$.

\rightarrow : $\mathcal{T} \times \mathcal{T}_T \rightarrow \mathcal{T}$:

It is easy to see that $(\nu \rightarrow \xi)(\Delta)$ is admissible for all $\Delta \in \mathcal{W}$ since ξ maps worlds to admissible relations. Also, $\nu \rightarrow \xi$ is obviously monotone. By Lemma A.3(3), it then suffices to show that

$$\nu \stackrel{n}{=} \nu' \wedge \xi \stackrel{n}{=} \xi' \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in (\nu \rightarrow \xi)(\Delta) \rightarrow (\nu' \rightarrow \xi')(\Delta')_{\perp}$$

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 $(in_{\rightarrow}, f, in_{\rightarrow}, f') \in (\nu \rightarrow \xi)(\Delta)$. If $n = 0$ we are done by Equation (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 (22), it suffices to show that $(in_{\rightarrow}(g), in_{\rightarrow}(g')) \in (\nu' \rightarrow \xi')(\Delta')$. 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 $(f, f') \in (\nu \rightarrow \xi)(\Delta)$ and $\Delta_1 \geq \Delta$. Then by (31), $\xi(\Delta_1) \stackrel{n}{=} \xi'(\Delta'_1)$, and therefore $(\pi_n^T(f w), \pi_n^T(f' w')) \in \xi'(\Delta'_1)$. But this means exactly that $(g(v), g'(v')) \in \xi'(\Delta'_1)$. \square

It is here, in order to show that ref is well-defined (and non-expansive), that we need the approximate locations λ_i^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 $ref(\nu)(\Delta) = \{(in_{Loc} l, in_{Loc} l) \mid l \in \text{dom}(\Delta) \wedge \dots\}$. 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.² 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}^Ξ 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 $\llbracket \tau \rrbracket_\Xi : \mathcal{T}^\Xi \rightarrow \mathcal{T}$ defined by induction on τ in Figure 5. The interpretation of recursive types is by appeal to Banach’s fixed-point theorem (see Theorem 4.29).

²In particular, the obvious definition of ref as $ref(\nu)(\Delta) = \{(in_{Loc} l, in_{Loc} l) \mid l \in \text{dom}(\Delta) \wedge \forall \Delta_1 \geq \Delta. App(\Delta(l))(\Delta_1) = \nu(\Delta_1)\}$ would *not* be well-defined, since it would not be non-expansive in Δ .

In more detail, well-definedness of $\llbracket \tau \rrbracket_{\Xi}$ 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 $\text{fix } f.\lambda x.t$.

Theorem 4.29. Let τ be a type such that $\Xi \vdash \tau$.

1. The function $\llbracket \tau \rrbracket_{\Xi} : \mathcal{T}^{\Xi} \rightarrow \mathcal{T}$ defined in Figure 5 is non-expansive.
2. If $\Xi = \Xi', \alpha$ then for all $\varphi' \in \mathcal{T}^{\Xi'}$ we have that $\lambda \nu. \lambda \Delta. \{ (in_{\mu} v, in_{\mu} v') \mid (v, v') \in \llbracket \tau \rrbracket_{\Xi', \alpha} \varphi' [\alpha \mapsto \nu] \Delta \}$ is a contractive function from \mathcal{T} to \mathcal{T} . In particular, $\llbracket \mu \alpha. \tau \rrbracket_{\Xi}$ is well-defined.

Proof. First, generalize Part 2 above:

- 2'. $\lambda \varphi \lambda \Delta. \{ (in_{\mu} v, in_{\mu} v') \mid (v, v') \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta \}$ is a contractive function from \mathcal{T}^{Ξ} 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 $\llbracket \tau \rrbracket_{\Xi}$ is a constant function and hence trivially non-expansive. If τ is a type variable α , then non-expansiveness of $\llbracket \tau \rrbracket_{\Xi}$ follows directly from the definition of the product metric. In the cases where τ is $\tau_1 \times \tau_2$, $\tau_1 + \tau_2$, $\text{ref } \tau'$, or $\tau_1 \rightarrow \tau_2$, 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 $\llbracket \mu \alpha. \tau' \rrbracket_{\Xi}$ is a (well-defined) function from \mathcal{T}^{Ξ} to \mathcal{T} . To show that $\llbracket \mu \alpha. \tau' \rrbracket_{\Xi}$ is non-expansive, let $\varphi \stackrel{n}{=} \varphi'$; we must show that $\llbracket \mu \alpha. \tau' \rrbracket_{\Xi} \varphi \stackrel{n}{=} \llbracket \mu \alpha. \tau' \rrbracket_{\Xi} \varphi'$. By Proposition 4.7 it suffices to show that the two contractive functions $g, g' : \mathcal{T} \rightarrow \mathcal{T}$ defined by

$$\begin{aligned} g &= \lambda \nu. \lambda \Delta. \{ (in_{\mu} v, in_{\mu} v') \mid (v, v') \in \llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi [\alpha \mapsto \nu] \Delta \} \\ g' &= \lambda \nu. \lambda \Delta. \{ (in_{\mu} v, in_{\mu} v') \mid (v, v') \in \llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi' [\alpha \mapsto \nu] \Delta \} \end{aligned}$$

satisfy that $g \stackrel{n}{=} g'$. So let $\nu \in \mathcal{T}$ be given; we must show that $g \nu \stackrel{n}{=} g' \nu$. But this follows from 2' and the induction hypothesis. Therefore, $\llbracket \mu \alpha. \tau' \rrbracket_{\Xi}$ is non-expansive.

Now assume that τ is $\forall \alpha. \tau'$ for some τ' such that $\Xi, \alpha \vdash \tau'$. First, $\llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi \Delta$ is complete for all $\Delta \in \mathcal{W}$ since arbitrary intersections of complete relations are complete. It is also easy to see that $\llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi$ is monotone

since $\text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu])$ is monotone for all $\nu \in \mathcal{T}$. By Lemma A.2(3), it then suffices to show that

$$\varphi \stackrel{n}{=} \varphi' \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_n \in \llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi \Delta \rightarrow (\llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi' \Delta')_{\perp}$$

for all φ and φ' in \mathcal{T}^{Ξ} and all Δ and Δ' in \mathcal{W} . So, let $(\text{in}_{\forall} c, \text{in}_{\forall} c') \in \llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi \Delta$. If $n = 0$ we are done by Equation (4); assume therefore that $n > 0$. By (21) it then suffices to show that $(\text{in}_{\forall}(\pi_n^T c), \text{in}_{\forall}(\pi_n^T c')) \in \llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi' \Delta'$. To this end, let $\Delta'_1 \geq \Delta'$ and $\nu \in \mathcal{T}$; we must show that $(\pi_n^T c, \pi_n^T c') \in \text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi'[\alpha \mapsto \nu])\Delta'_1$. Lemma 4.26 gives a $\Delta_1 \geq \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta'_1$. Then $(c, c') \in \text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu])\Delta_1$ since $(\text{in}_{\forall} c, \text{in}_{\forall} c') \in \llbracket \forall \alpha. \tau' \rrbracket_{\Xi} \varphi \Delta$. By the induction hypothesis, $\llbracket \tau' \rrbracket_{\Xi, \alpha}$ is non-expansive, and therefore

$$\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu] \stackrel{n}{=} \llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi'[\alpha \mapsto \nu]$$

by the definition of the product metric. The operator comp is non-expansive by Lemma 4.27, and therefore

$$\text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu]) \stackrel{n}{=} \text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi'[\alpha \mapsto \nu]).$$

Finally, by (31),

$$\text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu])\Delta_1 \stackrel{n}{=} \text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi'[\alpha \mapsto \nu])\Delta'_1,$$

and we conclude that

$$(\pi_n^T c, \pi_n^T c') \in \text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi'[\alpha \mapsto \nu])\Delta'_1$$

by Lemma A.3(1) and the fact that $(c, c') \in \text{comp}(\llbracket \tau' \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu])\Delta_1$.

2': Let $G = \lambda \varphi \lambda \Delta. \{ (\text{in}_{\mu} v, \text{in}_{\mu} v') \mid (v, v') \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta \}$; we must show that G is a contractive function from \mathcal{T}^{Ξ} 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}^{Ξ} and all Δ and Δ' in \mathcal{W} ,

$$\varphi \stackrel{n}{=} \varphi' \wedge \Delta \stackrel{n}{=} \Delta' \implies \pi_{n+1} \in G(\varphi)(\Delta) \rightarrow G(\varphi')(\Delta')_{\perp}.$$

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 $(\text{in}_{\mu} v, \text{in}_{\mu} v') \in G(\varphi)(\Delta)$. We know that $(v, v') \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta$ by definition of G . Part 1 gives that $\llbracket \tau \rrbracket_{\Xi}$ is non-expansive, and therefore $\llbracket \tau \rrbracket_{\Xi} \varphi \stackrel{n}{=} \llbracket \tau \rrbracket_{\Xi} \varphi'$. By (31), $\llbracket \tau \rrbracket_{\Xi} \varphi \Delta \stackrel{n}{=} \llbracket \tau \rrbracket_{\Xi} \varphi' \Delta'$,

and there are therefore two cases: either $\pi_n v = \pi_n v' = \perp$, in which case we are done by (20), or there exists $(w, w') \in \llbracket \tau \rrbracket_{\Xi} \varphi' \Delta'$ such that $\pi_n v = \lfloor w \rfloor$ and $\pi_n v' = \lfloor w' \rfloor$. But in the latter case, (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 0 , belongs to \mathcal{T} . \square

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}^Ξ and $\nu \in \mathcal{T}$,

$$\llbracket \tau \rrbracket_{\Xi} \varphi = \llbracket \tau \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu].$$

2. Let τ and τ' be types such that $\Xi, \alpha \vdash \tau$ and $\Xi \vdash \tau'$. For all φ in \mathcal{T}^Ξ ,

$$\llbracket \tau[\tau'/\alpha] \rrbracket_{\Xi} \varphi = \llbracket \tau \rrbracket_{\Xi, \alpha} (\varphi[\alpha \mapsto \llbracket \tau' \rrbracket_{\Xi} \varphi]).$$

Corollary 4.31. For $\Xi, \alpha \vdash \tau$ and $\varphi \in \mathcal{T}^\Xi$,

$$\llbracket \mu \alpha. \tau \rrbracket_{\Xi} \varphi = \lambda \Delta. \{ (in_\mu v, in_\mu v') \mid (v, v') \in \llbracket \tau[\mu \alpha. \tau/\alpha] \rrbracket_{\Xi} \varphi \Delta \}.$$

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 of the monad T .

Definition 4.32. For $\nu \in \mathcal{T}$ and $\xi \in \mathcal{T}_T$ and $\Delta \in \mathcal{W}$, let $\nu \xrightarrow{\Delta} \xi$ be the binary relation on functions $V \rightarrow TV$ defined by

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

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 $(\eta v, \eta v') \in \text{comp}(\nu)(\Delta)$.
2. If $(c, c') \in \text{comp}(\nu_1)(\Delta)$ and $(f, f') \in \nu_1 \xrightarrow{\Delta} \text{comp}(\nu_2)$, then

$$(c \star f, c' \star f') \in \text{comp}(\nu_2)(\Delta).$$

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 $(\eta v, \eta v') \in \text{comp}(\nu)(\Delta)$, let $\Delta_1 \geq \Delta$ and $(k, k') \in \text{cont}(\nu)(\Delta_1)$ and $(s, s') \in \text{states}(\Delta_1)$; 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 $\text{cont}(\nu)(\Delta_1)$ since $(v, v') \in \nu(\Delta) \subseteq \nu(\Delta_1)$ by monotonicity.

2: Assume that $(c, c') \in \text{comp}(\nu_1)(\Delta)$ and $(f, f') \in \nu_1 \xrightarrow{\Delta} \text{comp}(\nu_2)$. 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 $(c \star f, c' \star f') \in \text{comp}(\nu_2)(\Delta)$, let $\Delta_1 \geq \Delta$ and $(k, k') \in \text{cont}(\nu_2)(\Delta_1)$ and $(s, s') \in \text{states}(\Delta_1)$; 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 $(c, c') \in \text{comp}(\nu_1)(\Delta)$ and $\Delta_1 \geq \Delta$ and $(s, s') \in \text{states}(\Delta_1)$, it suffices to show that $(\lambda v. \lambda s_1. f v k s_1, \lambda v'. \lambda s'_1. f' v' k' s'_1) \in \text{cont}(\nu_1)(\Delta_1)$. So, let $\Delta_2 \geq \Delta_1$ and $(v, v') \in \nu_1(\Delta_2)$ and $(s_1, s'_1) \in \text{states}(\Delta_2)$; we must show that $(f v k s_1, f' v' k' s'_1) \in R_{Ans}$. First, $(f v, f' v') \in \text{comp}(\nu_2)(\Delta_2)$ by assumption on (f, f') . By monotonicity of $\text{cont}(\nu_2)$ we have $(k, k') \in \text{cont}(\nu_2)(\Delta_2)$, and by assumption, $(s_1, s'_1) \in \text{states}(\Delta_2)$. Therefore, it follows from the definition of $\text{cont}(\nu_2)(\Delta_2)$ that $(f v k s_1, f' v' k' s'_1) \in R_{Ans}$. \square

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 $V^{\text{dom}(\Gamma)}$ defined by

$$[[\Gamma]]_{\Xi} \varphi \Delta = \{ (\rho, \rho') \mid \forall x \in \text{dom}(\Gamma). (\rho(x), \rho'(x)) \in [[\Gamma(x)]]_{\Xi} \varphi \Delta \}.$$

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 $(\rho, \rho') \in [[\Gamma]]_{\Xi} \varphi \Delta$,

$$\left([[t]]_{\text{dom}(\Gamma)} \rho, [[t']]_{\text{dom}(\Gamma)} \rho' \right) \in \text{comp}([[\tau]]_{\Xi} \varphi)(\Delta).$$

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 $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi} \varphi \Delta$ be given. Then $(\rho(x), \rho'(x)) \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta$. Therefore, by Proposition 4.33(1),

$$(\llbracket x \rrbracket_{\text{dom}(\Gamma)} \rho, \llbracket x \rrbracket_{\text{dom}(\Gamma)} \rho') = (\eta(\rho(x)), \eta(\rho'(x))) \in \text{comp}(\llbracket \tau \rrbracket_{\Xi} \varphi)(\Delta),$$

as required.

2. If $\Xi \mid \Gamma \models t \sim t' : \mu\alpha.\tau$, then $\Xi \mid \Gamma \models \text{unfold } t \sim \text{unfold } t' : \tau[\mu\alpha.\tau/\alpha]$. Indeed, let $\varphi \in \mathcal{T}^\Xi$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi} \varphi \Delta$ be given. Recall that $\llbracket \text{unfold } t \rrbracket_{\text{dom}(\Gamma)} \rho = \llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho \star f$ and $\llbracket \text{unfold } t' \rrbracket_{\text{dom}(\Gamma)} \rho' = \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho' \star f$ where $f : V \rightarrow TV$ is given by

$$f v = \begin{cases} \eta(v_0) & \text{if } v = \text{in}_\mu v_0 \\ \text{error} & \text{otherwise.} \end{cases}$$

By assumption, $(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho, \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho') \in \text{comp}(\llbracket \mu\alpha.\tau \rrbracket_{\Xi} \varphi)(\Delta)$. Therefore, by Proposition 4.33(2), it suffices to show that

$$(f, f) \in \llbracket \mu\alpha.\tau \rrbracket_{\Xi} \varphi \xrightarrow{\Delta} \text{comp}(\llbracket \tau[\mu\alpha.\tau/\alpha] \rrbracket_{\Xi} \varphi).$$

To see this, let $\Delta_1 \geq \Delta$ and $(v, v') \in \llbracket \mu\alpha.\tau \rrbracket_{\Xi} \varphi \Delta_1$ be given; we must show that $(f v, f v') \in \text{comp}(\llbracket \tau[\mu\alpha.\tau/\alpha] \rrbracket_{\Xi} \varphi)(\Delta_1)$. By Corollary 4.31, $(v, v') = (\text{in}_\mu v_0, \text{in}_\mu v'_0)$ for some $(v_0, v'_0) \in \llbracket \tau[\mu\alpha.\tau/\alpha] \rrbracket_{\Xi} \varphi \Delta_1$. But then by Proposition 4.33(1),

$$(f v, f v') = (\eta(v_0), \eta(v'_0)) \in \text{comp}(\llbracket \tau[\mu\alpha.\tau/\alpha] \rrbracket_{\Xi} \varphi)(\Delta_1),$$

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 $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi} \varphi \Delta$ be given. Recall that $\llbracket \Lambda\alpha.t \rrbracket_{\text{dom}(\Gamma)} \rho = \eta(\text{in}_\forall(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho))$ and that $\llbracket \Lambda\alpha.t' \rrbracket_{\text{dom}(\Gamma)} \rho' = \eta(\text{in}_\forall(\llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho'))$. We must therefore show that

$$\left(\eta(\text{in}_\forall(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho)), \eta(\text{in}_\forall(\llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho')) \right) \in \text{comp}(\llbracket \forall\alpha.\tau \rrbracket_{\Xi} \varphi)(\Delta).$$

By Proposition 4.33(1) it suffices to show that

$$\left(\text{in}_\forall(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho), \text{in}_\forall(\llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho') \right) \in \llbracket \forall\alpha.\tau \rrbracket_{\Xi} \varphi \Delta.$$

We proceed according to the definition of $\llbracket \forall\alpha.\tau \rrbracket_{\Xi}$. Let $\nu \in \mathcal{T}$ be given; we must show that $(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho, \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho') \in \text{comp}(\llbracket \tau \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu])(\Delta)$. But this follows from the assumption that $\Xi, \alpha \mid \Gamma \models t \sim t' : \tau$ since Proposition 4.30(1) (weakening) gives that $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu]$.

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 $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi} \varphi \Delta$

be given. Recall that $\llbracket t[\tau_1] \rrbracket_{\text{dom}(\Gamma)} \rho = \llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho \star g$ and $\llbracket t'[\tau_1] \rrbracket_{\text{dom}(\Gamma)} \rho' = \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho' \star g$ where $g : V \rightarrow TV$ is given by

$$g v = \begin{cases} c & \text{if } v = \text{in}_{\forall} c \\ \text{error} & \text{otherwise.} \end{cases}$$

By assumption, $(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho, \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho') \in \text{comp}(\llbracket \forall \alpha. \tau \rrbracket_{\Xi} \varphi)(\Delta)$. Therefore, by Proposition 4.33(2), it suffices to show that

$$(g, g) \in \llbracket \forall \alpha. \tau \rrbracket_{\Xi} \varphi \stackrel{\Delta}{\rightarrow} \text{comp}(\llbracket \tau_0[\tau_1/\alpha] \rrbracket_{\Xi} \varphi).$$

To see this, let $\Delta_1 \geq \Delta$ and $(v, v') \in \llbracket \forall \alpha. \tau \rrbracket_{\Xi} \varphi \Delta_1$ be given; we must show that $(g v, g v') \in \text{comp}(\llbracket \tau_0[\tau_1/\alpha] \rrbracket_{\Xi} \varphi)(\Delta_1)$. By the definition of $\llbracket \forall \alpha. \tau_0 \rrbracket_{\Xi}$ we know that $(v, v') = (\text{in}_{\forall} c, \text{in}_{\forall} c')$ for some c and c' satisfying that $(c, c') \in \text{comp}(\llbracket \tau_0 \rrbracket_{\Xi, \alpha} \varphi[\alpha \mapsto \nu])(\Delta_1)$ for all $\nu \in \mathcal{T}$. Now choose $\nu = \llbracket \tau_1 \rrbracket_{\Xi} \varphi$: Proposition 4.30(2) (substitution) gives that

$$(g v, g v') = (c, c') \in \text{comp}(\llbracket \tau_0[\tau_1/\alpha] \rrbracket_{\Xi} \varphi)(\Delta_1),$$

as required.

5. If $\Xi \mid \Gamma \models t \sim t' : \tau$, then $\Xi \mid \Gamma \models \text{ref } t \sim \text{ref } t' : \text{ref } \tau$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi} \varphi \Delta$ be given. Recall that $\llbracket \text{ref } t \rrbracket_{\text{dom}(\Gamma)} \rho = \llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho \star \lambda v. \text{alloc } v$ and $\llbracket \text{ref } t' \rrbracket_{\text{dom}(\Gamma)} \rho' = \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho' \star \lambda v. \text{alloc } v$. By assumption, $(\llbracket t \rrbracket_{\text{dom}(\Gamma)} \rho, \llbracket t' \rrbracket_{\text{dom}(\Gamma)} \rho') \in \text{comp}(\llbracket \tau \rrbracket_{\Xi} \varphi)(\Delta)$. Therefore, by Proposition 4.33(2), it suffices to show that

$$(\text{alloc}, \text{alloc}) \in \llbracket \tau \rrbracket_{\Xi} \varphi \stackrel{\Delta}{\rightarrow} \text{comp}(\llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi).$$

To see this, let $\Delta_1 \geq \Delta$ and $(v, v') \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta_1$ be given; we must show that $(\text{alloc } v, \text{alloc } v') \in \text{comp}(\llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi)(\Delta_1)$. We proceed according to the definition of comp . Let $\Delta_2 \geq \Delta_1$ and $(k, k') \in \text{cont}(\llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi)(\Delta_2)$ and $(s, s') \in \text{states}(\Delta_2)$ be given; we must show that

$$(\text{alloc } v k s, \text{alloc } v' k' s') \in R_{Ans}. \quad (32)$$

We know that $\text{dom}(s) = \text{dom}(s') = \text{dom}(\Delta_2)$. Let $l_0 \in \mathbb{N}_0$ be the least number such that $l_0 \notin \text{dom}(\Delta_2)$; then

$$\text{alloc } v k s = k \lambda_{l_0} (s[l_0 \mapsto v]) \quad (33)$$

$$\text{alloc } v' k' s' = k' \lambda_{l_0} (s'[l_0 \mapsto v']). \quad (34)$$

We now aim to use the assumption that $(k, k') \in \text{cont}(\llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi)(\Delta_2)$. Define $\Delta_3 = \Delta_2[l_0 \mapsto \text{Lam}(\llbracket \tau \rrbracket_{\Xi} \varphi)]$ (where $\text{Lam} = \text{App}^{-1}$ is the isomorphism associated with the recursive metric-space equation.) Clearly $\Delta_3 \geq \Delta_2$ since $l_0 \notin \text{dom}(\Delta_2)$. Then $(\lambda_{l_0}, \lambda_{l_0}) \in \llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi \Delta_3 = \text{ref}(\llbracket \tau \rrbracket_{\Xi} \varphi)(\Delta_3)$ since for all $\Delta_4 \geq \Delta_3$ we have

$$\text{App}(\Delta_4(l_0)) = \text{App}(\Delta_3(l_0)) = \text{App}(\text{Lam}(\llbracket \tau \rrbracket_{\Xi} \varphi)) = \llbracket \tau \rrbracket_{\Xi} \varphi.$$

Furthermore, $(s[l_0 \mapsto v], s'[l_0 \mapsto v']) \in \text{states}(\Delta_3)$: for $l \in \text{dom}(\Delta_2)$ we have

$$\begin{aligned} ((s[l_0 \mapsto v])(l), (s'[l_0 \mapsto v'])(l)) &= (s(l), s'(l)) \in \text{App}(\Delta_2(l))(\Delta_2) \\ &= \text{App}(\Delta_3(l))(\Delta_2) \\ &\subseteq \text{App}(\Delta_3(l))(\Delta_3), \end{aligned}$$

by monotonicity of $\text{App}(\Delta_3(l)) \in \mathcal{W} \rightarrow_{\text{mon}} \text{CUREl}(V)$, and also,

$$\begin{aligned} ((s[l_0 \mapsto v])(l_0), (s'[l_0 \mapsto v'])(l_0)) &= (v, v') \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta_1 \\ &\subseteq \llbracket \tau \rrbracket_{\Xi} \varphi \Delta_3, \end{aligned}$$

by monotonicity of $\llbracket \tau \rrbracket_{\Xi} \varphi$. All in all, $(s[l_0 \mapsto v], s'[l_0 \mapsto v']) \in \text{states}(\Delta_3)$. Therefore, (33), (34), and the assumption that $(k, k') \in \text{cont}(\llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi)(\Delta_2)$ gives (32), as required.

6. If $\Xi \mid \Gamma \models t \sim t' : \text{ref } \tau$, then $\Xi \mid \Gamma \models !t \sim !t' : \tau$. Indeed, let $\varphi \in \mathcal{T}^{\Xi}$ and $\Delta \in \mathcal{W}$ and $(\rho, \rho') \in \llbracket \Gamma \rrbracket_{\Xi} \varphi \Delta$ be given. Using exactly the same reasoning as in the previous case, we see that it suffices to show that

$$(\text{lookup}, \text{lookup}) \in \llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi \xrightarrow{\Delta} \text{comp}(\llbracket \tau \rrbracket_{\Xi} \varphi).$$

Therefore, let $\Delta_1 \geq \Delta$ and $(v, v') \in \llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi \Delta_1$ be given; we must show that $(\text{lookup } v, \text{lookup } v') \in \text{comp}(\llbracket \tau \rrbracket_{\Xi} \varphi)(\Delta_1)$. According to the definition of $\llbracket \text{ref } \tau \rrbracket_{\Xi} \varphi = \text{ref}(\llbracket \tau \rrbracket_{\Xi} \varphi)$ there are two cases: either $v = v' = \lambda_l$ for some $l \in \text{dom}(\Delta_1)$, or $v = v' = \lambda_l^{n+1}$ for some $n \in \omega$ and $l \in \text{dom}(\Delta_1)$. Assume that we are in the latter case; the former case is completely similar, but easier.

We proceed according to the definition of *comp*. Let $\Delta_2 \geq \Delta_1$ and $(k, k') \in \text{cont}(\llbracket \tau \rrbracket_{\Xi} \varphi)(\Delta_2)$ and $(s, s') \in \text{states}(\Delta_2)$ be given; we must show that

$$(\text{lookup } v \ k \ s, \text{lookup } v' \ k' \ s') \in R_{\text{Ans}}. \quad (35)$$

By the definition of *ref* we have $v = v' = \lambda_l^{n+1}$ where $\text{App}(\Delta_1(l))(\Delta_2) \stackrel{n}{=} \llbracket \tau \rrbracket_{\Xi} \varphi \Delta_2$. Since $(s, s') \in \text{states}(\Delta_2)$ we know that

$$(s(l), s'(l)) \in \text{App}(\Delta_2(l))(\Delta_2).$$

Since $l \in \text{dom}(\Delta_1)$ and $\Delta_2 \geq \Delta_1$, we have

$$\text{App}(\Delta_2(l))(\Delta_2) = \text{App}(\Delta_1(l))(\Delta_2) \stackrel{n}{=} \llbracket \tau \rrbracket_{\Xi} \varphi \Delta_2.$$

There are therefore two cases: either $\pi_n(s(l)) = \pi_n(s'(l)) = \perp$, or else $(\pi_n(s(l)), \pi_n(s'(l))) = ([v_0], [v'_0])$ where $(v_0, v'_0) \in \llbracket \tau \rrbracket_{\Xi} \varphi \Delta_2$. In the first case, the definition of *lookup* gives

$$(\text{lookup } v \ k \ s, \text{lookup } v' \ k' \ s') = (\perp, \perp) \in R_{\text{Ans}}.$$

and we are done. In the second case, the definition of *lookup* gives

$$(\text{lookup } v \ k \ s, \text{lookup } v' \ k' \ s') = (k \ v_0 \ s, k' \ v'_0 \ s'). \quad (36)$$

By assumption, $(k, k') \in \text{cont}(\llbracket \tau \rrbracket_{\Xi} \varphi)(\Delta_2)$. Therefore, by definition of *cont*, we have $(k \ v_0 \ s, k' \ v'_0 \ s') \in R_{Ans}$. From (36) we then conclude (35), and we are done. \square

Corollary 4.37.

1. If $\emptyset \mid \emptyset \vdash t : \tau$ is a closed term of type τ , then $\llbracket t \rrbracket_{\emptyset} \emptyset \neq \text{error}$.
2. If $\emptyset \mid \emptyset \vdash t : \text{int}$ is a closed term of type *int*, then $\llbracket t \rrbracket^{\text{P}} \neq \text{error}_{Ans}$.

Proof. 1. The theorem gives that $(\llbracket t \rrbracket_{\emptyset} \emptyset, \llbracket t \rrbracket_{\emptyset} \emptyset) \in \text{comp}(\llbracket \tau \rrbracket_{\emptyset} \emptyset)(\emptyset)$. 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 $(k_0, k_0) \in \text{cont}(\llbracket \tau \rrbracket_{\emptyset} \emptyset)(\emptyset)$ and that $(s_{init}, s_{init}) \in \text{states}(\emptyset)$. 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 $\llbracket t \rrbracket_{\emptyset} \emptyset \ k_0 \ s_{init} \neq \text{error}_{Ans}$ which implies that $\llbracket t \rrbracket_{\emptyset} \emptyset \neq \text{error}$.

2. Recall that $\llbracket t \rrbracket^{\text{P}} = \llbracket t \rrbracket_{\emptyset} \emptyset \ k_{init} \ s_{init}$ where $s_{init} \in S$ is the empty store and where

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

We claim that $(k_{init}, k_{init}) \in \text{cont}(\llbracket \text{int} \rrbracket_{\emptyset} \emptyset)(\emptyset)$. Indeed, let $\Delta \in \mathcal{W}$ and $(v, v') \in (\llbracket \text{int} \rrbracket_{\emptyset} \emptyset)(\Delta)$ and $(s, s') \in \text{states}(\Delta)$ be given. Then $v = v' = \text{in}_{\mathbb{Z}}(m)$ for some $m \in \mathbb{Z}$, and therefore

$$(k_{init} \ v \ s, k_{init} \ v' \ s') = (\lfloor \iota_1 \ m \rfloor, \lfloor \iota_1 \ m \rfloor) \in R_{Ans},$$

as required. Furthermore, as already argued, $(s_{init}, s_{init}) \in \text{states}(\emptyset)$.

The theorem gives that $(\llbracket t \rrbracket_{\emptyset} \emptyset, \llbracket t \rrbracket_{\emptyset} \emptyset) \in \text{comp}(\llbracket \tau \rrbracket_{\emptyset} \emptyset)(\emptyset)$, which then implies that

$$(\llbracket t \rrbracket^{\text{P}}, \llbracket t \rrbracket^{\text{P}}) = (\llbracket t \rrbracket_{\emptyset} \emptyset \ k_{init} \ s_{init}, \llbracket t \rrbracket_{\emptyset} \emptyset \ k_{init} \ s_{init}) \in R_{Ans}.$$

Therefore $\llbracket t \rrbracket^{\text{P}} \neq \text{error}_{Ans}$ by definition of R_{Ans} . \square

5 Examples

The model can be used to prove the equivalences in Section 5 of our earlier work [11]. 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 6

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.

Example 5.1. We use the usual encoding of existential types by means of universal types [14]: $\exists\alpha.\tau = \forall\beta.(\forall\alpha.\tau \rightarrow \beta) \rightarrow \beta$. The type

$$\tau_m = \exists\alpha.(1 \rightarrow \alpha) \times (\alpha \rightarrow 1) \times (\alpha \rightarrow \mathbf{int})$$

can then 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 : $J = \Lambda\beta.\lambda c.c[\mathbf{ref\ int}]I$ and $J' = \Lambda\beta.\lambda c.c[\mathbf{ref\ int}]I'$ where

$$\begin{aligned} I &= (\lambda x.\mathbf{ref}(0), \lambda x.x := !x + 1, \lambda x.!x) \\ I' &= (\lambda x.\mathbf{ref}(0), \lambda x.x := !x - 1, \lambda x.-(!x)). \end{aligned}$$

By parametricity reasoning, i.e., by exploiting the universal quantification in the interpretation of universal types, one can show that $\emptyset \mid \emptyset \models J \sim J' : \tau_m$.

Example 5.2. One can alternatively implement an imperative counter module by means of a local reference and two closures. Consider the type $\tau_r = 1 \rightarrow ((1 \rightarrow 1) \times (1 \rightarrow \mathbf{int}))$ and the two counter implementations

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

where the $\mathbf{let\ } \dots \mathbf{in}$ construct is syntactic sugar for a β -redex in the usual way. Both M and M' are closed terms of type τ_r . By “store parametricity” reasoning, i.e., by exploiting the universal quantification over all larger worlds in the definition of *cont*, one can show that $\emptyset \mid \emptyset \models M \sim M' : \tau_r$.

Example 5.3. Consider the two terms $K = \lambda x.2$ and $K' = \lambda x.3$ of type $\mathbf{ref\ } 0 \rightarrow \mathbf{int}$. 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, the equivalence $\emptyset \mid \emptyset \models K \sim K' : \mathbf{ref\ } 0 \rightarrow \mathbf{int}$ does not hold. Briefly, the reason is the existence of approximated locations in the model.

6 Related Work

As already mentioned, the metric-space structure on uniform relations over universal domains is well-known [1, 4, 5, 13, 19]. The inverse-limit method

for solving recursive domain equations was first adapted to metric spaces by America and Rutten [6]; see also Rutten [25]. For a unified account covering both domains and metric spaces, see Wagner [30].

Semantic (or “approximated”) locations were first introduced in our earlier work [11]. That work contains an adequacy proof with respect to an operational semantics and 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.

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 [26], and in a more recent extension [27] (with F. Pottier) that includes an “anti-frame rule” for local reasoning about state. Furthermore, the technique has been used by the authors, in ongoing work [10], to construct an operational model of the logic of Schwinghammer et al. [26]. In other ongoing work, the two first authors and A. Buisse have used the technique to construct an operational model of concurrent separation logic for a language that allows locks to be stored in the heap (in the style of Gotsman et al. [17]).

The fundamental circularity between worlds and types in realizability-style possible-worlds models of polymorphism and general references was observed by Ahmed [2, 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.³

Ahmed et al. [3] 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. In future work we plan to investigate potential advantages of our

³In future work we plan to perform a more formal comparison.

approach as compared to the one of Ahmed et al. [3]: one advantage could be the removal of “approximation information” in definitions and equivalence proofs.⁴ We also plan to investigate local-state parameters in the style of Bohr and Birkedal [12]. 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.

A Concrete descriptions of some metric spaces

Recall that the set $\mathcal{T} = \mathcal{W} \rightarrow_{\text{mon}} \text{CUREl}(V)$ is a subset of the underlying set of the exponential $\mathcal{T} = \mathcal{W} \rightarrow \text{CUREl}(V)$ 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 $\text{App} : \widehat{\mathcal{T}} \rightarrow 1/2 \cdot \mathcal{T}$ obtained from Theorem 4.20. The fact that App is an isomorphism then implies that

$$d_2(\widehat{v}, \widehat{v}') = 1/2 \cdot d_1(\text{App } \widehat{v}, \text{App } \widehat{v}') \quad (37)$$

for all \widehat{v} and \widehat{v}' in $\widehat{\mathcal{T}}$.

Lemma A.1. For $\Delta, \Delta' \in \mathcal{W}$, we have that $\Delta \stackrel{n}{=} \Delta'$ if and only if either (1) $n = 0$, or (2) $n > 0$ and $\text{dom}(\Delta) = \text{dom}(\Delta')$ and

$$\text{App}(\Delta(l))\Delta_0 \stackrel{n-1}{=} \text{App}(\Delta'(l))\Delta_0$$

for all $l \in \text{dom}(\Delta)$ and all $\Delta_0 \in \mathcal{W}$.

(The “ $n - 1$ ” comes from the “ $1/2$ ” on the right-hand side of the isomorphism (30).)

Proof. “Only if”: Assume that $\Delta \stackrel{n}{=} \Delta'$ and that $n > 0$; we must show that $\text{dom}(\Delta) = \text{dom}(\Delta')$ and that $\text{App}(\Delta(l))\Delta_0 \stackrel{n-1}{=} \text{App}(\Delta'(l))\Delta_0$ for all $l \in \text{dom}(\Delta)$ 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 $\text{dom}(\Delta) = \text{dom}(\Delta')$ and that $d(\Delta, \Delta') = \max\{d_2(\Delta(l), \Delta'(l)) \mid l \in \text{dom}(\Delta)\}$. Now let $l \in \text{dom}(\Delta)$ and $\Delta_0 \in \mathcal{W}$. First, $d_2(\Delta(l), \Delta'(l)) \leq d(\Delta, \Delta') \leq 2^{-n}$, and (37) therefore gives that

$$d_1(\text{App}(\Delta(l)), \text{App}(\Delta'(l))) = 2 \cdot d_2(\Delta(l), \Delta'(l)) \leq 2^{-(n-1)}.$$

Then by definition of the metric d_1 on \mathcal{T} (Proposition 4.18),

$$d(\text{App}(\Delta(l))\Delta_0, \text{App}(\Delta'(l))(\Delta_0)) \leq d_1(\text{App}(\Delta(l)), \text{App}(\Delta'(l))) \leq 2^{-(n-1)},$$

i.e., $\text{App}(\Delta(l))\Delta_0 \stackrel{n-1}{=} \text{App}(\Delta'(l))(\Delta_0)$, and we are done.

⁴For a different approach to relational reasoning without such approximation information, see other recent work [15, 16].

“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 $\text{dom}(\Delta) = \text{dom}(\Delta')$, and that $\text{App}(\Delta(l))\Delta_0 \stackrel{n-1}{=} \text{App}(\Delta'(l))\Delta_0$ for all $l \in \text{dom}(\Delta)$ and all $\Delta_0 \in \mathcal{W}$. Then for every $l \in \text{dom}(\Delta)$ we have $d_1(\text{App}(\Delta(l)), \text{App}(\Delta'(l))) \leq 2^{-(n-1)}$ by definition of d_1 , and hence

$$d_2(\Delta(l), \Delta'(l)) = 1/2 \cdot d_1(\text{App} \Delta(l), \text{App} \Delta'(l)) \leq 2^{-n}$$

by (37). Therefore, $d(\Delta, \Delta') \leq 2^{-n}$, i.e., $\Delta \stackrel{n}{=} \Delta'$. \square

Lemma A.2. Let $(A, (\varpi_n)_{n \in \omega})$ be a uniform cpo. Abbreviate $CURel(A) = CURel(A, (\varpi_n)_{n \in \omega})$ and consider the metrics on $CURel(A)$ and $\mathcal{W} \rightarrow_{mon} CURel(A)$ given by Proposition 4.18.

1. For $R, S \in CURel(A)$, we have that $R \stackrel{n}{=} S$ if and only if $\varpi_n \in R \rightarrow S_\perp$ and $\varpi_n \in S \rightarrow R_\perp$.
2. For $\nu, \nu' \in \mathcal{W} \rightarrow_{mon} CURel(A)$, 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 $CURel(A)$ belongs to $\mathcal{W} \rightarrow_{mon} CURel(A)$ 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 $\varpi_n \in \nu(\Delta) \rightarrow \nu(\Delta')_\perp$.

Lemma A.3. Let $(D, (\varpi_n)_{n \in \omega})$ be a uniform cppo. Abbreviate $AURel(D) = AURel(D, (\varpi_n)_{n \in \omega})$ and consider the metrics on $AURel(D)$ and $\mathcal{W} \rightarrow_{mon} AURel(D)$ given by Propositions 4.14 and 4.15.

1. For $R, S \in AURel(D)$, we have that $R \stackrel{n}{=} S$ if and only if $\varpi_n \in R \rightarrow S$ and $\varpi_n \in S \rightarrow R$.
2. For $\xi, \xi' \in \mathcal{W} \rightarrow_{mon} AURel(D)$, 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 $AURel(D)$ belongs to $\mathcal{W} \rightarrow_{mon} AURel(D)$ 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 $\varpi_n \in \xi(\Delta) \rightarrow \xi(\Delta')$.

References

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

- [2] A. Ahmed. *Semantics of Types for Mutable State*. PhD thesis, Princeton University, 2004.
- [3] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *Proceedings of POPL*, pages 340–353, 2009.
- [4] R. M. Amadio. Recursion over realizability structures. *Information and Computation*, 91(1):55–85, 1991.
- [5] R. M. Amadio and P.-L. Curien. *Domains and Lambda-Calculi*. Cambridge University Press, 1998.
- [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] 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 *Proceedings of TLCA*, number 3461 in LNCS, pages 86–101, 2005.
- [9] 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, 2009. Submitted to journal.
- [10] L. Birkedal, K. Støvring, and J. Thamsborg. Semantics for higher-order store via Kripke models over recursively defined worlds, Dec. 2009. In preparation.
- [11] L. Birkedal, K. Støvring, and J. Thamsborg. Relational parametricity for references and recursive types. In *Proceedings of TLDI*, pages 91–104, 2009.
- [12] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Proceedings of APLAS*, number 4279 in LNCS, pages 79–96, 2006.
- [13] F. Cardone. Relational semantics for recursive types and bounded quantification. In *Proceedings of ICALP*, number 372 in LNCS, pages 164–178, 1989.
- [14] K. Crary and R. Harper. Syntactic logical relations for polymorphic and recursive types. *Electronic Notes in Theoretical Computer Science*, 172:259–299, 2007.

- [15] D. Dreyer, A. Ahmed, and L. Birkedal. Logical step-indexed logical relations. In *Proceedings of LICS*, pages 71–80, 2009.
- [16] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs, July 2009. To Appear in POPL’2010 (accepted for publication).
- [17] A. Gotsman, J. Berdine, B. Cook, N. Rinetzky, and M. Sagiv. Local reasoning for storable locks and threads. In *Proceedings of APLAS*, number 4807 in LNCS, pages 19–37, 2007.
- [18] P. B. Levy. Call-by-push-value: Decomposing call-by-value and call-by-name. *Higher-Order and Symbolic Computation*, 19(4):377–414, 2006.
- [19] D. B. MacQueen, G. D. Plotkin, and R. Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71(1/2):95–130, 1986.
- [20] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.
- [21] R. L. Petersen, L. Birkedal, A. Nanevski, and G. Morrisett. A realizability model for impredicative Hoare type theory. In *Proceedings of ESOP*, number 4960 in LNCS, pages 337–352, 2008.
- [22] B. C. Pierce. *Types and Programming Languages*. The MIT Press, 2002.
- [23] A. M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.
- [24] G. D. Plotkin and J. Power. Computational effects and operations: An overview. *Electronic Notes in Theoretical Computer Science*, 73: 149–163, 2004.
- [25] J. J. M. M. Rutten. Elements of generalized ultrametric domain theory. *Theoretical Computer Science*, 170(1-2):349–381, 1996.
- [26] 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.
- [27] J. Schwinghammer, H. Yang, L. Birkedal, F. Pottier, and B. Reus. A semantic foundation for hidden state, Oct. 2009. Submitted for publication.
- [28] 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.

- [29] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11(4):761–783, 1982.
- [30] K. R. Wagner. *Solving Recursive Domain Equations with Enriched Categories*. PhD thesis, Carnegie Mellon University, 1994.
- [31] G. Winskel. *The Formal Semantics of Programming Languages*. Foundation of Computing Series. The MIT Press, 1993.