Math. Struct. in Comp. Science (2010), *vol.* 20, *pp.* 655–703.
doi:10.1017/S0960129510000162

Realisability semantics of parametric polymorphism, general references and recursive types

LARS BIRKEDAL, KRISTIAN STØVRING

and JACOB THAMSBORG

IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
Email: {birkedal;kss;thamsborg}@itu.dk

Received 12 December 2009; revised 9 May 2010

We present a realisability model for a call-by-value, higher-order programming language with parametric polymorphism, general first-class references, and recursive types. The main novelty is a relational interpretation of open types that include general reference types. The interpretation uses a new approach to modelling 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 paper we develop a semantic model of a call-by-value programming language with impredicative and parametric polymorphism, general first-class references, and recursive types. Our motivations for conducting this study include:

— Extending the approach to reasoning about abstract data types via relational parametricity from pure languages to more realistic languages with effects, in this case general references. We have already discussed this point of view extensively in Birkedal *et al.* (2009).

— Investigating what semantic structures are needed in general models for effects. Indeed, we see the present work as a pilot study for research into general type theories and models of effects (see, for example, Levy (2006) and Plotkin and Power (2004)), in which we identify key ingredients needed for semantic modelling of general first-class references.

— Paving the way for developing models of separation logic for ML-like languages with reference types. Earlier such models of separation logic (Petersen *et al.* 2008) only treated 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.

Overview of the conceptual development of the paper

The development is centred around three recursively defined structures, which are defined in three steps. In slogan form, there is one recursively defined structure for each of the type constructors $\forall$, ref, and μ alluded to in the title.

Step 1: Since the language involves impredicative polymorphism, the semantic model is based on a realisability interpretation (Amadio 1991) over a certain recursively defined predomain V. Using this predomain, we can give a denotational semantics of an untyped version of the language. This part is mostly standard, except for the fact that we model locations as pairs (l, n), with l a natural number corresponding to a standard location and $n \in \mathbb{N} \cup \{\infty\}$ indicating the 'approximation stage' of the location (Birkedal *et al.* 2009). These pairs, called semantic locations, are needed for modelling reference types in step three. Intuitively, the problem with the more standard approach of modelling locations as natural numbers is that such 'flat' locations contain no approximation information that can be used to define relations by induction.

Step 2: To account for the dynamic allocation of typed reference cells, we follow earlier work on modelling simple integer references (Benton and Leperchey 2005) and use a Kripke-style possible worlds model. Here, however, the set of worlds needs to be recursively defined since we treat general references. Semantically, a world maps locations to semantic types, which, following the general realisability 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. This means we need to solve recursive equations of approximately the form

$$\mathcal{W} = \mathbb{N}_0 \rightharpoonup_{fin} \mathcal{T}$$
$$\mathcal{T} = \mathcal{W} \to CURel(V)$$

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

Step 3: Having defined the space in which types should be modelled, we can now define the actual semantics of types. For recursive types, this also involves a recursive definition. Since the space $\mathcal{T}$ of semantic types is a metric space, we can employ Banach's fixed-point theorem to find a solution as the fixed point of a contractive operator on $\mathcal{T}$.[†] This involves interpreting the various type constructors of the language as

[†] Note that the fixed point could also be found using the technique of Pitts (1996); the proof techniques are very similar because of the particular way the requisite metrics are defined. In this paper we do in any

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

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

Limitations

The model we construct does not validate standard equivalences involving local state; indeed, it can only be used to equate computations that allocate references that are essentially 'in lockstep'. More precisely, the model can only relate two program states if they contain the same number of locations. Furthermore, a certain technical requirement on the relations we consider ('uniformity') seems to be too restrictive. In recent work (Birkedal *et al.* 2010b) we have shown that both these problems can be overcome: one can use the techniques presented here to construct a model that validates sophisticated equivalences in the style of Ahmed *et al.* (2009). One key idea is to weaken the model such that the relations constructed can be thought of as inequalities rather than equalities; then one can prove results about contextual approximation rather than contextual equivalence. However, the model in Birkedal *et al.* (2010b) is rather more complicated than the one we present here, where our aim is just 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 paper is organised as follows. In Section 2 we sketch the language we will be considering. In Section 3 we present the untyped semantics – corresponding to step one in the outline above. In Section 4 we present the typed semantics – corresponding to the last two steps. In Section 5 we give an abstract proof that a certain simpler interpretation of reference types is impossible. In Section 6 we present a few examples of reasoning using the model. Finally, we discuss some related work in Section 7.

2. Language

We consider a standard call-by-value language with universal types, iso-recursive types, ML-style reference types, and a ground type of integers. The language is sketched in

case need the metric-space formulation, but not the extra separation of positive and negative arguments in recursive definitions of relations, so we define the meaning of recursive types via Banach's fixed-point theorem (Amadio 1991; Amadio and Curien 1998).

Figure 1. Terms are not intrinsically typed; this allows us to give a denotational semantics of untyped terms. The typing rules are standard (Pierce 2002). In Figure 1, Ξ and Γ range over contexts of type variables and term variables, respectively.

The constructs that involve references have the following informal meaning:

— The term $\mathsf{ref}\, t$ allocates a new reference cell initialised with the result of evaluating t.
— The term $!t$ looks up t in the current store (and gives an error if t evaluates to something that is not a location).
— The term $t_1 := t_2$ assigns the value of t_2 to t_1 (provided t_1 evaluates to a location).

2.1. *Operational semantics*

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

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

The set of *syntactic values* is then given by

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

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

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

The operational semantics is given by two judgements on configurations:

(1) $(\sigma, t) \Downarrow (\sigma', w)$ means that evaluation of t together with the store σ terminates with a value w and a possibly modified store σ'.
(2) $(\sigma, t) \Downarrow \mathit{error}$ means that evaluation of t together with the store σ results in an error, either due to a memory fault or, for example, an attempt to apply an integer constant to an argument.

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

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

$$\begin{aligned} (\varnothing, C[t]) \Downarrow \ &\iff \ (\varnothing, C[t']) \Downarrow \\ (\varnothing, C[t]) \Downarrow \mathit{error} \ &\iff \ (\varnothing, C[t']) \Downarrow \mathit{error}. \end{aligned}$$

3. Untyped semantics

We now give a denotational semantics for the untyped term language of the previous section. As is usual for models of untyped languages, the semantics is given by means

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

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

Typing rules:

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

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

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

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

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

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

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

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

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

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

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

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

Fig. 1. Programming language

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

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

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

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

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

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

Fig. 2. Operational semantics (selected rules)

of a 'universal' complete partial order (cpo) in which one can inject integers, pairs, functions, and so on. 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 modelled as elements of the cpo $Loc = \mathbb{N}_0 \times \overline{\omega}$ where $\overline{\omega}$ is the 'vertical natural numbers' cpo $1 \sqsubset 2 \sqsubset \cdots \sqsubset n \sqsubset \cdots \sqsubset \infty$. (For notational reasons it is convenient to call the least element 1 rather than 0.) The intuitive idea is that locations can be approximated: the element $(l, \infty) \in Loc$ 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 – see the remark before Proposition 3.2 below.

3.1. *Domain-theoretic preliminaries*

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

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

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

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, which will be represented using the Kleisli category for the lifting monad $(-)_\perp$. Let pCpo be the Kleisli category for the lifting monad: objects are cpos, while morphisms from A to B are continuous functions from A to $B_\perp$. The identity maps in pCpo are written $\overline{id}$, and are given by lifting: $\overline{id} = \lambda a.\lfloor a \rfloor$. Composition in pCpo is written $\mathbin{\overline{\circ}}$:

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

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

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

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

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

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

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

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

3.2. *Uniform cpos*

The standard methods for solving recursive domain equations give solutions that satisfy certain induction principles (Smyth and Plotkin 1982; Pitts 1996). 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 $\varpi_0(a)$, $\varpi_1(a)$, and so on. These functions therefore provide a handle for proving properties about A by induction on n.

Definition 3.1.

(1) A *uniform cpo* $(A, (\varpi_n)_{n\in\omega})$ is a cpo A together with a family $(\varpi_n)_{n\in\omega}$ of continuous functions from A to $A_\perp$ satisfying

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

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

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

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

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

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

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

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

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

Remark. Uniform cppos are exactly the algebras for a certain monad on the category of cppos and strict, continuous functions. The monad is given by a natural monoid structure on $\overline{\omega}$:

$$\begin{aligned} T_{\mathrm{u}} D &= \overline{\omega} \otimes D \\ \eta_{\mathrm{u}} e &= (\infty, e) \\ \mu_{\mathrm{u}}(m,(n,e)) &= (\min(m,n), e). \end{aligned}$$

Our locations are modelled using a free algebra for this monad:

$$Loc_\perp \cong \overline{\omega} \otimes (\mathbb{N}_0)_\perp.$$

Uniform cppos were called *rank-ordered cpos* in Baier and Majster-Cederbaum (1997).

3.3. *A universal uniform cpo*

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

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

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

(1) The following isomorphism holds in Cpo:

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

where

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

and

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

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

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

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

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

Proof (sketch). We solve the predomain equation (3.12) as usual (Smyth and Plotkin 1982). This gives an almost correct uniform cpo $(V, (\varpi_n)_{n\in\omega})$, the only problem being 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, and so on) as in the proposition (by induction on n). All the requirements in the definition of a uniform cpo except for the fact that $\sqcup_n \pi_n = \overline{id}$ are easy to show. To show that $\sqcup_n \pi_n = \overline{id}$, we first show by induction on m that $\pi_n \mathbin{\overline{\circ}} \varpi_m = \varpi_m \mathbin{\overline{\circ}} \pi_n$ for all n, and that

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

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

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

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

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

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

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

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

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

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

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

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

$$\pi_{n+1}(in_\to\, f) = \left\lfloor in_\to \left(\lambda v. \begin{cases} \pi^T_{n+1}\,(f\, v') & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \right) \right\rfloor \tag{3.22}$$

Here the functions $\pi^S_n : S \to S_\bot$ and $\pi^K_n : K \to K$ and $\pi^T_n : TV \to TV$ are defined as follows:

$$\pi^S_0 = \lambda s.\bot \qquad \pi^K_0 = \lambda k.\bot \qquad \pi^T_0 = \lambda c.\bot \tag{3.23}$$

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

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

$$\pi^T_{n+1}(c) = \lambda k.\lambda s. \begin{cases} c\,(\pi^K_{n+1}\, k)\, s' & \text{if } \pi^S_{n+1}\, s = s' \\ \bot & \text{otherwise}. \end{cases} \tag{3.26}$$

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

From now on, we let V and $(\pi_n)_{n\in\omega}$ be as in the proposition above. We will also use the same abbreviations, notation for injections, and so on, as we introduced in the proposition: in particular, $TV = (V \to S \to Ans) \to S \to Ans$. Additionally, we use the abbreviations $\lambda_l = in_{Loc}(l,\infty)$ and $\lambda^n_l = in_{Loc}(l,n)$; recall that we have $n \geqslant 1$ here. Let $error_{Ans} \in Ans$ be the 'error answer' and let $error \in TV$ be the 'error computation':

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

We shall need the following proposition later.

Proposition 3.3.

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

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

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

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

$$assign\, v_1\, v_2 = \lambda k\, \lambda s. \begin{cases} k\, (in_1 *)\, (s[l \mapsto v_2]) & \text{if } v_1 = \lambda_l \text{ and } l \in \mathrm{dom}(s) \\ k\, (in_1 *)\, (s[l \mapsto v_2']) & \text{if } v_1 = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n(v_2) = \lfloor v_2' \rfloor \\ \bot_{Ans} & \text{if } v_1 = \lambda_l^{n+1},\ l \in \mathrm{dom}(s), \\ & \text{and } \pi_n(v_2) = \bot \\ error_{Ans} & \text{otherwise} \end{cases}$$

Fig. 4. Functions used for interpreting reference operations

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

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

Notice that defining, for example, $lookup(\lambda_l^{n+1})(k)(s) = \bot$ for $l \in \mathrm{dom}(s)$, and hence avoiding mentioning the projection functions, would not be sufficient since then *lookup* would not be continuous.

We are now ready to define the untyped semantics.

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

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

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

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

where

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

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

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

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

Fig. 5. Untyped semantics of terms

3.4. *Soundness and adequacy of the untyped semantics*

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

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

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

We then have soundness:

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

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

Computational adequacy can be stated as follows:

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

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

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

4. Typed semantics

In this section we present a 'typed semantics', that is, 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, that is, 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'.

We assume familiarity with the basic properties of metric spaces (Smyth 1992), but the relevant definitions are repeated below.

4.1. *Ultrametric spaces*

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

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

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

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

(iii′) $d(x, z) \leqslant \max(d(x, y), d(y, z))$.

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

When we refer to 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 $m, n \geqslant N$.
(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 $n \geqslant N$.
(3) A *complete metric space* is a metric space in which every Cauchy sequence has a limit.

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

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

To avoid confusion, we will refer to the elements of $\mathbb{N}^\omega$ as *strings* rather than sequences. Here the ultrametric inequality simply states that if x and y agree on the first n 'characters' and y and z also agree on the first n characters, then x and z agree on the first n characters. A Cauchy sequence in $\mathbb{N}^\omega$ is a sequence of strings $(x_n)_{n\in\omega}$ in which the individual characters 'stabilise': for all m, there exists $N \in \omega$ such that $x_{n_1}(m) = x_{n_2}(m)$ for all $n_1, n_2 \geqslant N$. In other words, there is a number k such that $x_n(m) = k$ for almost all n, that is, 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 the (Euclidean) distance between elements, but rather as a measure of the degree of similarity between elements.

Definition 4.3.

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

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

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

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

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

Limits are pointwise for both products and exponentials.

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

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

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, that is, a unique element x of X such that $f(x) = x$.

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

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

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

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

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

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

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

Definition 4.8.

(1) A functor $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ is *locally non-expansive* if

$$d(F(f, g), F(f', g')) \leqslant \max(d(f, f'), d(g, g'))$$

for all non-expansive functions f, f', g, and g'.

(2) A functor $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ is *locally contractive* if there exists $c < 1$ such that

$$d(F(f, g), F(f', g')) \leqslant c \cdot \max(d(f, f'), d(g, g'))$$

for all non-expansive functions f, f', g, and g'.

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

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

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

(2) Let $F : \mathsf{CBUlt}^{\mathrm{op}} \times \mathsf{CBUlt} \to \mathsf{CBUlt}$ be a locally non-expansive functor. The functor $c \cdot F$ given by

$$\begin{aligned} (c \cdot F)((X_1, d_1), (X_2, d_2)) &= c \cdot F((X_1, d_1), (X_2, d_2)) \\ (c \cdot F)(f, g) &= F(f, g) \end{aligned}$$

is locally contractive.

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

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

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

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

4.2. *The space of semantic types*

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

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

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

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

Definition 4.12.

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

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

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

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

First, as in Amadio (1991), we obtain the following proposition.

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

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

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

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

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

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

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

We also have the converse 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}, and so on. Now define the relation R by

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

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

$$(\varpi_n(\varpi_m\, e), \varpi_n(\varpi_m\, e')) = (\varpi_m(\varpi_n\, e), \varpi_m(\varpi_n\, e')) \in R_{M_n},$$

and hence $(\varpi_m\, e, \varpi_m\, e') \in R$.

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

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

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

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

since $m \geqslant M_n$.

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

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

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

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

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

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

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

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

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

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

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

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

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

Proposition 4.18. Let $(A, (\varpi_n)_{n\in\omega})$ be a uniform cpo. In the following we use the abbreviation $CURel(A) = CURel(A, (\varpi_n)_{n\in\omega})$.

(1) The set $CURel(A)$ is a complete 1-bounded ultrametric space with the distance function given by

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

(2) Let $(X,d) \in CBUlt$, and let $(\mathbb{N}_0 \rightharpoonup_{fin} X) \to_{mon} CURel(A)$ be the set of functions v from $\mathbb{N}_0 \rightharpoonup_{fin} X$ to $CURel(A)$ that are both non-expansive and monotone in the sense that $\Delta \leqslant \Delta'$ implies $v(\Delta) \subseteq v(\Delta')$. This set is a complete 1-bounded ultrametric space with the 'sup'-metric given by

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

Proof.

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

$$R \in CURel(A) \quad \text{if and only if} \quad R_\perp \in AURel(A_\perp) \tag{4.2}$$

for all R in $Rel(A)$. Proposition 4.14 also 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, that is, $d(R,S) = d(R_\perp, S_\perp)$. 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 (that is, 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 (4.2) implies that it is isomorphic to $CURel(A)$, which is therefore also complete.

More generally, we let $(D, (\varpi'_n)_{n\in\omega})$ be a uniform cppo, and use the abbreviation $AURel(D) = AURel(D, (\varpi'_n)_{n\in\omega})$ and then show that the subset $SAURel(D) \subseteq AURel(D)$ of strict relations is closed. So we let $(R_m)_{m\in\omega}$ be a sequence of strict relations (elements of $SAURel(D)$) with limit R for some $R \in AURel(D)$, and then show that R is strict. So we let $(\perp, e) \in R$ and show that $e = \perp$. (The case where $(e,\perp) \in R$ is completely symmetric.) By (3.9), it suffices to show that $\varpi'_n\, e = \perp$ for all n. Given n, choose m large enough that $d(R, R_m) \leqslant 2^{-n}$. Then $\varpi'_n \in R \to R_m$ by

Proposition 4.13(1), so $(\bot, \varpi_n' e) = (\varpi_n' \bot, \varpi_n' e) \in R_m$. But this implies that $\varpi_n' e = \bot$ since R_m is strict, therefore we have shown that R is strict.

(2) In the proof of Part 1 we showed that $CURel(A)$ is isomorphic to the complete 1-bounded metric space $SAURel(A_\bot)$ of strict, uniform and admissible relations on $A_\bot$. The isomorphism is the lifting operator on relations, and this operator clearly preserves and reflects set-theoretic inclusion, that is, $R \subseteq S$ if and only if $R_\bot \subseteq S_\bot$. It therefore suffices to show that the set $(\mathbb{N}_0 \rightarrow_{fin} X) \rightarrow_{mon} SAURel(A_\bot)$ of non-expansive and monotone functions from $\mathbb{N}_0 \rightarrow_{fin} X$ to $SAURel(A_\bot)$ is a complete metric space with the 'sup' metric on functions:

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

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

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

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 $F : CBUlt^{\mathrm{op}} \rightarrow CBUlt$ in the natural way:

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

Proof. Let (X_1, d_1) and (X_2, d_2) be complete 1-bounded ultrametric spaces. For every non-expansive function f from X_2 to X_1, it is clear that the $F(f)$ given above is 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 $F(f)(v)$ is monotone for every v in $F(X_1, d_1)$: if we let $\Delta, \Delta' \in (\mathbb{N}_0 \rightarrow_{fin} X_2)$ such that $\Delta \leqslant \Delta'$, then $f \circ \Delta \leqslant f \circ \Delta'$ by the definition of $\leqslant$, and therefore

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

since v is monotone.

We now show for all non-expansive functions f and f' from X_2 to X_1, all v and v' in $(\mathbb{N}_0 \rightarrow_{fin} X_1) \rightarrow_{mon} CURel(V)$, and all Δ and Δ' in $(\mathbb{N}_0 \rightarrow_{fin} X_2)$, that

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

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

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

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

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

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

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

which shows (4.3).

Now, for all f and v, taking $f' = f$ and $v' = v$ in (4.3) shows that $F(f)(v)$ is non-expansive. Similarly, taking $f' = f$ and $\Delta' = \Delta$ in (4.3) shows that $F(f)$ is non-expansive. Summarising, 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:

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

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

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

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

Proposition 4.19, Proposition 4.9 (with $c = 1/2$), and Theorem 4.10 now immediately imply the following theorem.

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

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

holds in $CBUlt$.

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

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

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

4.3. *Interpretation of types*

In the following, let $\widehat{\mathscr{T}}$ be a complete 1-bounded ultrametric space satisfying (4.4), and let $i : \widehat{\mathscr{T}} \to \frac{1}{2}((\mathbb{N}_0 \to_{fin} \widehat{\mathscr{T}}) \to_{mon} CURel(V))$ be an isomorphism with inverse $i^{-1} : \frac{1}{2}((\mathbb{N}_0 \to_{fin} \widehat{\mathscr{T}}) \to_{mon} CURel(V)) \to \widehat{\mathscr{T}}$. For convenience, we use the following abbreviations (where the names $\mathscr{W}$ and $\mathscr{T}$ are intended to indicate 'worlds' and 'types', respectively):

$$\mathscr{W} = \mathbb{N}_0 \to_{fin} \widehat{\mathscr{T}}$$
$$\mathscr{T} = \mathscr{W} \to_{mon} CURel(V).$$

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

We choose $\mathscr{T}$ as our space of semantic types: types of the language will be interpreted as elements of $\mathscr{T}$, that is, as certain world-indexed families of relations on V. We also define families of relations on 'states' (elements of S), 'continuations' (elements of $K = V \to S \to Ans$) and 'computations' (elements of TV).

Definition 4.22. We use the abbreviations

$$AURel(TV) = AURel(TV, (\pi_n^T)_{n\in\omega})$$
$$AURel(K) = AURel(K, (\pi_n^K)_{n\in\omega})$$
$$CURel(S) = CURel(S, (\pi_n^S)_{n\in\omega}).$$

Let

$$\mathscr{T}_T = \mathscr{W} \to_{mon} AURel(TV)$$
$$\mathscr{T}_K = \mathscr{W} \to_{mon} AURel(K)$$

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

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

be the complete uniform 1-bounded ultrametric space obtained from Propositions 4.15 and 4.18 and the exponential in CBUlt. (The elements of $\mathscr{T}_S$ 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 the n-approximated equality of elements.

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

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

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

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

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

This property will be used frequently below.

In order to interpret the types of the language as elements of $\mathscr{T}$, we still need to define a number of operators on $\mathscr{T}$ (and $\mathscr{T}_T$ and $\mathscr{T}_K$) that will be used to interpret the various type constructors of the language; these operators are shown in the lower part of Figure 6. Notice that the operator *ref* is defined in terms of n-approximated equality $\overset{n}{=}$ 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 (for example, *ref* actually maps elements of $\mathscr{T}$ into $\mathscr{T}$.) In order to interpret recursive types, however, we also need to verify that the operators are non-expansive.

The proofs below depend on a number of lemmas, which are given in Appendix A, that provide more concrete descriptions of the metric spaces involved. In particular, the factor $1/2$ in (4.4) implies that worlds that are '$(n+1)$-equal' only contain 'n-equal' semantic types.

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

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

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

for all $\Delta, \Delta' \in \mathscr{W}$. Uniformity then follows from this implication by taking $\Delta' = \Delta$ and using Lemma A.2(1), and non-expansiveness of *states* follows from Lemma A.2(1) and symmetry. So, let $\Delta \overset{n}{=} \Delta'$ and let $(s, s') \in states(\Delta)$; 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 $(s_0, s_0') \in states(\Delta')$. If $n = 0$, we are done by (3.23), so we assume that $n > 0$. Then $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ by the definition of the metric on $\mathscr{W}$, and, furthermore, for every $l \in \mathrm{dom}(\Delta)$,

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

By transitivity (Proposition 4.24),

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

so Lemma A.2(1) gives

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

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

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

The following operators and elements are used in the above:

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

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

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

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

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

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

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

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

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

Fig. 6. Interpretation of types

Since the above holds for every $l \in \mathrm{dom}(\Delta)$, Equation (3.24) gives either $\pi_n^S(s) = \pi_n^S(s') = \bot$, and we are done, or $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ for some s_0 and s_0' such that $(s_0(l), s_0'(l)) \in i(\Delta'(l))\,(\Delta')$ for all $l \in \mathrm{dom}(\Delta')$. But the latter means exactly that $(s_0, s_0') \in \mathit{states}(\Delta')$. □

Lemma 4.26. Let Δ, Δ' and Δ_1 be elements of $\mathscr{W}$ such that $\Delta \overset{n}{=} \Delta'$ and $\Delta \leqslant \Delta_1$. There exists a Δ_1' such that $\Delta_1 \overset{n}{=} \Delta_1'$ and $\Delta' \leqslant \Delta_1'$.

Proof. If $n = 0$, we can take $\Delta_1' = \Delta'$; in fact, any extension of Δ' would do. If $n > 0$, we have $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ by definition of the metric on $\mathscr{W}$. Now define $\Delta_1' \in \mathscr{W}$ with $\mathrm{dom}(\Delta_1') = \mathrm{dom}(\Delta_1)$ by

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

Clearly, $\Delta' \leqslant \Delta_1'$ since $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$. Also, by the definition of the metric on $\mathscr{W}$ (as a maximum of the distances for each 'l'), $d(\Delta_1, \Delta_1') = d(\Delta, \Delta') \leqslant 2^{-n}$. □

Lemma 4.27. The operators $\times$, $+$, *ref*, $\to$, *cont* and *comp* defined in the lower part of Figure 6 are non-expansive.

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

$\times : \mathscr{T} \times \mathscr{T} \to \mathscr{T}$:

It is easy to see that $(v_1 \times v_2)(\Delta)$ is complete for all $\Delta \in \mathscr{W}$. To see that $v_1 \times v_2$ belongs to $\mathscr{T}$, it therefore suffices to verify the two conditions of Lemma A.2(3).

Condition (a), monotonicity, is immediate.

For Condition (b), we show the more general fact, which also implies the non-expansiveness of $\times$, that for all v_1, v_2, v_1', and v_2' in $\mathscr{T}$ and all Δ and Δ' in $\mathbb{N}_0 \to_{\mathit{fin}} \widehat{\mathscr{T}}$,

$$v_1 \overset{n}{=} v_1' \ \wedge\ v_2 \overset{n}{=} v_2' \ \wedge\ \Delta \overset{n}{=} \Delta' \implies \pi_n \in (v_1 \times v_2)(\Delta) \to (v_1' \times v_2')(\Delta)_\bot\,.$$

Condition (b) then follows by taking $v_1 = v_1'$ and $v_2 = v_2'$. Non-expansiveness of $\times$ follows by taking $\Delta = \Delta'$ and using parts 1 and 2 of Lemma A.2 (and symmetry).

So, we assume that $v_1 \overset{n}{=} v_1'$ and $v_2 \overset{n}{=} v_2'$ and $\Delta \overset{n}{=} \Delta'$, and let

$$(in_\times(v_1, v_2), in_\times(v_1', v_2')) \in (v_1 \times v_2)(\Delta)\,.$$

We must show that either:

(1) $\pi_n(in_\times(v_1, v_2)) = \pi_n(in_\times(v_1', v_2')) = \bot$; or

(2) $\pi_n(in_\times(v_1, v_2)) = \lfloor w \rfloor$ and $\pi_n(in_\times(v_1', v_2')) = \lfloor w' \rfloor$ for some w and w' such that $(w, w') \in (v_1' \times v_2')(\Delta')$.

If $n = 0$, we are done by Equation (3.4), so we assume that $n > 0$. By the definition of $(v_1 \times v_2)(\Delta)$, we know that $(v_1, v_1') \in v_1(\Delta)$ and $(v_2, v_2') \in v_2(\Delta)$. Since v_1 and v_2 are non-expansive functions, (4.5) gives

$$v_1(\Delta) \overset{n-1}{=} v_1'(\Delta') \quad \text{and} \quad v_2(\Delta) \overset{n-1}{=} v_2'(\Delta').$$

Therefore, $\pi_{n-1} \in v_1(\Delta) \to v_1'(\Delta')_\perp$ and $\pi_{n-1} \in v_2(\Delta) \to v_2'(\Delta')_\perp$ by Lemma A.2(1). By the definition of $v_i(\Delta) \to v_i'(\Delta')_\perp$ (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 $(w_1, w_1') \in v_1'(\Delta')$ and $(w_2, w_2') \in v_2'(\Delta')$ where $\pi_{n-1}(v_1) = \lfloor w_1 \rfloor$ and $\pi_{n-1}(v_1') = \lfloor w_1' \rfloor$ and $\pi_{n-1}(v_2) = \lfloor w_2 \rfloor$ and $\pi_{n-1}(v_2') = \lfloor w_2' \rfloor$.

For case (1), (3.18) gives $\pi_n(in_\times(v_1, v_2)) = \pi_n(in_\times(v_1', v_2')) = \perp$ and we are done.
For case (2), (3.18) gives $\pi_n(in_\times(v_1, v_2)) = \lfloor in_\times(w_1, w_2) \rfloor$ and $\pi_n(in_\times(v_1', v_2')) = \lfloor in_\times(w_1', w_2') \rfloor$. By the definition of $(v_1' \times v_2')(\Delta')$, we have $(in_\times(w_1, w_2), in_\times(w_1', w_2')) \in (v_1' \times v_2')(\Delta')$, and we are done.

ref $: \mathscr{T} \to \mathscr{T}$:

First, $ref(v)(\Delta)$ is complete for all Δ because of the general fact that if $R \overset{n}{=} S$ for all $n \in \omega$, then $d(R, S) = 0$ and hence $R = S$. It is also easy to see that $ref(v)$ is monotone. Using similar reasoning to the previous case, we then show that $ref(v)$ belongs to $\mathscr{T}$ and that *ref* is non-expansive by showing that

$$v \overset{n}{=} v' \ \wedge\ \Delta \overset{n}{=} \Delta' \implies \pi_n \in ref(v)(\Delta) \to ref(v')(\Delta')_\perp$$

for all v and v' in $\mathscr{T}$ and all Δ and Δ' in $\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathscr{T}}$.
So, we assume that $v \overset{n}{=} v'$ and $\Delta \overset{n}{=} \Delta'$, and let $(\lambda_l^m, \lambda_l^m) \in ref(v)(\Delta)$. (The case where $(\lambda_l, \lambda_l) \in ref(v)(\Delta)$ is similar, but slightly easier.) If $n = 0$, we are done by Equation (3.4). If $n > 0$, (3.17) gives $\pi_n(\lambda_l^m) = \lfloor \lambda_l^{\min(n,m)} \rfloor$, and it therefore just remains for us to show that $(\lambda_l^{\min(n,m)}, \lambda_l^{\min(n,m)}) \in ref(v')(\Delta')$. To do this, we let $l \in \mathrm{dom}(\Delta')$ and $\Delta_1' \geqslant \Delta'$ and show that

$$i(\Delta'(l))(\Delta_1') \overset{\min(n,m)-1}{=} v'(\Delta_1').$$

Lemma 4.26 gives a $\Delta_1 \geqslant \Delta$ such that $\Delta_1 \overset{n}{=} \Delta_1'$. So,

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

Hence, by transitivity, $i(\Delta'(l))(\Delta_1') \overset{\min(n,m)-1}{=} v'(\Delta_1')$.

$+ : \mathscr{T} \times \mathscr{T} \to \mathscr{T}$:

It is easy to see that $(v_1 + v_2)(\Delta)$ is complete for all $\Delta \in W$, and that $v_1 + v_2$ is monotone. It then suffices to show that

$$v_1 \overset{n}{=} v_1' \ \wedge\ v_2 \overset{n}{=} v_2' \ \wedge\ \Delta \overset{n}{=} \Delta' \implies \pi_n \in (v_1 + v_2)(\Delta) \to (v_1' + v_2')(\Delta')_\perp$$

for all v_1, v_2, v_1', and v_2' in $\mathscr{T}$ and all Δ and Δ' in $\mathbb{N}_0 \rightharpoonup_{fin} \widehat{\mathscr{T}}$.
So we assume that $v_1 \overset{n}{=} v_1'$ and $v_2 \overset{n}{=} v_2'$ and $\Delta \overset{n}{=} \Delta'$, and let

$$(in_+(\iota_1\, v), in_+(\iota_1 v')) \in (v_1 + v_2)(\Delta).$$

(The case with ι_2 instead of ι_1 is completely symmetric.) If $n = 0$, we are done by Equation (3.4), so we assume that $n > 0$. By the definition of $(v_1 + v_2)(\Delta)$, we have $(v, v') \in v_1(\Delta)$. Then, since $v_1 \stackrel{n}{=} v_1'$ and $\Delta \stackrel{n}{=} \Delta'$ implies $v_1(\Delta) \stackrel{n}{=} v_1'(\Delta')$, there are two cases:

(1) $\pi_{n-1}(v) = \pi_{n-1}(v') = \bot$, and we are done; or

(2) $\pi_{n-1}(v) = \lfloor w \rfloor$ and $\pi_{n-1}(v') = \lfloor w' \rfloor$ where $(w, w') \in v_1'(\Delta')$. But then (3.19) gives

$$\pi_n(in_+(\iota_1 v)) = \lfloor in_+(\iota_1 w) \rfloor$$
$$\pi_n(in_+(\iota_1 v')) = \lfloor in_+(\iota_1 w') \rfloor$$

with $(in_+(\iota_1 w), in_+(\iota_1 w')) \in (v_1' + v_2')(\Delta')$.

cont $: \mathscr{T} \to \mathscr{T}_K$:

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

$$v \stackrel{n}{=} v' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n^K \in cont(v)(\Delta) \to cont(v')(\Delta')$$

for all v and v' in $\mathscr{T}$ and all Δ and Δ' in $\mathscr{W}$. So we assume that $v \stackrel{n}{=} v'$ and $\Delta \stackrel{n}{=} \Delta'$, let $(k, k') \in cont(v)(\Delta)$ and then show that $(\pi_n^K(k), \pi_n^K(k')) \in cont(v')(\Delta')$.

If $n = 0$, this follows from (3.23) and the fact that $cont(v')(\Delta')$ is pointed.

If $n > 0$, we let $\Delta_1' \geqslant \Delta'$ and $(v, v') \in v'(\Delta_1')$ and $(s, s') \in states(\Delta_1')$, and then show that $(\pi_n^K(k)\, v\, s, \pi_n^K(k')\, v'\, s') \in R_{Ans}$. First, Lemma 4.26 gives a $\Delta_1 \geqslant \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta_1'$. By (4.5), $v(\Delta_1) \stackrel{n}{=} v'(\Delta_1')$. Furthermore, the fact that *states* belongs to $\mathscr{T}_S$, as we showed earlier, implies that $states(\Delta_1) \stackrel{n}{=} states(\Delta_1')$. Therefore, by (3.25), either:

(1) $\pi_n^K(k)\, v\, s = \pi_n^K(k')\, v'\, s' = \bot$, and we are done; or

(2) $\pi_n^K(k)\, v\, s = k\, w\, s_0$ and $\pi_n^K(k')\, v'\, s' = k'\, w'\, s_0'$ where $\pi_n(v) = \lfloor w \rfloor$ and $\pi_n(v') = \lfloor w' \rfloor$ and $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ with $(w, w') \in v(\Delta_1)$ and $(s_0, s_0') \in states(\Delta_1)$, in which case, $(k\, w\, s_0, k'\, w'\, s_0') \in R_{Ans}$ since $(k, k') \in cont(v)(\Delta)$ and $\Delta \leqslant \Delta_1$.

comp $: \mathscr{T} \to \mathscr{T}_T$:

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

$$v \stackrel{n}{=} v' \ \wedge \ \Delta \stackrel{n}{=} \Delta' \implies \pi_n^T \in comp(v)(\Delta) \to comp(v')(\Delta')$$

for all v and v' in $\mathscr{T}$ and all Δ and Δ' in $\mathscr{W}$. So we assume that $v \stackrel{n}{=} v'$ and $\Delta \stackrel{n}{=} \Delta'$, let $(c, c') \in comp(v)(\Delta)$, and then show that $(\pi_n^T(c), \pi_n^T(c')) \in comp(v')(\Delta')$.

If $n = 0$, this follows from (3.23) and the fact that $comp(v')(\Delta')$ is pointed.

If $n > 0$, we let $\Delta_1' \geqslant \Delta'$ and $(k, k') \in cont(v')(\Delta_1')$ and $(s, s') \in states(\Delta_1')$, and then show that $(\pi_n^T(c)\, k\, s, \pi_n^T(c')\, k'\, s') \in R_{Ans}$. Lemma 4.26 gives a $\Delta_1 \geqslant \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta_1'$. Since *cont* is non-expansive,

$$cont(v)(\Delta_1) \stackrel{n}{=} cont(v')(\Delta_1) \stackrel{n}{=} cont(v')(\Delta_1').$$

Furthermore, the fact that *states* belongs to $\mathscr{T}_S$ implies that $states(\Delta_1) \stackrel{n}{=} states(\Delta_1')$. Therefore, by (3.26), either:

(1) $\pi_n^T(c)\, k\, s = \pi_n^T(c')\, k'\, s' = \bot$, and we are done; or

(2) $\pi_n^T(c)\,k\,s = c\,(\pi_n^K(k))\,s_0$ and $\pi_n^T(c')\,k'\,s' = c'\,(\pi_n^K(k'))\,s_0'$ where $\pi_n^S(s) = \lfloor s_0 \rfloor$ and $\pi_n^S(s') = \lfloor s_0' \rfloor$ with $(\pi_n^K(k), \pi_n^K(k')) \in cont(v)(\Delta_1)$ and $(s_0, s_0') \in states(\Delta_1)$, in which case,

$$(c\,(\pi_n^K(k))\,s_0, c'\,(\pi_n^K(k'))\,s_0') \in R_{Ans}$$

since $(c, c') \in comp(v)(\Delta)$ and $\Delta \leqslant \Delta_1$.

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

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

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

for all v and v' in $\mathscr{T}$, all ξ and ξ' in $\mathscr{T}_T$, and all Δ and Δ' in $\mathscr{W}$. So we assume that $v \stackrel{n}{=} v'$ and $\xi \stackrel{n}{=} \xi'$ and $\Delta \stackrel{n}{=} \Delta'$, and let $(in_\rightarrow f, in_\rightarrow f') \in (v \rightarrow \xi)(\Delta)$. If $n = 0$, we are done by Equation (3.4), so we assume that $n > 0$. Define the two functions

$$g = \lambda v.\, \begin{cases} \pi_n^T\,(f\,w) & \text{if } \pi_{n-1}\,v = \lfloor w \rfloor \\ \perp & \text{otherwise} \end{cases}$$

$$g' = \lambda v'.\, \begin{cases} \pi_n^T\,(f'\,w') & \text{if } \pi_{n-1}\,v' = \lfloor w' \rfloor \\ \perp & \text{otherwise.} \end{cases}$$

By (3.22), it suffices to show that $(in_\rightarrow(g), in_\rightarrow(g')) \in (v' \rightarrow \xi')(\Delta')$. To do this, we let $\Delta_1' \geqslant \Delta'$ and $(v, v') \in v'(\Delta_1')$, and then show that $(g(v), g'(v')) \in \xi'(\Delta_1')$. Lemma 4.26 gives a $\Delta_1 \geqslant \Delta$ such that $\Delta_1 \stackrel{n}{=} \Delta_1'$. So $v(\Delta_1) \stackrel{n-1}{=} v'(\Delta_1')$, and there are therefore two cases:

(1) $\pi_{n-1}\,v = \pi_{n-1}\,v' = \perp$, and we are done; or

(2) $\pi_{n-1}\,v = \lfloor w \rfloor$ and $\pi_{n-1}\,v' = \lfloor w' \rfloor$ for some w, w' such that $(w, w') \in v(\Delta_1)$, in which case $(f\,w, f'\,w') \in \xi(\Delta_1)$ since $(f, f') \in (v \rightarrow \xi)(\Delta)$ and $\Delta_1 \geqslant \Delta$. Then, by (4.5), $\xi(\Delta_1) \stackrel{n}{=} \xi'(\Delta_1')$, so $(\pi_n^T(f\,w), \pi_n^T(f'\,w')) \in \xi'(\Delta_1')$. But this means precisely that $(g(v), g'(v')) \in \xi'(\Delta_1')$. □

It is at this point that we need the approximate locations λ_l^n in order to show that *ref* is well defined (and non-expansive). Suppose, for the sake of argument, that locations were modelled simply using a flat cpo of natural numbers, that is, suppose $Loc = \mathbb{N}_0$ and $\pi_1(in_{Loc}\,l) = \lfloor in_{Loc}\,l \rfloor$ for all $l \in \mathbb{N}_0$. The definition of *ref* would then have the form $ref(v)(\Delta) = \{(in_{Loc}\,l, in_{Loc}\,l) \mid l \in \mathrm{dom}(\Delta) \wedge \ldots\}$. The function $ref(v)$ from worlds to relations must be non-expansive. But assume then that $\Delta =_1 \Delta'$. Then $ref(v)(\Delta) =_1 ref(v)(\Delta')$ by non-expansiveness, and hence $ref(v)(\Delta) = ref(v)(\Delta')$ since π_1 is the (lifted) identity on locations. In other words, $ref(v)$ would only depend on the 'first approximation' of its argument world Δ, which can never be correct, no matter what the particular definition of *ref*[†]. This observation generalises to variants where $\pi_n(in_{Loc}\,l) = \lfloor in_{Loc}\,l \rfloor$) for some arbitrary finite n.

[†] In particular, the obvious definition of *ref* as $ref(v)(\Delta) = \{(in_{Loc}\,l, in_{Loc}\,l) \mid l \in \mathrm{dom}(\Delta) \wedge \forall \Delta_1 \geqslant \Delta.\ i(\Delta(l))\,(\Delta_1) = v(\Delta_1)\}$ would *not* be well defined since it would not be non-expansive in Δ.

For any finite set Ξ of type variables, the set $\mathscr{T}^{\Xi}$ of functions from Ξ to $\mathscr{T}$ 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 Ξ be a type environment such that $\Xi \vdash \tau$. The *relational interpretation of* τ *with respect to* Ξ is the non-expansive function $[\![\tau]\!]_{\Xi} : \mathscr{T}^{\Xi} \to \mathscr{T}$ defined by induction on τ in Figure 6. The interpretation of recursive types is by appeal to Banach's fixed-point theorem (see Theorem 4.29).

In more detail, well definedness of $[\![\tau]\!]_{\Xi}$ must be argued along with non-expansiveness by induction on τ (see below). This is similar to the more familiar situation encountered with the untyped semantics of terms presented in Section 3: there, well-definedness must be argued along with continuity because of the use of Kleene's fixed-point theorem in the interpretation of $\mathsf{fix}\, f.\lambda x.t$.

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

(1) The function $[\![\tau]\!]_{\Xi} : \mathscr{T}^{\Xi} \to \mathscr{T}$ defined in Figure 6 is non-expansive.
(2) If $\Xi = \Xi', \alpha$, then for all $\varphi' \in \mathscr{T}^{\Xi'}$, we have that $\lambda v.\, \lambda\Delta.\, \{\, (in_{\mu}\, v, in_{\mu}\, v') \mid (v, v') \in [\![\tau]\!]_{\Xi', \alpha} \varphi'[\alpha \mapsto v]\, \Delta \,\}$ is a contractive function from $\mathscr{T}$ to $\mathscr{T}$. In particular, $[\![\mu\alpha.\tau]\!]$ is well-defined.

Proof. First, we generalise Part 2:

(2′) $\lambda\varphi\, \lambda\Delta.\, \{\, (in_{\mu}\, v, in_{\mu}\, v') \mid (v, v') \in [\![\tau]\!]_{\Xi} \varphi\, \Delta \,\}$ is a contractive function from $\mathscr{T}^{\Xi}$ to $\mathscr{T}$.

By the definition of the product metric, 2′ implies 2.

We now show 1 and 2′ by simultaneous induction on τ:

(1) If τ is int, 1 or 0, then $[\![\tau]\!]_{\Xi}$ is a constant function and hence trivially non-expansive. If τ is a type variable α, then non-expansiveness of $[\![\tau]\!]_{\Xi}$ follows directly from the definition of the product metric. In the cases where τ is $\tau_1 \times \tau_2$, $\tau_1 + \tau_2$, $ref\ \tau'$ or $\tau_1 \to \tau_2$, non-expansiveness follows directly from Lemma 4.27 and the induction hypothesis.
We still need to consider the cases where τ is $\mu\alpha.\tau'$ or $\forall\alpha.\tau'$. First, assume that τ is $\mu\alpha.\tau'$ for some τ' such that $\Xi, \alpha \vdash \tau'$. We know from 2′ and the induction hypothesis that $[\![\mu\alpha.\tau']\!]_{\Xi}$ is a (well-defined) function from $\mathscr{T}^{\Xi}$ to $\mathscr{T}$. To show that $[\![\mu\alpha.\tau']\!]_{\Xi}$ is non-expansive, we let $\varphi \overset{n}{=} \varphi'$ and show that $[\![\mu\alpha.\tau']\!]_{\Xi}\varphi \overset{n}{=} [\![\mu\alpha.\tau']\!]_{\Xi}\varphi'$. By Proposition 4.7, it suffices to show that the two contractive functions $g, g' : \mathscr{T} \to \mathscr{T}$ defined by

$$g = \lambda v.\, \lambda\Delta.\, \{\, (in_{\mu}\, v, in_{\mu}\, v') \mid (v, v') \in [\![\tau']\!]_{\Xi, \alpha} \varphi[\alpha \mapsto v]\, \Delta \,\}$$
$$g' = \lambda v.\, \lambda\Delta.\, \{\, (in_{\mu}\, v, in_{\mu}\, v') \mid (v, v') \in [\![\tau']\!]_{\Xi, \alpha} \varphi'[\alpha \mapsto v]\, \Delta \,\}$$

satisfy $g \overset{n}{=} g'$. So we let $v \in \mathscr{T}$ be given and then show that $g\, v \overset{n}{=} g'\, v$. But this follows from 2′ and the induction hypothesis. Therefore, $[\![\mu\alpha.\tau']\!]_{\Xi}$ is non-expansive.
Now we assume that τ is $\forall\alpha.\tau'$ for some τ' such that $\Xi, \alpha \vdash \tau'$. First, $[\![\forall\alpha.\tau']\!]_{\Xi}\varphi\, \Delta$ is complete for all $\Delta \in \mathscr{W}$ since arbitrary intersections of complete relations are

complete. It is also easy to see that $[\![\forall\alpha.\tau']\!]_\Xi\varphi$ is monotone since $comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \to v])$ is monotone for all $v \in \mathscr{T}$. By Lemma A.2(3), it then suffices to show that

$$\varphi \overset{n}{=} \varphi' \ \wedge\ \Delta \overset{n}{=} \Delta' \implies \pi_n \in [\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta \to ([\![\forall\alpha.\tau']\!]_\Xi\varphi'\,\Delta')_\perp$$

for all φ and φ' in $\mathscr{T}^\Xi$ and all Δ and Δ' in $\mathscr{W}$. So, let $(in_\forall\, c, in_\forall\, c') \in [\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta$. If $n = 0$, we are done by Equation (3.4), so we assume that $n > 0$. By (3.21), it then suffices to show that $(in_\forall(\pi_n^T\, c), in_\forall(\pi_n^T\, c')) \in [\![\forall\alpha.\tau']\!]_\Xi\varphi'\,\Delta'$. To this end, we let $\Delta_1' \geqslant \Delta'$ and $v \in \mathscr{T}$ and show that $(\pi_n^T\, c, \pi_n^T\, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto v])\Delta_1'$. Lemma 4.26 gives a $\Delta_1 \geqslant \Delta$ such that $\Delta_1 \overset{n}{=} \Delta_1'$. Then $(c, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v])\Delta_1$ since $(in_\forall\, c, in_\forall\, c') \in [\![\forall\alpha.\tau']\!]_\Xi\varphi\,\Delta$. By the induction hypothesis, $[\![\tau']\!]_{\Xi,\alpha}$ is non-expansive, so

$$[\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v] \overset{n}{=} [\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto v]$$

by the definition of the product metric. Now, the operator *comp* is non-expansive by Lemma 4.27, so

$$comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v]) \overset{n}{=} comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto v])\,.$$

Finally, by (4.5),

$$comp([\![\tau']\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v])\Delta_1 \overset{n}{=} comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto v])\Delta_1'\,,$$

and we can conclude that

$$(\pi_n^T\, c, \pi_n^T\, c') \in comp([\![\tau']\!]_{\Xi,\alpha}\varphi'[\alpha \mapsto v])\Delta_1'$$

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

(2′) Let $G = \lambda\varphi\,\lambda\Delta.\,\{\,(in_\mu\, v, in_\mu\, v') \mid (v, v') \in [\![\tau]\!]_\Xi\varphi\,\Delta\,\}$. We must show that G is a contractive function from $\mathscr{T}^\Xi$ to $\mathscr{T}$. First, it is easy to see that $G(\varphi)$ is monotone and that $G(\varphi)(\Delta)$ is admissible for all φ and Δ. So in order to show that G has codomain $\mathscr{T}$, we still need to verify Condition (b) of Lemma A.2(3). We show the more general property, which also implies that G is contractive, that for all φ and φ' in $\mathscr{T}^\Xi$ and all Δ and Δ' in $\mathscr{W}$,

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

Notice the $n+1$ on the right-hand side: the above property implies that G is contractive with factor $\delta = 1/2$ (by taking $\Delta = \Delta'$ and using Lemma A.2(1) and symmetry).

So, let $\varphi \overset{n}{=} \varphi'$ and $\Delta \overset{n}{=} \Delta'$, and let $(in_\mu\, v, in_\mu\, v') \in G(\varphi)(\Delta)$. We know that $(v, v') \in [\![\tau]\!]_\Xi\varphi\,\Delta$ by the definition of G. Part 1 gives us that $[\![\tau]\!]_\Xi$ is non-expansive, so $[\![\tau]\!]_\Xi\varphi \overset{n}{=} [\![\tau]\!]_\Xi\varphi'$. By (4.5), $[\![\tau]\!]_\Xi\varphi\,\Delta \overset{n}{=} [\![\tau]\!]_\Xi\varphi'\,\Delta'$, so there are two cases:

(a) $\pi_n\, v = \pi_n\, v' = \perp$, in which case we are done by (3.20); or

(b) there exists $(w, w') \in [\![\tau]\!]_\Xi\varphi'\,\Delta'$ such that $\pi_n\, v = \lfloor w \rfloor$ and $\pi_n\, v' = \lfloor w' \rfloor$, in which case, (3.20) gives $\pi_{n+1}(in_\mu\, v) = \lfloor in_\mu\, w \rfloor$ and $\pi_{n+1}(in_\mu\, v') = \lfloor in_\mu\, w' \rfloor$ where $(in_\mu\, w, in_\mu\, w') \in G(\varphi')(\Delta')$.

Finally, in order that we can 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 $\mathscr{T}$ is non-empty. We have already observed that, for example, the constant function $\lambda\Delta.\varnothing$, used to interpret the type $\mathsf{0}$, belongs to $\mathscr{T}$. □

We need the following weakening and substitution properties, which are 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 $\mathscr{T}^{\Xi}$ and $v \in \mathscr{T}$,

$$[\![\tau]\!]_{\Xi}\varphi = [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v]\,.$$

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

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

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

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

4.4. *Interpretation of terms*

For the interpretation of terms, we must show that the untyped meaning of a typed term is related to itself at the appropriate type. We first show that *comp* respects the operations η and $\star$ of the monad T (defined in Section 3.1).

Definition 4.32. For $v \in \mathscr{T}$ and $\xi \in \mathscr{T}_T$ and $\Delta \in \mathscr{W}$, let $v \overset{\Delta}{\to} \xi$ be the binary relation on functions $V \to TV$ defined by

$$v \overset{\Delta}{\to} \xi = \{\,(f,f') \mid \forall\Delta_1 \geqslant \Delta.\forall(v,v') \in v(\Delta_1).\,(f\,v, f'\,v') \in \xi(\Delta_1)\,\}\,.$$

Proposition 4.33. Let $v, v_1, v_2 \in \mathscr{T}$ and $\Delta \in \mathscr{W}$.

(1) If $(v,v') \in v(\Delta)$, then $(\eta\, v, \eta\, v') \in comp(v)(\Delta)$.

(2) If $(c,c') \in comp(v_1)(\Delta)$ and $(f,f') \in v_1 \overset{\Delta}{\to} comp(v_2)$, then

$$(c \star f,\ c' \star f') \in comp(v_2)(\Delta)\,.$$

Proof.

(1) Assume that $(v,v') \in v(\Delta)$. By definition, $\eta\, v = \lambda k.\lambda s.\, k\, v\, s$, and similarly for $\eta\, v'$. To show that $(\eta\, v, \eta\, v') \in comp(v)(\Delta)$, we let $\Delta_1 \geqslant \Delta$ and $(k,k') \in cont(v)(\Delta_1)$ and $(s,s') \in states(\Delta_1)$, and then show that $((\eta\, v)\, k\, s, (\eta\, v')\, k'\, s') \in R_{Ans}$, that is, that $(k\, v\, s, k'\, v'\, s') \in R_{Ans}$. But this follows directly from the definition of $cont(v)(\Delta_1)$ since $(v,v') \in v(\Delta) \subseteq v(\Delta_1)$ by monotonicity.

(2) Assume that $(c,c') \in comp(v_1)(\Delta)$ and $(f,f') \in v_1 \overset{\Delta}{\to} comp(v_2)$. 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 $(c \star f, c' \star f') \in comp(v_2)(\Delta)$, we let $\Delta_1 \geqslant \Delta$ and $(k,k') \in cont(v_2)(\Delta_1)$ and $(s,s') \in states(\Delta_1)$, and show that $((c \star f)\, k\, s, (c' \star f')\, k'\, s') \in R_{Ans}$, that is, 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 $(c, c') \in comp(v_1)(\Delta)$ and $\Delta_1 \geqslant \Delta$ and $(s, s') \in states(\Delta_1)$, it suffices to show that $(\lambda v.\lambda s_1.f\ v\ k\ s_1,\ \lambda v'.\lambda s_1'.f'\ v'\ k'\ s_1') \in cont(v_1)(\Delta_1)$. So we let $\Delta_2 \geqslant \Delta_1$ and $(v, v') \in v_1(\Delta_2)$ and $(s_1, s_1') \in states(\Delta_2)$, and show that $(f\ v\ k\ s_1,\ f'\ v'\ k'\ s_1') \in R_{Ans}$. First, $(f\ v, f'\ v') \in comp(v_2)(\Delta_2)$ by assumption on (f, f'). By monotonicity of $cont(v_2)$, we have $(k, k') \in cont(v_2)(\Delta_2)$, and by assumption, $(s_1, s_1') \in states(\Delta_2)$. Therefore, it follows from the definition of $comp(v_2)(\Delta_2)$ that $(f\ v\ k\ s_1,\ f'\ v'\ k'\ s_1') \in R_{Ans}$. □

Definition 4.34. For every term environment $\Xi \vdash \Gamma$, every $\varphi \in \mathscr{T}^{\Xi}$ and every $\Delta \in \mathscr{W}$, let $[\![\Gamma]\!]_{\Xi}\varphi\,\Delta$ be the binary relation on $V^{\mathrm{dom}(\Gamma)}$ defined by

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

Definition 4.35. Two typed terms $\Xi \mid \Gamma \vdash t : \tau$ and $\Xi \mid \Gamma \vdash t' : \tau$ of the same type are *semantically related*, written $\Xi \mid \Gamma \models t \sim t' : \tau$, if for all $\varphi \in \mathscr{T}^{\Xi}$, all $\Delta \in \mathscr{W}$ and all $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi}\varphi\,\Delta$,

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

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. We will show 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. We will just give the details for some illustrative cases:

(1) If $\Gamma(x) = \tau$, then $\Xi \mid \Gamma \models x \sim x : \tau$.
Indeed, let $\varphi \in \mathscr{T}^{\Xi}$ and $\Delta \in \mathscr{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi}\varphi\,\Delta$ be given. Then $(\rho(x), \rho'(x)) \in [\![\tau]\!]_{\Xi}\varphi\,\Delta$. Therefore, by Proposition 4.33(1),

$$([\![x]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![x]\!]_{\mathrm{dom}(\Gamma)}\rho') = (\eta(\rho(x)),\ \eta(\rho'(x))) \in comp([\![\tau]\!]_{\Xi}\varphi)(\Delta),$$

as required.

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

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

By assumption,

$$([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in comp([\![\mu\alpha.\tau]\!]_{\Xi}\varphi)(\Delta).$$

So, by Proposition 4.33(2), it suffices to show that

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

To see this, we let $\Delta_1 \geqslant \Delta$ and $(v, v') \in [\![\mu\alpha.\tau]\!]_{\Xi}\varphi\,\Delta_1$ be given, and show that $(f\ v, f\ v') \in comp([\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi)(\Delta_1)$. By Corollary 4.31, $(v, v') = (in_{\mu}\,v_0,\ in_{\mu}\,v_0')$ for some $(v_0, v_0') \in$

$[\![\tau[\mu\alpha.\tau/\alpha]]\!]_\Xi\varphi\,\Delta_1$. But then by Proposition 4.33(1),

$$(f\,v, f\,v') = (\eta(v_0), \eta(v_0')) \in comp([\![\tau[\mu\alpha.\tau/\alpha]]\!]_\Xi\varphi)(\Delta_1),$$

as required.

(3) If $\Xi, \alpha \mid \Gamma \models t \sim t' : \tau$ and $\Xi \vdash \Gamma$, then $\Xi \mid \Gamma \models \Lambda\alpha.t \sim \Lambda\alpha.t' : \forall\alpha.\tau$.
Indeed, let $\varphi \in \mathscr{T}^\Xi$ and $\Delta \in \mathscr{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_\Xi\varphi\,\Delta$ be given. Recall that $[\![\Lambda\alpha.t]\!]_{\mathrm{dom}(\Gamma)}\rho = \eta(in_\forall([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho))$ and that $[\![\Lambda\alpha.t']\!]_{\mathrm{dom}(\Gamma)}\rho' = \eta(in_\forall([\![t']\!]_{\mathrm{dom}(\Gamma)}\rho'))$. We must therefore show that

$$\big(\eta(in_\forall([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho)),\ \eta(in_\forall([\![t']\!]_{\mathrm{dom}(\Gamma)}\rho'))\big) \in comp([\![\forall\alpha.\tau]\!]_\Xi\varphi)(\Delta).$$

By Proposition 4.33(1), it suffices to show that

$$\big(in_\forall([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho),\ in_\forall([\![t']\!]_{\mathrm{dom}(\Gamma)}\rho')\big) \in [\![\forall\alpha.\tau]\!]_\Xi\varphi\,\Delta.$$

We proceed according to the definition of $[\![\forall\alpha.\tau]\!]_\Xi$. We let $v \in \mathscr{T}$ be given and show that

$$([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in comp([\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v])(\Delta).$$

But this follows from the assumption that $\Xi, \alpha \mid \Gamma \models t \sim t' : \tau$ since Proposition 4.30(1) (weakening) gives $(\rho, \rho') \in [\![\Gamma]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v]$.

(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 $\varphi \in \mathscr{T}^\Xi$ and $\Delta \in \mathscr{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_\Xi\varphi\,\Delta$ be given. Recall that $[\![t\,[\tau_1]]\!]_{\mathrm{dom}(\Gamma)}\rho = [\![t]\!]_{\mathrm{dom}(\Gamma)}\rho \star g$ and $[\![t'\,[\tau_1]]\!]_{\mathrm{dom}(\Gamma)}\rho' = [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho' \star g$ where $g : V \to TV$ is given by

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

By assumption,

$$([\![t]\!]_{\mathrm{dom}(\Gamma)}\rho,\ [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho') \in comp([\![\forall\alpha.\tau]\!]_\Xi\varphi)(\Delta).$$

So, by Proposition 4.33(2), it suffices to show that

$$(g, g) \in [\![\forall\alpha.\tau]\!]_\Xi\varphi \xrightarrow{\Delta} comp([\![\tau_0[\tau_1/\alpha]]\!]_\Xi\varphi).$$

To see this, we let $\Delta_1 \geqslant \Delta$ and $(v, v') \in [\![\forall\alpha.\tau]\!]_\Xi\varphi\,\Delta_1$ be given, and then show that $(g\,v, g\,v') \in comp([\![\tau_0[\tau_1/\alpha]]\!]_\Xi\varphi)(\Delta_1)$. By the definition of $[\![\forall\alpha.\tau_0]\!]_\Xi$, we know that $(v, v') = (in_\forall\,c,\ in_\forall\,c')$ for some c and c' satisfying

$$(c, c') \in comp([\![\tau_0]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto v])(\Delta_1)$$

for all $v \in \mathscr{T}$. Now choose $v = [\![\tau_1]\!]_\Xi\varphi$. Proposition 4.30(2) (substitution) then gives

$$(g\,v, g\,v') = (c, c') \in comp([\![\tau_0[\tau_1/\alpha]]\!]_\Xi\varphi)(\Delta_1),$$

as required.

(5) If $\Xi \mid \Gamma \models t \sim t' : \tau$, then $\Xi \mid \Gamma \models \mathsf{ref}\ t \sim \mathsf{ref}\ t' : \mathsf{ref}\ \tau$.
Indeed, let $\varphi \in \mathscr{T}^\Xi$ and $\Delta \in \mathscr{W}$ and $(\rho, \rho') \in [\![\Gamma]\!]_\Xi\varphi\,\Delta$ be given. Recall that $[\![\mathsf{ref}\ t]\!]_{\mathrm{dom}(\Gamma)}\rho = [\![t]\!]_{\mathrm{dom}(\Gamma)}\rho \star \lambda v.\,alloc\,v$ and $[\![\mathsf{ref}\ t']\!]_{\mathrm{dom}(\Gamma)}\rho' = [\![t']\!]_{\mathrm{dom}(\Gamma)}\rho' \star \lambda v.\,alloc\,v$.

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

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

To see this, we let $\Delta_1 \geqslant \Delta$ and $(v, v') \in \llbracket \tau \rrbracket_\Xi \varphi\ \Delta_1$ be given, and then show that

$$(alloc\ v, alloc\ v') \in comp(\llbracket \mathsf{ref}\ \tau \rrbracket_\Xi \varphi)(\Delta_1).$$

We proceed according to the definition of *comp*. We let $\Delta_2 \geqslant \Delta_1$ and $(k, k') \in cont(\llbracket \mathsf{ref}\ \tau \rrbracket_\Xi \varphi)(\Delta_2)$ and $(s, s') \in states(\Delta_2)$ be given, and show that

$$(alloc\ v\ k\ s, alloc\ v'\ k'\ s') \in R_{Ans}\,. \tag{4.6}$$

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

$$alloc\ v\ k\ s = k\ \lambda_{l_0}\,(s[l_0 \mapsto v]) \tag{4.7}$$

$$alloc\ v'\ k'\ s' = k'\ \lambda_{l_0}\,(s'[l_0 \mapsto v'])\,. \tag{4.8}$$

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

$$i(\Delta_4(l_0)) = i(\Delta_3(l_0)) = i(i^{-1}(\llbracket \tau \rrbracket_\Xi \varphi)) = \llbracket \tau \rrbracket_\Xi \varphi\,.$$

Furthermore,

$$(s[l_0 \mapsto v],\ s'[l_0 \mapsto v']) \in states(\Delta_3)$$

since for $l \in \mathrm{dom}(\Delta_2)$, we have

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

by monotonicity of $i(\Delta_3(l)) \in \mathscr{W} \to_{mon} CURel(V)$, 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 $\llbracket \tau \rrbracket_\Xi \varphi$. Therefore, (4.7), (4.8) and the assumption that $(k, k') \in cont(\llbracket \mathsf{ref}\ \tau \rrbracket_\Xi \varphi)(\Delta_2)$ gives (4.6), as required.

(6) If $\Xi \mid \Gamma \models t \sim t' : \mathsf{ref}\ \tau$, then $\Xi \mid \Gamma \models\ !t \sim\ !t' : \tau$.
Indeed, let $\varphi \in \mathscr{T}^\Xi$ and $\Delta \in \mathscr{W}$ 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

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

Therefore, we let $\Delta_1 \geqslant \Delta$ and $(v, v') \in \llbracket \mathsf{ref}\ \tau \rrbracket_\Xi \varphi\ \Delta_1$ be given, and show that

$$(lookup\ v, lookup\ v') \in comp(\llbracket \tau \rrbracket_\Xi \varphi)(\Delta_1).$$

According to the definition of $[\![\mathsf{ref}\,\tau]\!]_\Xi\varphi = ref([\![\tau]\!]_\Xi\varphi)$, there are two cases:

(a) $v = v' = \lambda_l$ for some $l \in \mathrm{dom}(\Delta_1)$; or

(b) $v = v' = \lambda_l^{n+1}$ for some $n \in \omega$ and $l \in \mathrm{dom}(\Delta_1)$.

We will just consider the second case; the first is similar, but easier.

We proceed according to the definition of *comp*. So we let $\Delta_2 \geqslant \Delta_1$ and $(k, k') \in cont([\![\tau]\!]_\Xi\varphi)(\Delta_2)$ and $(s, s') \in states(\Delta_2)$ be given, and then show that

$$(lookup\ v\ k\ s,\ lookup\ v'\ k'\ s') \in R_{Ans}. \tag{4.9}$$

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

$$(s(l), s'(l)) \in i(\Delta_2(l))(\Delta_2).$$

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

$$i(\Delta_2(l))(\Delta_2) = i(\Delta_1(l))(\Delta_2) \stackrel{n}{=} [\![\tau]\!]_\Xi\varphi\,\Delta_2\,.$$

There are therefore two cases:

(a) $\pi_n(s(l)) = \pi_n(s'(l)) = \bot$, in which case, the definition of *lookup* gives

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

and we are done; or

(b) $(\pi_n(s(l)), \pi_n(s'(l))) = (\lfloor v_0 \rfloor, \lfloor v_0' \rfloor)$ where $(v_0, v_0') \in [\![\tau]\!]_\Xi\varphi\,\Delta_2$, in which case, the definition of *lookup* gives

$$(lookup\ v\ k\ s,\ lookup\ v'\ k'\ s') = (k\ v_0\ s,\ k'\ v_0'\ s'). \tag{4.10}$$

By assumption, $(k, k') \in cont([\![\tau]\!]_\Xi\varphi)(\Delta_2)$. Therefore, by the definition of *cont*, we have $(k\ v_0\ s,\ k'\ v_0'\ s') \in R_{Ans}$. We can then conclude (4.9) from (4.10), and we are done.

For many of the remaining cases, the core of the proof is to show that *comp* preserves logical relatedness. For example:

$$([\![t_1]\!]\rho, [\![t_1']\!]\rho') \in comp([\![\tau \to \tau']\!]_\Xi\varphi)(\Delta)\ \wedge\ ([\![t_2]\!]\rho, [\![t_2']\!]\rho') \in comp([\![\tau]\!]_\Xi\varphi)(\Delta) \implies ([\![t_1\ t_2]\!]\rho, [\![t_1'\ t_2']\!]\rho') \in comp([\![\tau']\!]_\Xi\varphi)(\Delta) \tag{4.11}$$

$$([\![t_1]\!]\rho, [\![t_1']\!]\rho') \in comp([\![\tau_1]\!]_\Xi\varphi)(\Delta)\ \wedge\ ([\![t_2]\!]\rho, [\![t_2']\!]\rho') \in comp([\![\tau_2]\!]_\Xi\varphi)(\Delta) \implies ([\![(t_1, t_2)]\!]\rho, [\![(t_1', t_2')]\!]\rho') \in comp([\![\tau_1 \times \tau_2]\!]_\Xi\varphi)(\Delta) \tag{4.12}$$

□

Corollary 4.37.

(1) If $\varnothing \mid \varnothing \vdash t : \tau$ is a closed term of type τ, then $[\![t]\!]_\varnothing\varnothing \neq error$.

(2) If $\varnothing \mid \varnothing \vdash t : \mathsf{int}$ is a closed term of type int, then $[\![t]\!]^{\mathrm{p}} \neq error_{Ans}$.

Proof.

(1) The theorem gives $(\llbracket t \rrbracket_\varnothing \varnothing, \llbracket t \rrbracket_\varnothing \varnothing) \in comp(\llbracket \tau \rrbracket_\varnothing \varnothing)(\varnothing)$. Now let $s_{init} \in S$ be the empty store and $k_0 \in K$ be the continuation that always gives the answer 0, that is, $k_0\, v\, s = \iota_1\, 0$ for all v and s. It follows immediately from the definitions that $(k_0, k_0) \in cont(\llbracket \tau \rrbracket_\varnothing \varnothing)(\varnothing)$ and that $(s_{init}, s_{init}) \in states(\varnothing)$. Therefore,

$$(\llbracket t \rrbracket_\varnothing \varnothing\, k_0\, s_{init}, \llbracket t \rrbracket_\varnothing \varnothing\, k_0\, s_{init}) \in R_{Ans}.$$

By the definition of R_{Ans}, we must then have $\llbracket t \rrbracket_\varnothing \varnothing\, k_0\, s_{init} \neq error_{Ans}$, which implies that $\llbracket t \rrbracket_\varnothing \varnothing \neq error$.

(2) Recall that $\llbracket t \rrbracket^{\mathrm{p}} = \llbracket t \rrbracket_\varnothing \varnothing\, k_{init}\, s_{init}$ where $s_{init} \in S$ is the empty store and where

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

It is easy to show that $(k_{init}, k_{init}) \in cont(\llbracket \mathsf{int} \rrbracket_\varnothing \varnothing)(\varnothing)$. Furthermore, as we have already argued, $(s_{init}, s_{init}) \in states(\varnothing)$.

The theorem gives $(\llbracket t \rrbracket_\varnothing \varnothing, \llbracket t \rrbracket_\varnothing \varnothing) \in comp(\llbracket \tau \rrbracket_\varnothing \varnothing)(\varnothing)$, which then implies that

$$(\llbracket t \rrbracket^{\mathrm{p}}, \llbracket t \rrbracket^{\mathrm{p}}) = (\llbracket t \rrbracket_\varnothing \varnothing\, k_{init}\, s_{init}, \llbracket t \rrbracket_\varnothing \varnothing\, k_{init}\, s_{init}) \in R_{Ans}\,.$$

Therefore $\llbracket t \rrbracket^{\mathrm{p}} \neq error_{Ans}$ by the definition of R_{Ans}. □

5. On the need for approximate locations

Consider the following tentative interpretation of reference types:

$$ref(v)(\Delta) = \{(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \wedge \forall \Delta_1 \geqslant \Delta.\, i(\Delta(l))(\Delta_1) = v(\Delta_1)\}\,.$$

As we argued in the discussion following the proof of Lemma 4.27, this function $ref(v)$ is not non-expansive and hence does not belong to the space $\mathscr{T}$ of semantic types. Accordingly, we have introduced approximate locations and settled for a more complicated interpretation of reference types.

Still, it might be that one could interpret semantic types and worlds differently, perhaps by solving a recursive equation in a different category so that we could have achieved a 'simple' interpretation of reference types.

In this section we show, in an abstract setup, and under some relatively mild assumptions, that such a simple interpretation is impossible. The proof works for an arbitrary set V, which is not required to be a domain, and there is no requirement that the elements λ_l are particular values such as integers. The intuitive idea of the proof is that semantic worlds (as expressed by a solution to the recursive equations between types and worlds) can in a certain sense be used to encode recursive types, and that recursive types and 'simple' reference types are incompatible.

Let V be a set and $GRel(V)$ be a set of subsets of $V \times V$ that is closed under finite intersections and contains the empty set. We think of $GRel(V)$ as the set of 'good' relations on V. For example in the context of this paper, V would be the universal predomain from Proposition 3.2 and $GRel(V)$ would be $CURel(V)$.

Now assume that $\hat{\mathscr{T}}$ is a set and that $i : \hat{\mathscr{T}} \to \mathscr{T}$ is a bijection into some subset $\mathscr{T}$ of $(Loc \to_{fin} \hat{\mathscr{T}}) \to_{mon} GRel(V)$. That is, $\mathscr{T}$ is a subset of those functions that are monotone with respect to the 'extension' order on $\mathscr{W} = Loc \to_{fin} \hat{\mathscr{T}}$ and the inclusion order on $GRel(V)$. We assume that $\mathscr{T}$ contains all the constant functions, and that for all $v_1 \in \mathscr{T}$ and $v_2 \in \mathscr{T}$ the monotone function $v_1 \wedge v_2$ given by

$$(v_1 \wedge v_2)(\Delta) = v_1(\Delta) \cap v_2(\Delta)$$

belongs to $\mathscr{T}$.

The idea is that $\mathscr{T}$ should be thought of as the set of 'good' monotone functions. In the concrete metric-space context, $\mathscr{T}$ consists of the non-expansive, monotone functions.

We now add the following assumption.

Assumption 5.1. There is an injective function $c : V \to V$ such that for every $l \in \omega$, the monotone function v_l given by

$$v_l(\Delta) = \begin{cases} \{(c(v), c(v')) \mid (v, v') \in i(\Delta(l))(\Delta)\} & \text{if } l \in \mathrm{dom}(\Delta) \\ \varnothing & \text{otherwise} \end{cases}$$

belongs to $\mathscr{T}$.

For every $R \in GRel(V)$, we define $c(R) = \{(c(v), c(v') \mid (v, v') \in R\}$. With this notation, the assumption implies that $v_l(\Delta) = c(i(\Delta(l))(\Delta))$ when $l \in \mathrm{dom}(\Delta)$. The idea is that v_l is a 'self-application' operator up to some function c, which could potentially be the identity function. We also assume that for every $R \in GRel(V)$, the relation $c(R)$ belongs to $GRel(V)$.

In the concrete model presented in this paper, we can choose $c = in_\mu$. Then the function v_l defined above is indeed non-expansive (and hence belongs to $\mathscr{T}$): Lemma A.1 says that if $\Delta =_n \Delta'$ for some $n > 0$, then $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$. The same lemma also says that if l belongs to these two domains, then $i(\Delta(l)) =_{n-1} i(\Delta'(l))$ and hence $i(\Delta(l))(\Delta) =_{n-1} i(\Delta'(l))(\Delta')$. Finally, by the definition of the projection functions, $in_\mu(i(\Delta)(\Delta)) =_n in_\mu(i(\Delta')(\Delta'))$, as desired.

Returning to the abstract case, we now show that v_l has the following 'universality' property.

Proposition 5.2. For every $v \in \mathscr{T}$ there exists $\Delta_0 \in \mathscr{W}$ such that

$$v_l(\Delta) = c(v(\Delta)) \qquad \text{for all } \Delta \geqslant \Delta_0.$$

Proof. Choose $\Delta_0 = [l \mapsto i^{-1}(v)]$. Then for all $\Delta \geqslant \Delta_0$, we have

$$v_l(\Delta) = c(i(\Delta(l))(\Delta)) = c(v(\Delta)).$$ □

Even though the proof of the proposition is about as simple as it could be, the proposition itself is in our opinion slightly counterintuitive: how can it possibly be that

every type $v \in \mathscr{T}$ 'agrees with v_l' somewhere? One answer is that the set of worlds $\mathscr{W}$ is not just some abstract preorder; it is a set that is so large that it includes $\mathscr{T}$ itself.

It follows that *all* endofunctions on $\mathscr{T}$ have 'fixed points up to c' in a certain sense.

Corollary 5.3. For every function F from $\mathscr{T}$ to $\mathscr{T}$ there exists some $v' \in \mathscr{T}$ and $\Delta_0 \in \mathscr{W}$ such that

$$v'(\Delta) = c(F(v')(\Delta)) \qquad \text{for all } \Delta \geqslant \Delta_0.$$

Furthermore, for every location $l \in \omega$, one can choose the v' and Δ_0 above such that $\mathrm{dom}(\Delta_0) = \{l\}$.

Proof. Choose $v = F(v_l)$ in the previous proposition. □

Using these results, we show that the simple interpretation of reference types is impossible:

Proposition 5.4. Let $\lambda_{(-)} : \omega \to V$ be an arbitrary, injective function. There is no function $ref : \mathscr{T} \to \mathscr{T}$ satisfying

$$ref(v)(\Delta) = \{(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \wedge \forall \Delta_1 \geqslant \Delta.\, i(\Delta(l))(\Delta_1) = v(\Delta_1)\}$$

for all $v \in \mathscr{T}$ and $\Delta \in \mathscr{W}$.

Proof. We assume that *ref* is such a function and derive a contradiction.

First, observe that every singleton relation $\{(\lambda_l, \lambda_l)\}$ belongs to $GRel(V)$: for an arbitrary $v \in \mathscr{T}$, we have $\{(\lambda_l, \lambda_l)\} = ref(v)([l \mapsto i^{-1}(v)])$, so $\{(\lambda_l, \lambda_l)\}$ is in the image of $ref(v) \in \mathscr{T}$. Then, by assumption on c, the relation $\{(c(\lambda_l), c(\lambda_l))\}$ also belongs to $GRel(V)$. Finally, the function $single(c(\lambda_l)) = \lambda\Delta.\{(c(\lambda_l), c(\lambda_l))\}$ belongs to $\mathscr{T}$ since we have assumed that $\mathscr{T}$ contains all the constant functions.

Let $l \in \omega$ be an arbitrarily chosen number and define $F : \mathscr{T} \to \mathscr{T}$ by $F(v) = ref(v \wedge single(c(\lambda_l)))$. Corollary 5.3 gives $v \in \mathscr{T}$ and $\Delta_0 \in \mathscr{W}$ such that

$$v(\Delta) = c(F(v)(\Delta))$$

for all $\Delta \geqslant \Delta_0$. By the last clause of the corollary, we can assume that $l \notin \mathrm{dom}(\Delta_0)$ (by choosing $\mathrm{dom}(\Delta_0) = \{l'\}$ for some $l' \neq l$). We now let $\Delta_0' = \Delta_0[l \mapsto i^{-1}(\lambda\Delta.\varnothing)]$ and aim to show that

$$(\lambda_l, \lambda_l) \notin F(v)(\Delta) \qquad \text{for all } \Delta \geqslant \Delta_0'. \tag{5.1}$$

To do this, we assume that $(\lambda_l, \lambda_l) \in F(v)(\Delta)$ for some $\Delta \geqslant \Delta_0'$. First, the definition of F and the assumption on *ref* give $i(\Delta(l))(\Delta) = (v \wedge single(c(\lambda_l))(\Delta)$ (here we use the fact that $\lambda_{(-)}$ is injective). Then, since $\Delta \geqslant \Delta_0'$,

$$\begin{aligned} v(\Delta) \cap \{(c(\lambda_l), c(\lambda_l))\} &= (v \wedge single(c(\lambda_l)))(\Delta) \\ &= i(\Delta(l))(\Delta) \\ &= i(\Delta_0'(l))(\Delta) \\ &= \varnothing, \end{aligned}$$

which means that $(c(\lambda_l), c(\lambda_l)) \notin v(\Delta)$. But $v(\Delta) = c(F(v)(\Delta))$ since $\Delta \geqslant \Delta'_0 \geqslant \Delta_0$, and we therefore conclude that $(\lambda_l, \lambda_l) \notin F(v)(\Delta)$, which gives a contradiction. This shows (5.1).

In particular, taking $\Delta = \Delta'_0$ in (5.1), we get that $(\lambda_l, \lambda_l) \notin F(v)(\Delta'_0)$. But then, by the definition of F and the assumption on *ref*, there is some $\Delta \geqslant \Delta'_0$ such that $i(\Delta'_0(l))(\Delta) \neq (v \wedge \mathit{single}(c(\lambda_l)))(\Delta)$. Since $i(\Delta'_0(l))(\Delta) = \varnothing$, this means that $(v \wedge \mathit{single}(c(\lambda_l)))(\Delta)$ is non-empty; in other words, $(c(\lambda_l), c(\lambda_l)) \in v(\Delta)$. Therefore, since $v(\Delta) = c(F(v)(\Delta))$ and since c is injective, $(\lambda_l, \lambda_l) \in F(v)(\Delta)$. But this contradicts (5.1). □

This proof only depends on the existence of a particular 'recursive type' $v(\Delta) = c(F(v)(\Delta))$. Accordingly, one can show a more syntactic variant of the proposition above that does *not* depend on Assumption 5.1: below we merely assume that $c : V \to V$ is some injective function satisfying $c(R) \in \mathit{GRel}(V)$ for all $R \in \mathit{GRel}(V)$. The semantic type v will then be obtained as the interpretation of a syntactic recursive type.

In order to express the syntactic counterpart of the semantic type v, we add finite intersection types $\tau_1 \wedge \tau_2$ to the language.

Proposition 5.5. Let $\lambda_{(-)} : \omega \to V$ be an arbitrary, injective function. There is no interpretation function $[\![-]\!]_{(-)}$, mapping types in context Ξ to functions from $\mathscr{T}^{\Xi}$ to $\mathscr{T}$, that satisfies

$$\begin{aligned}
[\![\alpha]\!]_{\Xi}\varphi(\Delta) &= \varphi(\alpha)(\Delta) \\
[\![\mathsf{ref}\,\tau]\!]_{\Xi}\varphi(\Delta) &= \{(\lambda_l, \lambda_l) \mid l \in \mathrm{dom}(\Delta) \wedge \forall \Delta_1 \geqslant \Delta.\, i(\Delta(l))(\Delta_1) = [\![\tau]\!]_{\Xi}\varphi(\Delta_1)\} \\
[\![\mu\alpha.\tau]\!]_{\Xi}\varphi(\Delta) &= \{(c(v), c(v')) \mid (v, v') \in [\![\tau[\mu\alpha.\tau/\alpha]]\!]_{\Xi}\varphi(\Delta)\} \\
[\![\tau_1 \wedge \tau_2]\!]_{\Xi}\varphi(\Delta) &= [\![\tau_1]\!]_{\Xi}\varphi(\Delta) \cap [\![\tau_2]\!]_{\Xi}\varphi(\Delta)
\end{aligned}$$

for all τ and Δ.

To show this, one essentially repeats the argument in the previous proof, but taking

$$v = [\![\mu\alpha.\,\mathsf{ref}\,(\alpha \wedge \beta)]\!]_{\beta}[\beta \mapsto \mathit{single}(c(\lambda_l))].$$

6. Examples

The model can be used to prove the equivalences in Birkedal *et al.* (2009, Section 5). More specifically, one can use the model to prove that some equivalences between different *functional* implementations of abstract data types are still valid in the presence of general references, and also prove some simple equivalences involving *imperative* abstract data types. (See Section 7 for more about extending the model to account properly for local state.) Here we will only sketch two of these examples, together with a 'non-example': an equivalence that cannot be shown because of the existence of approximated locations in the model.

6.1. *Encoding of existential types*

We use the usual encoding of existential types by means of universal types (Crary and Harper 2007): $\exists\alpha.\tau = \forall\beta.(\forall\alpha.\,\tau \to \beta) \to \beta$ where β does not occur free in τ. At the term level, we define $\mathsf{pack}(\tau', t) = \Lambda\beta.\,\lambda x.\,x\,[\tau']\,t$ where x does not occur free in t.

The natural proof principle for relatedness at existential types is then derivable.

Proposition 6.1. Let $\Xi \mid \Gamma \vdash \mathsf{pack}(\tau_1, t) : \exists\alpha.\tau$ and $\Xi \mid \Gamma \vdash \mathsf{pack}(\tau_2, t') : \exists\alpha.\tau$. Assume that $v \in \mathscr{T}$ satisfies the condition that for all $\Delta \in \mathscr{W}$, all $\varphi \in \mathscr{T}^{\Xi}$ and all $(\rho, \rho') \in [\![\Gamma]\!]\varphi(\Delta)$,

$$\big([\![t]\!]\rho,\ [\![t']\!]\rho'\big) \in comp([\![\tau]\!](\varphi[\alpha \mapsto v]))(\Delta).$$

Then

$$\Xi \mid \Gamma \models \mathsf{pack}(\tau_1, t) \sim \mathsf{pack}(\tau_2, t') : \exists\alpha.\tau.$$

Proof. By the congruence properties of semantic relatedness (see the proof of Theorem 4.36), it suffices to show

$$\Xi, \beta \mid \Gamma,\ x : \forall\alpha.\tau \to \beta \models x\,[\tau_1]\,t \sim x\,[\tau_2]\,t' : \beta.$$

(Here we have renamed β and x to ensure that they do not occur in Ξ and Γ, respectively.) To this end, we let $\Delta \in \mathscr{W}$ and $\varphi \in \mathscr{T}^{\Xi,\beta}$ and $(\rho, \rho') \in [\![\Gamma, x : \forall\alpha.\tau \to \beta]\!]\varphi(\Delta)$ be given, and then show

$$\big([\![x\,[\tau_1]\,t]\!]\rho,\ [\![x\,[\tau_2]\,t']\!]\rho'\big) \in comp(\varphi(\beta))(\Delta). \tag{6.1}$$

Since ρ and ρ' are related, we have $\rho(x) = in_\forall\, c$ and $\rho'(x) = in_\forall\, c'$ for some c and c' with $(in_\forall\, c,\ in_\forall\, c') \in [\![\forall\alpha.\tau \to \beta]\!]\varphi(\Delta)$. The interpretation of universal types then gives

$$\big([\![x\,[\tau_1]]\!]\rho,\ [\![x\,[\tau_2]]\!]\rho'\big) = (c, c') \in comp([\![\tau \to \beta]\!](\varphi[\alpha \mapsto v]))(\Delta). \tag{6.2}$$

The assumption of the proposition (and weakening with respect to β and x) gives

$$\big([\![t]\!]\rho,\ [\![t']\!]\rho'\big) \in comp([\![\tau]\!](\varphi[\alpha \mapsto v]))(\Delta). \tag{6.3}$$

Then, using (4.11), we get that (6.2) and (6.3) imply (6.1). □

6.2. *Example: imperative counter module*

Here we use the encoding of existential types presented in the previous section. The type

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

can be used to model imperative counter modules: the idea is that a value of type τ_{m} consists of some hidden type α, which is 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, that is, closed terms of type τ_{m}:

$$J = \mathsf{pack}(\mathsf{ref\ int}, I)$$
$$J' = \mathsf{pack}(\mathsf{ref\ int}, I')$$

where

$$I = (\lambda x.\,\mathsf{ref}\ 0,\ \lambda x.\ x := !x + 1,\ \lambda x.\,!x)$$
$$I' = (\lambda x.\,\mathsf{ref}\ 0,\ \lambda x.\ x := !x - 1,\ \lambda x.\,-(!x).$$

We now show by parametricity reasoning, that is, by exploiting the universal quantification in the interpretation of universal types, that $\varnothing \mid \varnothing \models J \sim J' : \tau_{\mathrm{m}}$.

We use the abbreviation $\tau = (1 \to \alpha) \times (\alpha \to 1) \times (\alpha \to \mathsf{int})$. By Proposition 6.1 it suffices to find $v \in \mathscr{T}$ such that $(\llbracket I \rrbracket, \llbracket I' \rrbracket) \in comp(\llbracket \tau \rrbracket[\alpha \to v])(\Delta)$ for all Δ.

Let $R = \{(in_Z(n), in_Z(-n)) \mid n \in \omega\}$ and choose $v = ref(\lambda\Delta.R)$. That is, we choose v to be the semantic type of locations that point to R-related values. Now let $\Delta \in \mathscr{W}$ be given. By (4.12), it suffices to show

$$(\llbracket \lambda x.\,\mathsf{ref}\ 0 \rrbracket,\ \llbracket \lambda x.\,\mathsf{ref}\ 0 \rrbracket) \in comp(\llbracket 1 \to \alpha \rrbracket[\alpha \mapsto v])(\Delta) \tag{6.4}$$
$$(\llbracket \lambda x.\ x := !x + 1 \rrbracket,\ \llbracket \lambda x.\ x := !x - 1 \rrbracket) \in comp(\llbracket \alpha \to 1 \rrbracket[\alpha \mapsto v])(\Delta) \tag{6.5}$$
$$(\llbracket \lambda x.\,!x \rrbracket,\ \llbracket \lambda x.\,-!x \rrbracket) \in comp(\llbracket \alpha \to \mathsf{int} \rrbracket[\alpha \mapsto v])(\Delta). \tag{6.6}$$

In all three cases we use the first part of Proposition 4.33. Then, (6.5) and (6.6) follow straightforwardly from the definitions given our choice of v.

To show (6.4), we let $\eta(in_{\to} f) = \llbracket \lambda x.\,\mathsf{ref}\ 0 \rrbracket$ and show that

$$(in_{\to} f,\ in_{\to} f) \in \llbracket 1 \to \alpha \rrbracket[\alpha \mapsto v](\Delta). \tag{6.7}$$

Let $\Delta_1 \geqslant \Delta$ and $(v, v') \in \llbracket 1 \rrbracket(\Delta)$ be given. We must now show that $(f(v), f(v')) \in comp(v)(\Delta_1)$. To do this, we unfold the definition of $comp$: let $\Delta_2 \geqslant \Delta_1$ and $(k, k') \in cont(v)(\Delta_2)$ and $(s, s') \in states(\Delta_2)$ be given. We must show that

$$(f(v)\,k\,s,\ f(v')\,k'\,s') \in R_{Ans}.$$

By the interpretation of allocation, $f(v)\,k\,s = k\,\lambda_l\,(s[l \mapsto in_Z 0])$ where l is the smallest number such that $l \notin \mathrm{dom}(s)$. Since $(s, s') \in states(\Delta_2)$, we have $\mathrm{dom}(s) = \mathrm{dom}(s') = \mathrm{dom}(\Delta_2)$, and, therefore, $f(v')\,k'\,s' = k'\,\lambda_l\,(s'[l \mapsto in_Z 0])$. (That is, the new location is the same one as above.) Since $(k, k') \in cont(v)(\Delta_2)$, it therefore suffices to find $\Delta_3 \geqslant \Delta_2$ such that

$$(\lambda_l, \lambda_l) \in v(\Delta_3) \tag{6.8}$$
$$(s[l \mapsto in_Z 0],\ s'[l \mapsto in_Z 0]) \in states(\Delta_3). \tag{6.9}$$

Choose $\Delta_3 = \Delta_2[l \mapsto i^{-1}(\lambda\Delta.R)]$. Then (6.8) follows directly from the fact that $v = ref(\lambda\Delta.R)$: indeed, for all $\Delta_4 \geqslant \Delta_3$, we have

$$i(\Delta_4(l))(\Delta_4) = R = (\lambda\Delta.R)(\Delta_4).$$

To show (6.9), first it is clear that

$$((s[l \mapsto in_Z 0])(l),\ (s'[l \mapsto in_Z 0])(l)) = (in_Z 0, in_Z 0) \in i(\Delta_3(l))(\Delta_3).$$

Then, for $l' \neq l$, the fact that $(s, s') \in \mathit{states}(\Delta_2)$ (and monotonicity) gives

$$\begin{aligned}((s[l \mapsto in_Z(0)])(l'),\, (s'[l \mapsto in_Z(0)])(l')) = (s(l'), s'(l')) &\in i(\Delta_2(l))(\Delta_2)\\ &= i(\Delta_3(l))(\Delta_2)\\ &\subseteq i(\Delta_3(l))(\Delta_3).\end{aligned}$$

We can now conclude that (6.9) holds, and hence that $\varnothing \mid \varnothing \models J \sim J' : \tau_{\mathrm{m}}$.

6.3. *Example: local references and closures*

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

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

where the $\mathtt{let}\ldots\mathtt{in}$ construct is syntactic sugar for a β-redex in the usual way. Both M and M' are closed terms of type τ_{lr}. By 'store parametricity' reasoning, that is, by exploiting the universal quantification over all larger worlds in the definition of *cont*, one can show that $\varnothing \mid \varnothing \models M \sim M' : \tau_{\mathrm{lr}}$.

In a little more detail, let $\eta(in_{\to} f) = [\![M]\!]$ and $\eta(in_{\to} f') = [\![M']\!]$. We let $\Delta \in \mathscr{W}$ be given and show that $(in_{\to} f, in_{\to} f') \in [\![\tau_{\mathrm{lr}}]\!](\Delta)$. To do this, we let $\Delta_1 \geqslant \Delta$ and $(v, v') \in [\![1]\!](\Delta_1)$ be given and show that

$$(f(v), f'(v')) \in \mathit{comp}([\![(1 \to 1) \times (1 \to \mathtt{int})]\!])(\Delta_1). \tag{6.10}$$

We unfold the definition of *comp*: let $\Delta_2 \geqslant \Delta_1$ and

$$(k, k') \in \mathit{cont}([\![(1 \to 1) \times (1 \to \mathtt{int})]\!])(\Delta_2)$$

and $(s, s') \in \mathit{states}(\Delta_2)$ be given. We must now show that

$$(f(v)\, k\, s,\; f'(v')\, k'\, s') \in R_{\mathit{Ans}}. \tag{6.11}$$

Let $l \in \omega$ be the least number not in $\mathrm{dom}(\Delta_2)$. Then

$$\begin{aligned}f(v)\, k\, s &= k\, (in_{\times}(in_{\to} g_1, in_{\to} g_2))\, (s[l \mapsto in_Z(0)])\\ f'(v')\, k'\, s' &= k'\, (in_{\times}(in_{\to} g_1', in_{\to} g_2'))\, (s'[l \mapsto in_Z(0)])\end{aligned}$$

where

$$\eta(in_{\to} g_1) = [\![\lambda y : 1.\, r \;:= !r + 1]\!][r \mapsto \lambda_l]$$

and

$$\eta(in_{\to} g_2) = [\![\lambda y : 1.\, !r]\!][r \mapsto \lambda_l],$$

and similarly for g_1' and g_2'. In order to show (6.11), it therefore suffices to find $\Delta_3 \geqslant \Delta_2$ such that

$$(in_{\to} g_1,\ in_{\to} g_1') \in [\![1 \to 1]\!](\Delta_3)$$
$$(in_{\to} g_2,\ in_{\to} g_2') \in [\![1 \to \mathtt{int}]\!](\Delta_3)$$
$$(s[l \mapsto in_Z(0)],\ s'[l' \mapsto in_Z(0)]) \in states(\Delta_3).$$

It is not hard to see that these conditions can be satisfied by choosing $\Delta_3 = \Delta_2[l \mapsto i^{-1}(\lambda\Delta.R)]$ where $R = \{(in_Z(n), in_Z(-n)) \mid n \in \omega\}$ as in the previous example.

6.4. *Example: references to the empty type*

Consider the two terms $K = \lambda x.2$ and $K' = \lambda x.3$ of type $\mathsf{ref}\,0 \to \mathsf{int}$. Given a standard operational semantics for the language, a simple bisimulation-style argument should suffice to show that K and K' are contextually equivalent: no reference cell can ever contain a value of type 0, and therefore neither function can ever be applied. However, as we will show next, the equivalence $\varnothing \mid \varnothing \models K \sim K' : \mathsf{ref}\,0 \to \mathsf{int}$ does not hold. Briefly, the reason for this is the existence of approximate locations in the model.

First, by the congruence property of semantic relatedness (Theorem 4.36), if $\varnothing \mid \varnothing \models K \sim K' : \mathsf{ref}\,0 \to \mathsf{int}$ holds, then so does $\varnothing \mid x : \mathsf{ref}\,0 \models K\,x \sim K'\,x : \mathsf{int}$. Hence it suffices to show that the latter does not hold.

Let $R \in CURel(V)$ be the singleton relation $\{(in_1 *, in_1 *)\}$. (Any other non-empty relation in $CURel(V)$ would also do.) Also, let $l \in \omega$ be an arbitrary location and define $\Delta = [l \mapsto i^{-1}(\lambda\Delta'.R)]$. Then

$$(\lambda_l^1, \lambda_l^1) \in [\![\mathsf{ref}\,0]\!](\Delta) \tag{6.12}$$

since $l \in \mathrm{dom}(\Delta)$ and $i(\Delta(l))(\Delta_1) \overset{0}{=} [\![0]\!](\Delta_1)$ holds vacuously for all Δ_1. Now let $\rho = [x \mapsto \lambda_l^1]$, so $(\rho, \rho) \in [\![x : \mathsf{ref}\,0]\!](\Delta)$. We claim that

$$\big([\![K\,x]\!]\rho, [\![K'\,x]\!]\rho\big) \notin comp([\![\mathsf{int}]\!])(\Delta), \tag{6.13}$$

which shows that $\varnothing \mid x : \mathsf{ref}\,0 \models K\,x \sim K'\,x : \mathsf{int}$ does not hold. Indeed, $[\![K\,x]\!]\rho = \eta(in_Z 2)$ while $[\![K'\,x]\!]\rho = \eta(in_Z 3)$, so

$$\big([\![K\,x]\!]\rho\, k_{init}\ [l \mapsto in_1 *],\ [\![K'\,x]\!]\rho\, k_{init}\ [l \mapsto in_1 *]\big) = \big(\lfloor \iota_1(2) \rfloor, \lfloor \iota_1(3) \rfloor\big) \notin Ans,$$

which shows (6.13). We conclude that $\varnothing \mid x : \mathsf{ref}\,0 \models K\,x \sim K'\,x : \mathsf{int}$ does not hold and hence that $\varnothing \mid \varnothing \models K \sim K' : \mathsf{ref}\,0 \to \mathsf{int}$ does not hold.

Looking back, the problem is indeed the presence of approximate locations: equation (6.12) intuitively implies that $\mathsf{ref}\,0$ is 'inhabited', while the operational equivalence between K and K' expresses exactly the fact that $\mathsf{ref}\,0$ is not inhabited.

7. Related work

Metric spaces

As we have already mentioned, the metric-space structure on uniform relations over universal domains is well known (Abadi and Plotkin 1990; Amadio 1991; Amadio and

Curien 1998; Cardone 1989; MacQueen *et al.* 1986). The inverse-limit method for solving recursive domain equations was first adapted to metric spaces by America and Rutten (America and Rutten 1989).

Approximate locations

Approximate (or 'semantic') locations were first introduced in Birkedal *et al.* (2009), which contains an adequacy proof with respect to an operational semantics, and also an entirely different, quasi-syntactic, interpretation of open types. Here we have instead presented what is in some ways a more natural interpretation that results from solving a recursive metric-space equation, thereby obtaining a proper universe of semantic types. Open types are then interpreted in the expected way, that is, as maps from environments of semantic types to semantic types.

Game semantics

Abramsky *et al.* (1998) presented a game semantics for a language with general reference types. References are, informally speaking, modelled as pairs of 'get' and 'set' functions, following an idea by Reynolds. This model of references allows a compositional interpretation of reference types, as in this paper. In other words, the meaning of the reference type $\mathsf{ref}\,\tau$ is completely determined by the meaning of the type τ. On the other hand, modelling references as pairs of functions introduces the so-called 'bad variable' problem: there is no guarantee that a pair of functions inhabiting the reference type behaves like a reference cell, in the sense that, for example, calling 'get' returns the most recent value given to 'set'. Another consequence is that the model cannot be used to interpret the equality testing of references.

Our model has a related, but milder variant of the bad variable problem. As illustrated by the example in Section 6.4, there is no guarantee that an inhabitant of a semantic reference type behaves like an actual location in terms of observations performed by lookup and assignment. It is, however, guaranteed that every inhabitant corresponds to an actual reference cell, and that lookup and assignment operations are domain-theoretic *approximations* of the 'real' operations on that reference cell. The model can easily be used to interpret equality testing of references. On the other hand, Abramsky *et al.*'s model is fully abstract for a language that includes a 'bad-variable constructor', while our model is far from fully abstract.

Laird (2007) presents a fully abstract trace semantics for a language with general references. This model, which has a strong game-semantics 'flavour', does not exhibit the bad variable problem.

Game semantics has been used to model recursive types (McCusker 2000) and impredicative polymorphism (Abramsky and Jagadeesan 2005). The latter model features a very different approach to parametricity that allows for non-relational reasoning about terms of polymorphic type. We are not aware of any game semantics that models the combination of impredicative polymorphism, general reference types and recursive types that we have considered in this article.

Step-indexing and recursively defined worlds

The technique of solving a metric-space equation in order to build a Kripke-style model, as presented in this paper, has subsequently been used by Schwinghammer *et al.* in their model of separation logic for a language with higher-order store (Schwinghammer *et al.* 2009), and in their more recent extension (with F. Pottier) in Schwinghammer *et al.* (2010), which includes an 'anti-frame rule' for local reasoning about state.

The fundamental circularity between worlds and types in realisability-style possible-worlds models of polymorphism and general references was observed in Ahmed (2004, page 62) in the setting of operational semantics (and for unary relations). Rather than solve a recursive equation, her solution was to stratify worlds and types into different levels, represented by natural numbers. So-called step-indexing is used in the definition to ensure that a stratified variant of the fundamental theorem holds. These stratified worlds and types are somewhat analogous to the approximants of recursive-equation solutions that are employed in the inverse-limit method. The main advantage in 'going to the limit' of the approximations and working with an actual solution (as we do here) is that approximation information is then not ubiquitous in definitions and proofs; by analogy, the only 'approximation information' in our model is in the interpretation of references and in the requirement that user-supplied relations are uniform[†].

In recent work, the authors of the current paper, and others, have shown that the metric-space approach presented here can be used with step-indexing to provide a high-level approach to giving operational semantics that involve recursively defined structures (Birkedal *et al.* 2010a). In particular, we have presented a model of Charguéraud and Pottier's type-and-capability system for an ML-like higher-order language, together with an operational model of the logic of Schwinghammer *et al.* (2009).

Ahmed *et al.* 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 (Ahmed *et al.* 2009): this allows for proofs of much more advanced equivalences involving local state. As we described in the introduction, we have recently shown that our approach extends to this style of worlds (Birkedal *et al.* 2010b). In future work, we also plan to investigate local-state parameters in the style of Bohr and Birkedal (2006). By contrast, in the present paper, our intention has just been to present the fundamental ideas behind Kripke logical relations over recursively defined sets of worlds as required for the semantic modelling of parametric polymorphism, recursive types, and general references.

Appendix A. Concrete descriptions of some metric spaces

Recall that the set $\mathscr{T} = \mathscr{W} \to_{mon} CURel(V)$ is a subset of the underlying set of the exponential $\mathscr{T} = \mathscr{W} \to CURel(V)$ in CBUlt, and as such is given a natural metric by Proposition 4.18. We call this metric d_1 in the following, and let d_2 be the metric on $\widehat{\mathscr{T}}$ associated with the isomorphism $i : \widehat{\mathscr{T}} \to 1/2 \cdot \mathscr{T}$ obtained from Theorem 4.20. The fact

[†] We plan to carry out a more formal comparison in future work.

that i is an isomorphism then implies that

$$d_2(\widehat{v},\widehat{v}') = 1/2 \cdot d_1(i(\widehat{v}), i(\widehat{v}')) \tag{A.1}$$

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

Lemma A.1. For $\Delta, \Delta' \in \mathscr{W}$, we have that $\Delta \stackrel{n}{=} \Delta'$ if and only if either

(1) $n = 0$; or
(2) $n > 0$ and $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and

$$i\,(\Delta(l))\Delta_0 \stackrel{n-1}{=} i\,(\Delta'(l))\Delta_0$$

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

(The '$n-1$' comes from the '$1/2$' on the right-hand side of the isomorphism (4.4).)

Proof.

Only if: We assume that $\Delta \stackrel{n}{=} \Delta'$ and that $n > 0$ and show that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and that $i\,(\Delta(l))\Delta_0 \stackrel{n-1}{=} i\,(\Delta'(l))\Delta_0$ for all $l \in \mathrm{dom}(\Delta)$ and all $\Delta_0 \in \mathscr{W}$. Since $d(\Delta,\Delta') \leqslant 2^{-n} < 1$, the definition of the metric on $\mathscr{W}$ implies that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and that $d(\Delta,\Delta') = \max\{d_2(\Delta(l),\Delta'(l)) \mid l \in \mathrm{dom}(\Delta)\}$. Now let $l \in \mathrm{dom}(\Delta)$ and $\Delta_0 \in \mathscr{W}$. First, $d_2(\Delta(l),\Delta'(l)) \leqslant d(\Delta,\Delta') \leqslant 2^{-n}$, and (A.1) therefore gives

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

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

$$d(i(\Delta(l))\Delta_0,\ i(\Delta'(l))(\Delta_0)) \leqslant d_1(i(\Delta(l)), i(\Delta'(l))) \leqslant 2^{-(n-1)},$$

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

If: The relation $\Delta \stackrel{0}{=} \Delta'$ holds for any Δ and Δ' since $\mathscr{W}$ is 1-bounded. So we assume that $n > 0$, that $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and that $i\,(\Delta(l))\Delta_0 \stackrel{n-1}{=} i\,(\Delta'(l))\Delta_0$ for all $l \in \mathrm{dom}(\Delta)$ and all $\Delta_0 \in \mathscr{W}$. Then for every $l \in \mathrm{dom}(\Delta)$, we have $d_1(i\,(\Delta(l)), i(\Delta'(l))) \leqslant 2^{-(n-1)}$ by the definition of d_1, and hence

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

by (A.1). Therefore, $d(\Delta,\Delta') \leqslant 2^{-n}$, that is, $\Delta \stackrel{n}{=} \Delta'$. □

Lemma A.2. Let $(A,(\varpi_n)_{n\in\omega})$ be a uniform cpo. We use the abbreviation

$$CURel(A) = CURel(A,(\varpi_n)_{n\in\omega})$$

and consider the metrics on $CURel(A)$ and $\mathscr{W} \to_{mon} CURel(A)$ given by Proposition 4.18:

(1) For $R, S \in CURel(A)$, we have $R \stackrel{n}{=} S$ if and only if $\varpi_n \in R \to S_\perp$ and $\varpi_n \in S \to R_\perp$.
(2) For $v, v' \in \mathscr{W} \to_{mon} CURel(A)$, we have $v \stackrel{n}{=} v'$ if and only if $v(\Delta_0) \stackrel{n}{=} v'(\Delta_0)$ for all $\Delta_0 \in \mathscr{W}$.
(3) A function v from $\mathscr{W}$ to $CRel(A)$ belongs to $\mathscr{W} \to_{mon} CURel(A)$ if and only if it satisfies the following two conditions for all $\Delta, \Delta' \in \mathscr{W}$:
   (a) If $\Delta \leqslant \Delta'$, then $v(\Delta) \subseteq v(\Delta')$.
   (b) If $\Delta \stackrel{n}{=} \Delta'$, then $\varpi_n \in v(\Delta) \to v(\Delta')_\perp$.

Lemma A.3. Let $(D, (\varpi_n)_{n\in\omega})$ be a uniform cppo. We use the abbreviation

$$AURel(D) = AURel(D, (\varpi_n)_{n\in\omega})$$

and consider the metrics on $AURel(D)$ and $\mathscr{W} \to_{mon} AURel(D)$ given by Propositions 4.14 and 4.15.

(1) For $R, S \in AURel(D)$, we have $R \overset{n}{=} S$ if and only if $\varpi_n \in R \to S$ and $\varpi_n \in S \to R$.
(2) For $\xi, \xi' \in \mathscr{W} \to_{mon} AURel(D)$, we have $\xi \overset{n}{=} \xi'$ if and only if $\xi(\Delta_0) \overset{n}{=} \xi'(\Delta_0)$ for all $\Delta_0 \in \mathscr{W}$.
(3) A function ξ from $\mathscr{W}$ to $ARel(D)$ belongs to $\mathscr{W} \to_{mon} AURel(D)$ if and only if it satisfies the following two conditions for all $\Delta, \Delta' \in \mathscr{W}$:
   (a) If $\Delta \leqslant \Delta'$, then $\xi(\Delta) \subseteq \xi(\Delta')$.
   (b) If $\Delta \overset{n}{=} \Delta'$, then $\varpi_n \in \xi(\Delta) \to \xi(\Delta')$.

Acknowledgements

We are grateful to the anonymous referees for comments and suggestions. We would also like to thank Nick Benton, Lennart Beringer, Stephen Brookes, Martin Hofmann, Andrew Kennedy, Paul Blain Levy, Rasmus Møgelberg, François Pottier, Bernhard Reus, John Reynolds, Jan Schwinghammer and Hongseok Yang for comments and discussions regarding this work.

References

Abadi, M. (2000) Top-top-closed relations and admissibility. *Mathematical Structures in Computer Science* **10** (3) 313–320.

Abadi, M. and Plotkin, G. D. (1990) A per model of polymorphism and recursive types. In: *Proceedings of LICS* 355–365.

Abramsky, S., Honda, K. and McCusker, G. (1998) A fully abstract game semantics for general references. In: *Proceedings of LICS* 334–344.

Abramsky, S. and Jagadeesan, R. (2005) A game semantics for generic polymorphism. *Annals of Pure and Applied Logic* **133** (1–3) 3–37.

Ahmed, A. (2004) *Semantics of Types for Mutable State*, Ph.D. thesis, Princeton University.

Ahmed, A., Dreyer, D. and Rossberg, A. (2009) State-dependent representation independence. In: *Proceedings of POPL* 340–353.

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

Amadio, R. M. and Curien, P.-L. (1998) *Domains and Lambda-Calculi*, Cambridge University Press.

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

Baier, C. and Majster-Cederbaum, M. E. (1997) The connection between initial and unique solutions of domain equations in the partial order and metric approach. *Formal Aspects of Computing* **9** (4) 425–445.

Benton, N. and Hur, C.-K. (2009) Biorthogonality, step-indexing and compiler correctness. In: *Proceedings of ICFP* 97–108.

Benton, N. and Leperchey, B. (2005) Relational reasoning in a nominal semantics for storage. In: Proceedings of TLCA. *Springer-Verlag Lecture Notes in Computer Science* **3461** 86–101.

Birkedal, L., Reus, B., Schwinghammer, J., Støvring, K., Thamsborg, J. and Yang, H. (2010a) Step-indexed kripke models over recursive worlds. Submitted for publication. Available at `http://www.itu.dk/people/birkedal/papers/step-recworld-conf.pdf`.

Birkedal, L., Støvring, K. and Thamsborg, J. (2009) Relational parametricity for references and recursive types. In: *Proceedings of TLDI* 91–104.

Birkedal, L., Støvring, K. and Thamsborg, J. (2010b) A relational realizability model for higher-order stateful ADTs. Submitted for publication. Available at `http://www.itu.dk/people/birkedal/papers/relrmhoadt.pdf`.

Bohr, N. and Birkedal, L. (2006) Relational reasoning for recursive types and references. In: Proceedings of APLAS. *Springer-Verlag Lecture Notes in Computer Science* **4279** 79–96.

Cardone, F. (1989) Relational semantics for recursive types and bounded quantification. In: Proceedings of ICALP. *Springer-Verlag Lecture Notes in Computer Science* **372** 164–178.

Crary, K. and Harper, R. (2007) Syntactic logical relations for polymorphic and recursive types. *Electronic Notes in Theoretical Computer Science* **172** 259–299.

Laird, J. (2007) A fully abstract trace semantics for general references. In: Proceedings of ICALP. *Springer-Verlag Lecture Notes in Computer Science* **4596** 667–679.

Levy, P. B. (2006) Call-by-push-value: Decomposing call-by-value and call-by-name. *Higher-Order and Symbolic Computation* **19** (4) 377–414.

MacQueen, D. B., Plotkin, G. D. and Sethi, R. (1986) An ideal model for recursive polymorphic types. *Information and Control* **71** (1/2) 95–130.

McCusker, G. (2000) Games and full abstraction for FPC. *Information and Computation* **160** (1–2) 1–61.

Moggi, E. (1991) Notions of computation and monads. *Information and Computation* **93** 55–92.

Petersen, R. L., Birkedal, L., Nanevski, A. and Morrisett, G. (2008) A realizability model for impredicative Hoare type theory. In: Proceedings of ESOP. *Springer-Verlag Lecture Notes in Computer Science* **4960** 337–352.

Pierce, B. C. (2002) *Types and Programming Languages*, The MIT Press.

Pitts, A. M. (1996) Relational properties of domains. *Information and Computation* **127** 66–90.

Pitts, A. M. (1998) Existential types: Logical relations and operational equivalence. In: Proceedings of ICALP. *Springer-Verlag Lecture Notes in Computer Science* **1443** 309–326.

Plotkin, G. D. and Power, J. (2004) Computational effects and operations: An overview. *Electronic Notes in Theoretical Computer Science* **73** 149–163.

Schwinghammer, J., Birkedal, L., Reus, B. and Yang, H. (2009) Nested Hoare triples and frame rules for higher-order store. In: Proceedings of CSL. *Springer-Verlag Lecture Notes in Computer Science* **5771** 440–454.

Schwinghammer, J., Yang, H., Birkedal, L., Pottier, F. and Reus, B. (2010) A semantic foundation for hidden state. In: Proceedings of FOSSACS. *Springer-Verlag Lecture Notes in Computer Science* **6014** 2–17.

Smyth, M. B. (1992) Topology. In: Abramsky, S., Gabbay, D. and Maibaum, T. S. E. (eds.) *Handbook of Logic in Computer Science*, Oxford University Press.

Smyth, M. B. and Plotkin, G. D. (1982) The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing* **11** (4) 761–783.

Wagner, K. R. (1994) *Solving Recursive Domain Equations with Enriched Categories*, Ph.D. thesis, Carnegie Mellon University.

Winskel, G. (1993) *The Formal Semantics of Programming Languages*, Foundation of Computing Series, The MIT Press.