

Kripke Models over Recursively Defined Metric Worlds: Steps and Domains

Lars Birkedal

IT University of Copenhagen
birkedal@itu.dk

Kristian Støvring

IT University of Copenhagen
kss@itu.dk

Jacob Thamsborg

IT University of Copenhagen
thamsborg@itu.dk

Abstract

We show that models of higher-order store phenomena naturally can be given as Kripke models over worlds that are recursively defined in a category of metric spaces. It leads to a unification of methods based on classical domain theory and on step-indexed operational models. We show that our metric approach covers a wide range of step-indexed models, by demonstrating how it can be specialized to Hobor et. al.'s recent indirection theory, and by developing a new step-indexed model of separation logic for higher-order store.

1 Introduction

Over the last decade, there has been a lot of research on semantic models for reasoning about advanced programming language features involving recursive structures arising from various forms of *higher-order store*, see, e.g. [12, 19, 23, 25, 29, 30]. Many proposed methods have been based on either traditional domain theory or on more recent step-indexed models [2–4, 8, 9]. In this paper, we argue that the essence of these models is that they can be seen as Kripke models over recursively defined sets of worlds. Indeed, we show how to define such worlds using appropriate recursively-defined metric spaces and, moreover, show how this method applies both to domain-theoretic models and to step-indexed models, thus achieving a unification of methods. In earlier work, we have used solutions to recursive metric-space equations in connection with domain-theoretic models [18], so in this paper, we focus mostly on step-indexed models. In particular, we show how our metric approach can be specialized to Hobor et. al.'s recent abstract description of step-indexed models [24] and argue why it is useful to take the metric viewpoint we suggest. The latter is done, in part, by presenting a step-indexed model of a separation logic for higher-order store [32], the soundness of which involves the use of a recursively defined operation on the recursively-defined set of worlds.

Higher-order Store We use “higher-order store” loosely to refer to programming language features that involve some form of dynamic allocation of data whose type / specification depends on the types / specifications of already stored data. Thus higher-order store can, e.g., describe the ability to dynamically allocate heap storage and store code directly in the heap; C function pointers; ML references; but also dynamically allocated locks that protect resource invariants that depend on already allocated locks’ resource invariants. For expressiveness, type systems and logics for higher-order store often involve some form of recursive types / specifications, and also some form of universal quantification over types (impredicative polymorphism) or specifications. The semantic methods we present scale well to these features (see, e.g., [18]), but they are not at the heart of the challenge of modeling higher-order store, so we shall not dwell too long on them in this paper, but just sketch how the methods apply.

Semantic Models of What Semantic models of higher-order store can, among other things, be used to show soundness of type systems and logics for reasoning about programs. The latter can involve Hoare-style logics for reasoning about a single program or logics for relational reasoning about equality of programs. Our methodology applies to all of these, but relational reasoning involves a host of other mostly orthogonal issues, so we focus on models for type systems and logics for reasoning about a single program using unary predicates instead of relations, except for some discussion in the related work section.

2 Introductory Example: ML references

By way of introduction of our general setup, let us consider how to model a programming language with impredicative polymorphism and general ML-like references; i.e., an extension of the polymorphic lambda calculus with a standard call-by-value operational semantics. We first describe the general idea at an intuitive level and then, in the following two subsections, we explain how to realize the

general idea in a domain-theoretic setting (based on an adequate denotational semantics of the programming language) and in a step-indexed setting (based purely on the operational semantics of the programming language).

Recall that for the polymorphic lambda calculus, without general references, we can model types as predicates (subsets) on some fixed set of values. But since our language of interest now includes dynamic allocation, it is natural to follow earlier work on models of languages with dynamic allocation of simple integer cells (e.g., [11, 25]), and use a Kripke-style possible-worlds model. Here, however, the set of worlds \mathcal{W} needs to be recursively defined since we consider general references: Semantically, a world maps locations (modeled as natural numbers) to semantic types in \mathcal{T} , and we thus arrive at the following recursive equations:

$$\begin{aligned} V &= \text{set of values, including locations} \\ \mathcal{W} &= \mathbb{N} \rightarrow_{\text{fin}} \mathcal{T} \\ \mathcal{T} &= \mathcal{W} \rightarrow \text{Pred}(V) \end{aligned}$$

With such a semantic model of types, one can give meaning to types in \mathcal{T} , in particular, the meaning of a reference type $\text{ref } \tau$ can be defined roughly as

$$(\text{ref } \tau)w = \{l \mid w(l) = \tau\},$$

i.e., for a world w , it is the set of locations l such that the semantic type recorded in the world at l is the same as τ .

Some readers might have expected that semantic types would be monotone wrt. an extension ordering of worlds. Indeed, it is often advantageous to build monotonicity into the model of types, but since this issue is mostly orthogonal to the point we are trying to make now, we will omit discussion of monotonicity until Section 3.2.

Observe that the natural model of types here is a *Kripke model over a recursively-defined set of worlds*. It is a Kripke model because the semantic types are parameterized over worlds. The problem is, of course, that, for cardinality reasons, there is no solution to the above equations in the category of sets (unfolding the above equation we get $\mathcal{W} = \mathbb{N} \rightarrow_{\text{fin}} (\mathcal{W} \rightarrow \text{Pred}(V))$ with \mathcal{W} in a negative position, see also [3]).

This observation leads Hobor et. al. [24] to propose that we should give up solving the equation and instead use an approximate solution, where the equations are not solved up to isomorphism, but where one only finds a retraction between two sets, equipped with some additional approximation information akin to the indices used in step-indexed models. As Hobor et. al. show, this suffices for many step-indexed models.

Instead, we here propose to solve the equation in a certain simple category of metric spaces. Our approach applies not only to step-indexed models but also to domain-theoretic models; it can be specialized to Hobor et. al.'s

indirection theory (see Section 3.1), and has a number of other advantages that we shall explain in Section 3.2. In the following two subsections we exemplify the approach in a domain-theoretic setting and in step-indexed setting, but let us first call to mind some facts about the metric spaces we are going to use.

Recap of ultrametric spaces A 1-bounded ultrametric space (X, d) is a metric space where the distance function $d : X \times X \rightarrow \mathbb{R}$ takes values in the closed interval $[0, 1]$ and satisfies the strong triangle inequality $d(x, y) \leq \max\{d(x, z), d(z, y)\}$, for $x, y, z \in X$. An (ultra-)metric space is complete if every Cauchy sequence has a limit. A function $f : X_1 \rightarrow X_2$ between metric spaces $(X_1, d_1), (X_2, d_2)$ is *non-expansive* if for all $x, y \in X_1$, $d_2(f(x), f(y)) \leq d_1(x, y)$, i.e., if application does not increase the distance between points. It is *contractive* if for some $\delta < 1$, $d_2(f(x), f(y)) \leq \delta \cdot d_1(x, y)$ for all $x, y \in X_1$.

The complete, 1-bounded, non-empty, ultrametric spaces and non-expansive functions between them form a Cartesian closed category $\text{CBUltr}_{\text{ne}}$. Products in $\text{CBUltr}_{\text{ne}}$ are given by the set-theoretic product where the distance is the maximum of the componentwise distances, and exponentials are given by the non-expansive functions equipped with the sup-metric. (i.e., the exponential $(X_1, d_1) \rightarrow (X_2, d_2)$ has the set of non-expansive functions from (X_1, d_1) to (X_2, d_2) as underlying set, and distance function: $d_{X_1 \rightarrow X_2}(f, g) = \sup\{d_2(f(x), g(x)) \mid x \in X_1\}$). For any set S and space $(X, d) \in \text{CBUltr}_{\text{ne}}$, the set of finite partial functions $S \rightarrow_{\text{fin}} X$ from S to X is again a complete bounded ultrametric space with distance function given by $d(f, g) = 1$, if the domain of f and g are not equal, and $d(f, g) = \max\{d(f(s), g(s)) \mid s \in \text{dom}(f)\}$, if the domain of f and g are equal.

A functor $F : \text{CBUltr}_{\text{ne}}^{\text{op}} \times \text{CBUltr}_{\text{ne}} \rightarrow \text{CBUltr}_{\text{ne}}$ is *locally non-expansive* if $d(F(f, g), F(f', g')) \leq \max\{d(f, f'), d(g, g')\}$ for all non-expansive f, f', g, g' , and it is *locally contractive* if $d(F(f, g), F(f', g')) \leq \delta \cdot \max\{d(f, f'), d(g, g')\}$ for some $\delta < 1$. By multiplication of the distances of (X, d) with a shrinking factor $\delta < 1$ one obtains a new ultrametric space, $\delta \cdot (X, d) = (X, d')$ where $d'(x, y) = \delta \cdot d(x, y)$. By shrinking, a locally non-expansive functor F yields a locally contractive functor $(\delta \cdot F)(X_1, X_2) = \delta \cdot (F(X_1, X_2))$. For a less condensed introduction to ultrametric spaces we refer to [33].

It is well-known that one can solve recursive domain-equations in $\text{CBUltr}_{\text{ne}}$, by an adaptation of the inverse-limit method from classical domain theory:

Theorem 2.1 (America-Rutten [7]). Let $F : \text{CBUltr}_{\text{ne}}^{\text{op}} \times \text{CBUltr}_{\text{ne}} \rightarrow \text{CBUltr}_{\text{ne}}$ be a locally contractive functor. Then there exists a unique (up to isomorphism) $(X, d) \in \text{CBUltr}_{\text{ne}}$ such that $F((X, d), (X, d)) \cong (X, d)$.

2.1 Domain-Theoretic Model

In [18] we gave a relationally parametric domain-theoretic model of a call-by-value language with impredicative polymorphism, general references, and recursive types. We now explain how it fits the intuitive model from above.

The model is based on an adequate domain-theoretic “untyped” model of the programming language that is defined in a mostly standard way, using a recursively defined pre-domain (complete partial order) V of values. The pre-domain V comes equipped with a family of projections $\pi_n : V \rightarrow V_\perp$, satisfying the usual conditions for projections arising from solutions to recursive domain equations. In particular, the minimal invariance property: the least upper bound of the projections $\bigsqcup_n \pi_n$ is the identity on V .

For the modeling of types, we use not all predicates on V , but only those that are *complete* (admissible), i.e., closed under least upper bounds of chains, and *uniform*. A subset P of V is *uniform* if it is closed under all the projections, i.e., if $\forall v \in P. \forall n. \pi_n(v) \in P_\perp$. For subsets P and Q of V , we write $\pi_n : P \rightarrow Q$ to mean that $\forall v \in P. \pi_n(v) \in Q_\perp$. It is well-known from earlier work on interpreting recursive types and impredicative polymorphism [1, 5, 6, 20, 26] that the set $CUPred(V)$ of all complete uniform predicates on V form a complete 1-bounded ultrametric space. The distance function d is defined by

$$d(P, Q) = \begin{cases} 2^{-\max\{n \in \mathbb{N} \mid \pi_n \in P \rightarrow Q \wedge \pi_n \in Q \rightarrow P\}} & \text{if } P \neq Q \\ 0 & \text{if } P = Q. \end{cases}$$

The distance is well-defined by properties of the projection functions, in particular the minimal invariance property.

It is easy to see that the functor $X \mapsto \mathbb{N} \rightarrow_{fin} \frac{1}{2}(X \rightarrow CUPred(V))$ from CBUlt_{ne}^{op} to CBUlt_{ne} is locally contractive and thus, by Theorem 2.1, there exists a complete 1-bounded ultrametric space \mathcal{W} satisfying:

$$\mathcal{W} \cong \mathbb{N} \rightarrow_{fin} \frac{1}{2}(\mathcal{W} \rightarrow CUPred(V)) \quad \text{in } \text{CBUlt}_{ne}.$$

Note that the rightmost function-space arrow in the display above denotes the function space in CBUlt_{ne} , i.e., the set of non-expansive functions. The $\frac{1}{2}$ is an example of a shrinking factor and, technically, ensures that the functor is locally contractive; it is a standard technique [7]. The intuitive reason for why it is ok to use the $\frac{1}{2}$ shrinking factor is that “it takes a computation step to dereference a location” (in [18] this is modelled via so-called semantic locations).

Having now succeeded in establishing the existence of the recursively defined set of worlds \mathcal{W} , we can define semantic types \mathcal{T} to be the set of non-expansive functions from \mathcal{W} to $CUPred(V)$. With this semantic model of types, we can give an interpretation of all the types of the programming language. (For recursive types, we employ Banach’s fixed point theorem to find a solution as the fixed point of

a contractive operator on \mathcal{T} .) Finally, we can define the typed meaning of terms by proving the fundamental theorem of logical relations wrt. the untyped semantics of terms. See [18] for a detailed treatment.

2.2 Step-Indexed Model

Now suppose that we want to build a semantic model over the operational semantics directly, without passing through a domain-theoretic model of the programming language. Then we can let V be the set of closed syntactic values v used in the operational semantics (i.e., v can be a pair of syntactic values v_1 and v_2 , a syntactic lambda abstraction, etc.), and, in keeping with the ideas of step-indexed models [2, 3, 8, 9], we can model types as (world-indexed) subsets of $\mathbb{N} \times V$ that are downwards-closed in the step (\mathbb{N}) component. More precisely, we define $UPred(V)$ to be

$$\{P \subseteq \mathbb{N} \times V \mid \forall (k, v) \in P. \forall j \leq k. (j, v) \in P\}.$$

We can define a distance function on $UPred(V)$, which measures “up-to-what-level” two uniform predicates agree, as follows: First, for $P \in UPred(V)$, let \bar{P}^k denote $\{(m, v) \in P \mid m < k\}$, and then define distance function d by:

$$d(P, Q) = \begin{cases} 2^{-n} & \text{if } P \neq Q \text{ and } n = \max\{k \mid \bar{P}^k = \bar{Q}^k\} \\ 0 & \text{if } P = Q. \end{cases}$$

Lemma 2.2. ($UPred(V), d$) is a well-defined object in CBUlt_{ne} .

Thus, by an application of Theorem 2.1, there exists a complete 1-bounded ultrametric space \mathcal{W} satisfying

$$\mathcal{W} \cong \mathbb{N} \rightarrow_{fin} \frac{1}{2}(\mathcal{W} \rightarrow UPred(V)) \quad \text{in } \text{CBUlt}_{ne},$$

and we can then define semantic types \mathcal{T} to be the set of non-expansive functions from \mathcal{W} to $UPred(V)$.

Thus by working in CBUlt_{ne} we can indeed solve the wished-for equations, even in a setting based on operational semantics. With this semantic model of types, one can then define an interpretation of all the types of the programming language, with definitions similar to those used in existing step-indexed models [3], but knowing that one has a solution to the wished-for recursive equation of worlds. We show how this can be done in Appendix A. Again, the intuitive reason for why it is ok to use the $\frac{1}{2}$ shrinking factor is that “it takes a computation step to dereference a location”.

We remark that it is not surprising that there is a connection between metric spaces and step-indexed models; this was already pointed out in [8]. The point is that it is useful not to forget this connection because it, e.g., allows us to define solutions to recursive world equations such as the one above. (See also the discussion in Section 3.2.)

We do not present a formal relationship to existing models for this particular example, but rather show, in the following section, how all the step-indexed models described via the indirection theory of Hobor et. al. can be obtained by a specialization of our general approach. In Section 3.2 we explain why it is useful to solve the world equation using metric spaces, and in Section 4 we present a new application.

3 The Essence of Step-Indexed Models

3.1 Specialization to Indirection Theory

Faced with a higher-order store recursive equation, Hobor, Dockins and Appel [24] provide an approximative solution. This is a section-retraction pair characterized by the two axioms of indirection theory that elegantly capture the approximative nature of the solution. Our approach is different, we solve the recursion proper in a certain category of metric spaces. In both cases, however, the solution provides a notion of worlds¹ to be used in Kripke models as exemplified amply in *loc.cit.* and in the previous section. In this section we shall argue that our approach is the more general in the sense that, for the same recursive equation, one may build the approximative solution of Hobor et. al. from our solution – this is Theorem 3.6. A consequence is that, somewhat indirectly, we have shown our method applicable to all examples considered by Hobor et. al.

This specialization to indirection theory is not unconditional. The construction presented by Hobor et. al. is parameterized over a set-theoretic functor $F : \text{Set} \rightarrow \text{Set}$ but our approach deals in metric-space equations phrased in terms of a locally non-expansive functor on CBUlt_{ne} . So we must have one of the latter corresponding to the former or, more precisely, we must have a *plain lift* of $F : \text{Set} \rightarrow \text{Set}$, as defined below, to apply the specialization. Fortunately, in many cases such a lift exists; indeed it always holds for functors on Set built with standard constructors as is made precise in Proposition 3.7. In particular we have plain lifts of the functors of all the examples of Hobor et. al.²

Definition 3.1. A functor $\hat{F} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$ is called *non-shrinking* if for any object X and any morphism $\varphi : X \rightarrow X$ of CBUlt_{ne} and any $m > 0$ such that

$$\forall x, y \in X. x =_m y \Rightarrow \varphi(x) = \varphi(y)$$

we also have that

$$\forall x, y \in \hat{F}(X). x =_m y \Rightarrow \hat{F}(\varphi)(x) = \hat{F}(\varphi)(y).$$

¹There is a conflict of nomenclature, what we call *worlds* are known as *knots* to Hobor et. al. Their worlds are pairs of knots and values.

²With the possible exception of [24, Example 2.7.]. The functor in that example is complex and the presentation dense to it is a bit hard to tell.

Here $x =_m y$ is short for $d(x, y) \leq 2^{-m}$ where d is the distance on X . Intuitively, elements of $\hat{F}(X)$ contain components from X . If closeness of two elements of $\hat{F}(X)$ implies similar closeness between the components, then \hat{F} is non-shrinking because $\hat{F}(\varphi)$ applies φ to all components. Note that the condition is required only to hold for $m > 0$; the case $m = 0$ comes down to preserving constant functions and that would preclude, e.g., constant functors.

Definition 3.2. A metric space is *bisected* if any non-zero distance is of the form 2^{-m} for some $m \in \mathbb{N}$.

For bisected metric spaces we have the following proposition which is useful for showing maps non-expansive:

Proposition 3.3. A map $\varphi : X \rightarrow X$ on a bisected metric space X is non-expansive if and only if we have

$$\forall m \in \mathbb{N}. \forall x, y \in X. x =_m y \Rightarrow \varphi(x) =_m \varphi(y).$$

Definition 3.4. We say that a functor $\hat{F} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$ is the *lift* of a functor $F : \text{Set} \rightarrow \text{Set}$ if the following diagram commutes

$$\begin{array}{ccc} \text{CBUlt}_{\text{ne}} & \xrightarrow{\hat{F}} & \text{CBUlt}_{\text{ne}} \\ U \downarrow & & \downarrow U \\ \text{Set} & \xrightarrow{F} & \text{Set}, \end{array}$$

where $U : \text{CBUlt}_{\text{ne}} \rightarrow \text{Set}$ is the obvious forgetful functor. Furthermore, we say that a functor $\hat{F} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$ is *plain* if it is non-shrinking, locally non-expansive and, on objects, preserves the property of being bisected.

Theorem 3.5. Let $\hat{F} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$ be a locally non-expansive functor and O a non-empty set. Then there is a non-empty, complete, 1-bounded ultrametric space X and an isometry

$$\Phi : X \cong \hat{F} \left(\frac{1}{2} (X \rightarrow_{\text{ne}} \text{UPred}(O)) \right).$$

This is an easy consequence of Theorem 2.1: \hat{F} is assumed locally non-expansive and the functor $\frac{1}{2}((-) \rightarrow_{\text{ne}} \text{UPred}(O))$ is a locally contractive contravariant functor on CBUlt_{ne} and so is the composite of the two.

Envision now a functor $F : \text{Set} \rightarrow \text{Set}$, a non-empty set O of values and a request for a solution to the recursive equation $K \cong F(K \times O \rightarrow 2)$. Indirection theory provides an approximative such, the above theorem another and the next theorem builds the former from the latter, thus demonstrating the generality of our approach.

We deviate from indirection theory as introduced by Hobor et. al. on two counts: We do not parameterize over the set of truth values but stick to $2 = \{0, 1\}$; the generalization, while probably technically feasible, appears unmotivated. More importantly, we build a solution that features

only *hereditary* maps from $K \times O$ to 2 , see the definition below. This is a direct consequence of the uniformity required of members of $UPred(O)$, lifting the latter constraint would most likely remove the former too. But we regard it as a strength, not a shortcoming, as we really would like to stay hereditary all the way and now we know that ‘unsquashing a knot’ does not invalidate this wish – compare with the discussion in the last paragraph of [24, Section 10].

Theorem 3.6. Let $F : \text{Set} \rightarrow \text{Set}$ be a functor with a plain lift $\hat{F} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$. We can, from the isometry of Theorem 3.5, build a set K , a subset of *hereditary* maps $K \times O \rightarrow_{\text{her}} 2$ of the full function space $K \times O \rightarrow 2$ and two maps

$$K \begin{array}{c} \xrightarrow{\text{unsquash}} \\ \xleftarrow{\text{squash}} \end{array} \mathbb{N} \times F(K \times O \rightarrow_{\text{her}} 2)$$

with the following three properties:

1. $\text{squash} \circ \text{unsquash} = 1_K$.
2. $(\text{unsquash} \circ \text{squash})(m, \nu) = (m, F(\text{approx}_m)(\nu))$.
3. $\forall \psi \in K \times O \rightarrow 2. \psi \in K \times O \rightarrow_{\text{her}} 2 \Leftrightarrow \psi = \square \psi$.

Here the $\text{level} = \text{fst} \circ \text{unsquash} : K \rightarrow \mathbb{N}$ and the map $\text{approx}_m : (K \times O \rightarrow_{\text{her}} 2) \rightarrow (K \times O \rightarrow_{\text{her}} 2)$ is defined, for each $m \in \mathbb{N}$, by

$$\text{approx}_m(\psi)(k, o) = \psi(k, o) \wedge \text{level}(k) < m.$$

And for $\psi \in K \times O \rightarrow 2$ we define $\square \psi \in K \times O \rightarrow 2$ by

$$(\square \psi)(k, o) = \forall l \in K. k A^* l \Rightarrow \psi(l, o),$$

where A^* is the reflexive, transitive closure of the relation A on K defined, for any two $k, l \in K$, by

$$k A l \Leftrightarrow \text{unsquash}(k) = (m + 1, \nu) \wedge l = \text{squash}(m, \nu).$$

Proposition 3.7. There is a plain lift of any functor built from the identity, constant non-empty sets, products, sums and (possibly finite and partial) maps from a constant set.

3.2 Advantages of Metric Solution Approach

Having proved that our metric-space approach specializes to the indirection theory of Hobor et. al. we now proceed to argue some advantages of our approach in general.

A first remark is this: We do not think of the operational semantics based version of our metric-space approach as more expressive than standard step-indexed models. Rather we view it as a framework for doing step-indexing, a conceptual guideline of sorts. This goes even if we disregard

higher-order store circularities. Consider, e.g., the interpretation of recursive types in Section 2.2 above and in Appendix A below. The idea of ‘stepping one down’ when interpreting $\mu\alpha.\tau$ seems natural to anyone familiar with step-indexed models. But coming up with the correct criteria on the interpretation function for this to work out properly, also with nested recursive types, is not, a priori, so easy. If, however, we employ the metric approach, including Banach’s fixed-point theorem, then writing down the requirements as done in the appendix is straightforward. Another example is the step-indexed model of Section 4, where we crucially rely on the metric on the worlds to define the \otimes operator by Banach’s fixed-point theorem. A similar construction could possibly be pushed through either with hand-built approximate worlds as employed by Ahmed et. al. [4] or with the indirection theory of Hobor et. al. [24]. But the precise course of action is less immediate and we expect that one could end up reinventing parts of metric theory on the way.

In direct comparison with the indirection theory in [24], we believe that our alternative approach of solving recursive metric equations has benefits. Both yield worlds to be used in Kripke models. There is, however, already a body of supporting theory for the metric-space approach that makes available a far greater range of worlds. To illustrate this point, let us focus on the step-indexed model of ML references discussed in Section 2.2 above and in Sections 2.1, 4.1 and 5 of [24]. In the model provided by indirection theory, types are arbitrary maps from worlds to values, modulo currying and nomenclature. But, as argued in [24, Section 5.1], we really want types that are both hereditary and monotone. In [24, Section 5.1] such types are elegantly identified using modal operators, but this does not change the problem that the types in a world may fail these criteria. This is recognized in the last paragraph of [24, Section 10] where an alternative, and less straightforward, model with only hereditary types in the worlds is sketched. But that means starting the model construction all over from scratch and does not buy us monotonicity. On the other hand, to obtain hereditary types with the metric approach we just use the uniformity condition on $UPred(V)$, verify Lemma 2.2 and apply Theorem 2.1. And to work with monotone types we can apply a slightly stronger existence result, cf. Appendix A and [17, Proposition 5.4] for *pre-ordered* metric spaces. A similar argument goes for the extension to mixed variance functors discussed [24, Section 10]: it is already supported by the metric-space approach. Indeed, in unpublished work we have used mixed-variance functors to verify that the metric-space approach scales to the elaborate worlds of [4].

Finally, we think that it is advantageous that the metric approach applies both to models based on domain theory and to models based on operational semantics.

4 Application: Step-Indexed Model of Separation Logic for Nested Hoare Triples

We next turn to a new application of recursively-defined sets of Kripke worlds: a step-indexed model of separation logic for nested Hoare triples.

In recent work, Schwinghammer et al. [32] present a domain-theoretic model of a variant of separation logic for a language that allows code to be stored in the heap (a form of “higher-order store”). The model is used to prove soundness of rules for “recursion through the heap” as well as soundness of higher-order frame rules that take stored code into account. (Both kinds of rules will be explained in more detail below.) The model is based on the solution of a recursive world equation using complete uniform subsets of a domain, akin to the situation in Section 2.1.

In this section we present a new, *operational* model of the same logic, following the approach outlined in Section 2.2. We do this for three reasons: first, to substantiate the claim that the metric-space technique works for both domain-theoretic and step-indexed models, and second, to illustrate the use of obtaining a *solution* (rather than an approximation of a solution) of a recursive equation for “worlds,” and three, to obtain a *simpler* model than the one in [32]. (We do not claim that the new model is sound for more inference rules than the one in [32].)

The development in the rest of this section mostly follows the one in [32]; we shall highlight some key differences. The reader is assumed to be familiar with basic properties of separation logic [31].

4.1 Programming language

Figure 1 presents the language we consider [32]. It deviates from the “standard” core programming language of separation logic [31] in two ways. First, stack variables are immutable: only heap cells can be updated. Second, commands are first-class values that can be stored in the heap: there is a new form of expression called a *quoted command*, written $\text{'}C\text{'}$, and a new command for evaluating stored commands, written $\text{eval } [e]$. Informally, if the value of e is an integer n , and if the current heap contains the quoted command $\text{'}C\text{'}$ at location n , then the command $\text{eval } [e]$ executes C as a subroutine.

We write $\text{fv}(C)$ for the set of free variables of the command C , and similarly for expressions. Let V be the set of closed values of the language, and let H be the set of *heaps*, i.e., finite maps from integers to closed values:

$$\begin{aligned} V &= \mathbb{Z} \cup \{\text{'}C\text{'} \mid \text{fv}(C) = \emptyset\}, \\ H &= \mathbb{Z} \rightarrow_{\text{fin}} V. \end{aligned}$$

For two heaps $h_1, h_2 \in H$ we write $h_1 \# h_2$ if they have disjoint domains and $h_1 \cdot h_2$ for their union if this is the

Expressions:

$$e ::= x \mid \text{'}C\text{'} \mid n \mid e_1 + e_2 \mid \dots \quad (n \in \mathbb{Z})$$

Commands:

$$\begin{aligned} C ::= & [e_1] := e_2 \mid \text{let } x = [e] \text{ in } C \mid \text{eval } [e] \\ & \mid \text{let } x = \text{new}(e) \text{ in } C \mid \text{free } e \\ & \mid \text{skip} \mid C_1; C_2 \mid \text{if } (e_1 = e_2) \text{ then } C_1 \text{ else } C_2 \end{aligned}$$

Figure 1. Programming language.

case. An *environment* is a finite map η from variables to closed values. When C is a command satisfying that $\text{fv}(C) \subseteq \eta$, we let $\eta(C)$ denote the result of applying η to C as a capture-avoiding substitution. Given an expression e and an environment η such that $\text{fv}(e) \subseteq \text{dom}(\eta)$, we define $\llbracket e \rrbracket_\eta \in V$ as follows. When e is a quoted command $\text{'}C\text{'}$ we let $\llbracket \text{'}C\text{'} \rrbracket_\eta = \llbracket \text{'}C\text{'} \rrbracket_\eta = \eta(C)$. When e is an arithmetic expression, $\llbracket e \rrbracket_\eta$ is defined in the expected way, *except* that arithmetic operations on quoted commands are, for definiteness, given the meaning 0. Thence we avoid the complications of introducing undefined expressions in a Hoare-style logic.³

The operational semantics of the language is defined by a small-step semantics, with configurations of the form (C, h) or abort . Configurations of the form (skip, h) or abort are terminal; An abort configuration indicates a memory fault or a runtime “type error” due to confusion between integers and quoted commands. The semantics is standard, all the reduction rules can be found in Appendix; here we just present the reduction rule for $\text{eval } [e]$:

$$(\text{eval } [e], h) \rightsquigarrow (C, h) \quad \text{if } \llbracket e \rrbracket = n \text{ and } h(n) = \text{'}C\text{'}$$

Example 4.1 (Iteration). The language does not include any high-level constructs for iteration. One can encode a “while” loop by means of “Landin’s knot” in the heap:

$$\begin{aligned} \text{while } [e] \neq 0 \text{ do } C \\ \stackrel{\text{def}}{=} \text{let } x = \text{new}(\text{'skip'}) \text{ in} \\ ([x] := \text{'let } y = [e] \text{ in} \\ \quad \text{if } (y = 0) \text{ then free } x \\ \quad \text{else } (C; \text{eval } [x])\text{'}; \\ \text{eval } [x]) \end{aligned}$$

(Here $x, y \notin \text{fv}(e, C)$.) With that abbreviation, the following rule is derivable in the logic we present below:

$$\frac{\Gamma \vdash \{\exists y. e \mapsto y * I(y) \wedge y \neq 0\} C \{\exists y. e \mapsto y * I(y)\}}{\Gamma \vdash \{\exists y. e \mapsto y * I(y)\} \text{while } [e] \neq 0 \text{ do } C \{e \mapsto 0 * I(0)\}}$$

³A more robust approach would be to introduce a simple type system that distinguishes integers from quoted commands; for simplicity we do not do so here.

4.2 Logic

The formulas of the logic [32] are called *assertions* and are generated by the grammar:

$$\begin{aligned}
P, Q ::= & \text{false} \mid \text{true} \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid \\
& \forall x. P \mid \exists x. P \mid \text{int}(e) \mid e_1 = e_2 \mid e_1 \leq e_2 \mid \\
& e_1 \mapsto e_2 \mid \text{emp} \mid P * Q \mid P \multimap Q \\
& \{P\}e\{Q\} \mid P \otimes Q \mid \dots
\end{aligned}$$

where the dots refer to atomic predicates and recursively defined predicates of the form $(\mu\alpha(x).P)(e)$ with α in P only occurring in “contractive” positions. (For space reasons, we do not formalize recursively defined assertions syntactically, but just treat them semantically, see below.) Unlike in standard separation logic, assertions are used both to describe predicates on heaps and to describe specifications of commands.

Indeed, the assertion $\{P\}e\{Q\}$ means, intuitively, that the value of e is a quoted command ‘ C ’ which satisfies the Hoare triple with precondition P and postcondition Q in the usual sense of separation logic. Since Hoare triples are assertions, they can appear in pre- and post-conditions of other triples. Such *nested* triples are useful for reasoning about stored code: the specification of a command C can depend on the specification of other code in the heap, e.g.,

$$\{P * \exists y. x \mapsto y \wedge \{P'\}y\{Q'\}\}'C'\{Q\}. \quad (1)$$

Here a part of the precondition of C is that x points to a command y satisfying $\{P'\}y\{Q'\}$. Presumably, the reason is that C contains one or more occurrences of `eval` $[x]$.

The assertion $P \otimes Q$ should be thought of as “the assertion P extended with the invariant Q ” and this assertion form is used to codify higher-order frame rules [15]. See [32] for detailed discussion of soundness and unsoundness of variations higher-order frame rules in the presence of higher-order store.

Proof Rules The proof rules include the standard rules for intuitionistic predicate logic and the logic of bunched implications [28]. Moreover, there are variations of standard separation logic proof rules (for dereferencing, sequencing, and so on). For brevity we only show the rule for dereferencing (the rest can be found in Appendix):

$$\frac{\Gamma, x \vdash \{P * e \mapsto x\}'C'\{Q\}}{\Gamma \vdash \{\exists x. P * e \mapsto x\}'\text{let } x = [e] \text{ in } C'\{Q\}} \quad (x \notin \text{fv}(e, Q))$$

Here Γ ranges over finite sets of variables.

In addition to these standard rules, there are two kinds of frame rules and a rule for executing stored code, see Figure 2. Rule (\otimes -FRAME) is a deep frame rule in which the invariant Q intuitively is added to all pre- and post-conditions inside P . The latter intuition is captured by the

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \otimes Q} \quad (\otimes\text{-FRAME})$$

$$\frac{}{\Gamma \vdash \{P\}e\{Q\} \Rightarrow \{P * R\}e\{Q * R\}} \quad (*\text{-FRAME})$$

$$\frac{\Gamma, k \vdash R[k] \Rightarrow \{P * e \mapsto R[_]\}k\{Q\}}{\Gamma \vdash \{P * e \mapsto R[_]\}'\text{eval } [e]\{Q\}} \quad (\text{EVAL})$$

Figure 2. Selected Proof Rules.

axioms in Figure 3. Rule ($*$ -FRAME) is a shallow (first-order) frame axiom. Finally, rule (EVAL) is the rule for executing stored code. Here, $e \mapsto R[_]$ is an abbreviation of $\exists x. e \mapsto x \wedge R[x]$ (for an x not free in R).

4.3 A step-indexed model

To model invariant extension $P \otimes Q$, Schwinghammer et. al. [32] models an assertion as a *function* that takes the meaning of a second, arbitrary assertion (to be thought of as the “invariant” that the first assertion is extended with) and gives a predicate on heaps.⁴ This approach introduces a circularity, however, since such a function will in particular be applicable to itself. In the next section we show how to formalize and solve the circularity using metric spaces.

4.3.1 Semantic predicates

Following Section 2.2, we let $UPred(H)$ be the set of subsets of $\mathbb{N} \times H$ that are downwards closed in the first component:

$$\{p \subseteq \mathbb{N} \times H \mid \forall (k, h) \in p. \forall j \leq k. (j, h) \in p\}.$$

We give $UPred(H)$ the same distance function as in Section 2.2; the set then becomes a complete, bounded ultrametric space. Using Theorem 2.1 we obtain a unique $\mathcal{W} \in \text{CBUlt}_{\text{ne}}$ satisfying

$$\mathcal{W} \cong \frac{1}{2}(\mathcal{W} \rightarrow UPred(H)). \quad (2)$$

Define $Pred = \frac{1}{2}(\mathcal{W} \rightarrow UPred(H))$ and let $i : Pred \rightarrow \mathcal{W}$ be the isomorphism. We will model assertions as elements of $Pred$.

Let the letters p and q range over elements of $Pred$. We order the elements of $Pred$ pointwise:

$$p \leq q \iff \forall w \in \mathcal{W}. p(w) \subseteq q(w)$$

⁴This idea follows earlier work on invariant extension [15, 16], which does not, however, deal with nested Hoare triples.

$$\begin{array}{ll}
P \circ R \stackrel{\text{def}}{=} (P \otimes R) * R & \{P\}e\{Q\} \otimes R \iff \{P \circ R\}e\{Q \circ R\} \\
(\kappa x.P) \otimes R \iff \kappa x.(P \otimes R) & (\kappa \in \{\forall, \exists\}, x \notin \text{fv}(R)) \\
(P \oplus Q) \otimes R \iff (P \otimes R) \oplus (Q \otimes R) & (\oplus \in \{\Rightarrow, \wedge, \vee, *, *\}) \\
P \otimes R \iff P & (P \text{ is true, false, emp, } e_1 = e_2, e_1 \mapsto e_2, \text{ or int}(e)) \\
(P \otimes R) \otimes R' \iff P \otimes (R \circ R') & P \otimes \text{emp} \iff P
\end{array}$$

Figure 3. Axioms for invariant extension.

Lemma 4.2. With the ordering above and the following operations, $Pred$ is a complete BI-algebra [14]:

$$\begin{aligned}
\text{emp}(w) &= \{(n, []) \mid n \in \mathbb{N}\} \\
(p * q)(w) &= \{(n, h) \mid \exists h_1, h_2. h = h_1 \cdot h_2 \\
&\quad \wedge (n, h_1) \in p(w) \wedge (n, h_2) \in q(w)\} \\
(p \multimap q)(w) &= \{(n, h) \mid \forall m \leq n. \\
&\quad ((m, h') \in p(w) \wedge h \# h') \Rightarrow (m, h \cdot h') \in q(w)\}
\end{aligned}$$

The fact that $Pred$ is a complete BI algebra immediately gives us a sound interpretation of most of the assertions in the logic [14], but to interpret recursive predicates we also need to know that the operations are non-expansive:

Lemma 4.3. The BI-algebra operations on $Pred$ given by the previous lemma are non-expansive:

$$\begin{aligned}
*, \multimap, \rightarrow, \wedge, \vee : Pred \times Pred &\rightarrow Pred \\
\bigvee_I, \bigwedge_I : (I \rightarrow Pred) &\rightarrow Pred.
\end{aligned}$$

(In the last two operations, the indexing set I is given the discrete metric.)

Proof. Easy verification. One first shows the analogous property for $UPred(H)$. To illustrate what follows, consider $*$: $UPred(H) \times UPred(H) \rightarrow UPred(H)$: It suffices to show that if $p \stackrel{n}{\approx} p'$ and $q \stackrel{n}{\approx} q'$, then also $(p * q) \stackrel{n}{\approx} (p' * q')$. The latter is equivalent to showing that $\forall m < n. (m, h) \in p * q \iff (m, h) \in p' * q'$, which follows easily by the assumption. \square

4.3.2 Interpretation of invariant extension

To interpret invariant-extension assertions $P \otimes Q$, we need an operator \otimes on the set of semantic predicates $Pred$. The most convenient way to specify \otimes is to give a certain recursive equation that it must satisfy. Using the metric-space setup we can then prove that there exists a unique operator satisfying this specification, by an easy application of Banach's fixed point theorem, as in [32].

Proposition 4.4. There exists a unique function $\otimes : Pred \times \mathcal{W} \rightarrow Pred$ in CBUlt_{ne} satisfying

$$p \otimes w = \lambda w'. p(w \circ w')$$

where $\circ : \mathcal{W} \times \mathcal{W} \rightarrow \mathcal{W}$ is given by

$$w_1 \circ w_2 = i((i^{-1}(w_1) \otimes w_2) * i^{-1}(w_2)).$$

Observe that it is here that we exploit that we have obtained a proper solution to the world equation (2) as a metric space such that we can now easily establish the existence of the recursively-defined \otimes -operation.

The basic properties of \otimes and \circ are conveniently summarized as follows:

Proposition 4.5. 1. $(\mathcal{W}, \circ, \text{emp})$ is a monoid.

2. The operator \otimes is a monoid action of \mathcal{W} on $Pred$: for all $P \in Pred$ and $w_1, w_2 \in \mathcal{W}$ we have $P \otimes \text{emp} = P$ and $(P \otimes w_1) \otimes w_2 = P \otimes (w_1 \circ w_2)$.

4.3.3 Interpretation of assertions

We next define a semantic interpretation of Hoare triples. To this end we let Safe_m be the set of configurations in the operational semantics that are safe for m reduction steps, that is, those configurations that do not reduce to **abort** in m (or fewer) steps. We write \rightsquigarrow_k for the k -step reduction relation of the operational semantics.

Now say that $w \models_n (p, C, q)$ holds iff: For all $r \in UPred$, all $m < n$ and all heaps h , if $(m, h) \in p(w) * i^{-1}(w)(\text{emp}) * r$, then:

1. $(C, h) \in \text{Safe}_m$.
2. For all $k \leq m$ and all $h' \in H$, if $(C, h) \rightsquigarrow_k (\text{skip}, h')$, then $(m - k, h') \in q(w) * i^{-1}(w)(\text{emp}) * r$.

This definition is similar to the one in [32] with its use of the invariant w and the baking-in of the first order frame rule, i.e., the quantification over r . The difference is that the meaning is now relative to the operational semantics (rather than denotational) and that we use step-indexing to measure to what extent pre- and post-conditions should hold.

The intention is, of course, that a Hoare-triple assertion is interpreted using the above semantic construct. However, to see that this interpretation gives a well-defined member of $Pred$, we need to know that a semantic Hoare triple is “non-expansive in w ”:

Proposition 4.6. If $w =_k w'$ and $w \models_n (p, C, q)$, then $w' \models_{n \wedge (k-1)} (p, C, q)$.

Proof. Easy verification, using the fact that the separating conjunction $*$ on $UPred(V)$ is non-expansive (Lemma 4.3). \square

The interpretation of an assertion $\Gamma \vdash P$ is now defined to be an element $\llbracket P \rrbracket_\eta$ in $Pred$, for η an environment mapping the variables in the domain of Γ to V . The definition uses the complete BI-algebra structure on $Pred$ given earlier to interpret the standard logical connectives, e.g.,

$$\llbracket P * Q \rrbracket_\eta w = \llbracket P \rrbracket_\eta w * \llbracket Q \rrbracket_\eta w.$$

Invariant extension is interpreted as follows:

$$\llbracket P \otimes Q \rrbracket_\eta w = \left(\llbracket P \rrbracket_\eta \otimes i(\llbracket Q \rrbracket_\eta) \right) w$$

and, finally, Hoare triples are interpreted like this:

$$\llbracket \{P\}e\{Q\} \rrbracket_\eta w = \begin{cases} \{(n, h) \mid w \models_n (\llbracket P \rrbracket_\eta, C, \llbracket Q \rrbracket_\eta)\} & \text{if } \llbracket e \rrbracket_\eta = 'C' \\ \emptyset & \text{otherwise.} \end{cases}$$

The concrete interpretation of all the logical connectives can be found in Appendix. As in [32], recursively defined predicates are interpreted via Banach's fixed point theorem:

Proposition 4.7. Let I be a set and suppose that, for each $i \in I$, $F_i : Pred^I \rightarrow Pred$ is a contractive function. Then there exists a unique $\vec{p} = (p_i)_{i \in I} \in Pred^I$ such that $F_i(\vec{p}) = p_i$, for all $i \in I$.

4.3.4 Soundness of proof rules

We define semantic validity of (open) assertions as follows: For an assertion P with free variables belonging to Γ , say that $\Gamma \models P$ iff: For all environments η with $\Gamma \subseteq \text{dom}(\eta)$ and all $w \in \mathcal{W}$ we have $\llbracket P \rrbracket_\eta w \in \mathbb{N} \times H$. This amounts to saying that $\llbracket P \rrbracket_\eta$ is the top element of the BI algebra $Pred$.

Theorem 4.8. If $\Gamma \vdash P$, then $\Gamma \models P$.

Proof. By showing the stronger property that each proof rule holds semantically, that is, with \vdash replaced by \models . We only include the proof case for $\text{eval } [e]$ (the other interesting cases are the ones for invariant extension; there one uses Proposition 4.5). We must show: if $\Gamma, z \models R[z] \Rightarrow \{P * e \mapsto R[_]\}z\{Q\}$, then $\Gamma \models \{P * e \mapsto R[_]\}'\text{eval } [e]\{Q\}$.

Let η be an environment with $\Gamma \subseteq \text{dom}(\eta)$, and let w and n be arbitrary. We must show that

$$w \models_n \{ \llbracket P * e \mapsto R[_] \rrbracket_\eta \}' \eta(\text{eval } [e]) \{ \llbracket Q \rrbracket_\eta \}. \quad (3)$$

So let $k < n$ and $r \in UPred$ and let $(k, h) \in \llbracket P * e \mapsto R[_] \rrbracket_\eta(w) * i^{-1}(w)(\text{emp}) * r$. Then $h = h_1 * [l \mapsto v] * h_2 * h_3$, where $(k, h_1) \in \llbracket P \rrbracket_\eta(w)$ and $\llbracket e \rrbracket_\eta = l$ and $(k, [l \mapsto v]) \in \llbracket R[z] \rrbracket_\eta[z \mapsto v](w)$ and $(k, h_2) \in i^{-1}(w)(\text{emp})$ and $(k, h_3) \in r$. Using validity of the premise, we get that $(k, [l \mapsto v]) \in \llbracket \{P * e \mapsto R[z]\}z\{Q\} \rrbracket_\eta[z \mapsto v](w)$, which means that $v = 'C'$ for some C , and that $w \models_k \{ \llbracket P * e \mapsto R[z] \rrbracket_\eta \}' C \{ \llbracket Q \rrbracket_\eta \}$.

Now, if $k = 0$, then conditions 1 and 2 in the definition of \models are clearly satisfied (item 2 because $(\eta(\text{eval } [e]), h)$ takes a reduction step), so (3) holds, as required. If $k > 0$ then, first observe that by downwards closure we have $(k-1, h) \in \llbracket P * e \mapsto R[_] \rrbracket_\eta(w) * i^{-1}(w)(\text{emp}) * r$. Therefore, $(C, h) \in \text{Safe}_{k-1}$, which implies that $(\eta(\text{eval } [e]), h) \in \text{Safe}_k$, so condition 1 in definition of \models is satisfied. For condition 2, we finally assume that $(\eta(\text{eval } [e]), h) \rightsquigarrow_m (\text{skip}, h')$ for some h' and $m \leq k$. Then $(C, h) \rightsquigarrow_{m-1} (\text{skip}, h')$. Since $m-1 \leq k-1$, we then get $((k-1)-(m-1), h') \in \llbracket Q \rrbracket_\eta(w) * i^{-1}(w)(\text{emp}) * r$, as required. \square

4.4 Discussion

In summa, we have developed a new step-indexed model of separation logic with nested Hoare triples for reasoning about higher-order store. The new model is arguably simpler than the one in [32], since it is phrased directly in terms of the operational semantics without passing through a domain-theoretic denotational semantics. A usual advantage of using domain-theory is a more abstract semantics, but in [32], it was necessary to employ certain ‘‘step-like,’’ rank functions, so in the end the model of *loc.cit.* was not more abstract than the new one presented here.

5 Related and Future Work

Relational Reasoning We have focused on unary reasoning in this paper, but as mentioned in the introduction the techniques developed here also apply to relational reasoning. Relational reasoning about higher-order store, e.g., logical relations for reasoning about contextual equivalence of programs, have been developed both based on domain theory, e.g., [12, 18], and on step-indexed models, e.g., [4]. For such relational reasoning, the worlds are typically more sophisticated than the worlds we have discussed so far, in order to describe situations in which programs are contextually equivalent even though they use local state in different ways. The third author has recently phrased the state-of-the-art world model from [4] as a recursive world equation over a domain-theoretic model. The reason for doing so is to obtain more abstract reasoning principles when using the resulting model for proving actual program equivalences, without having to reason about step-indices. As an alternative, Dreyer et. al. [22] have shown how to extend the relational step-indexed model [4] to a model of a modal logic for more abstract reasoning about program equivalences. The latter modal logic has been derived from the step-indexed model; we believe it is still a challenge to develop relational step-indexed models of some existing logics; e.g., for Hoare Type Theory [27]. Alternatively, one might try to develop a new formulation of (the ideas of) Hoare Type Theory based on a step-indexed model.

Formalization An often mentioned advantage of the traditional step-indexed approach is that it lends itself well to formalization in theorem provers and, indeed, impressive formalization work has been carried out in, e.g., Coq [10].

Thus, one may wonder, whether our proposed metric approach will hinder formalizations. We claim that it will not. Indeed, Varming has recently formalized the solution of recursive metric-space equations in Coq [34], following the treatment in [17]. This formalization can be used in concert with a formalization of operational semantics to yield, e.g., formalizations of the models in Sections 2.2–4, or with the formalizations of Benton et. al. of domain theory [13] to yield formalizations of models based on domain theory.

6 Conclusion and Acknowledgements

In conclusion, the key *conceptual contributions* of this paper are (1) the realization that models of higher-order store phenomena naturally can be given as Kripke models over worlds that are recursively defined in a category of metric spaces; and (2) a unification of methods based on classical domain theory and on step-indexed operational models. Our *technical contributions* include that (1) we have shown how to solve world equations for concrete step-indexed models; (2) we have shown that our metric approach can be specialized to Hobor et. al.’s recent proposal [24] (and argued that the metric approach has some advantages); and (3) we have developed a new model of separation logic with nested Hoare triples for reasoning about higher-order store, a model which shows the utility of the metric approach since it relies on a recursively defined operator on worlds.

We would like to thank Aquinas Hobor, Robert Dockins, Andrew W. Appel, Francois Pottier, and Carsten Varming for helpful discussions and insightful comments.

References

- [1] Martín Abadi and Gordon D. Plotkin. A per model of polymorphism and recursive types. In *Proceedings of LICS*, pages 355–365, 1990.
- [2] A. J. Ahmed, A. W. Appel, and R. Virga. A stratified semantics of general references. In *Proceedings of LICS 2002*, 2002.
- [3] Amal Ahmed. *Semantics of Types for Mutable State*. PhD thesis.
- [4] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. State-dependent representation independence. In *Proceedings of POPL*, 2009. To appear.
- [5] Roberto M. Amadio. Recursion over realizability structures. *Information and Computation*, 91(1):55–85, 1991.
- [6] Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*. Cambridge University Press, 1998.
- [7] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.*, 39(3):343–375, 1989.
- [8] A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5), 2001.
- [9] A. W. Appel, P. Melliès, C. D. Richards, and J. Vouillon. A very modal model of a modern, major, general type system. In *Proceedings of POPL 2007*, 2007.
- [10] A.W. Appel, R. Dockins, and A. Hobor. Mechanized semantic library. <http://msl.cs.princeton.edu/>, 2009.
- [11] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *Proceedings of TLCA*, 2005.
- [12] N. Benton, L. Beringer, M. Hofmann, and A. Kennedy. Relational semantics for effect-based program transformations: Higher-order store. In *Proceedings of PPDP’09*, 2009.
- [13] N. Benton, A. Kennedy, and C. Varming. Some domain theory and denotational semantics in coq. In *Proceedings of TPHOLS’09*, 2009.
- [14] B. Biering, L. Birkedal, and N. Torp-Smith. Bi-hyperdoctrines, higher-order separation logic, and abstraction. *ACM Trans. Program. Lang. Syst.*, 29(5), 2007.
- [15] L. Birkedal, N. Torp-Smith, and H. Yang. Semantics of separation-logic typing and higher-order frame rules for Algol-like languages. *LMCS*, 2(5:1), 2006. URL [http://dx.doi.org/10.2168/LMCS-2\(5:1\)2006](http://dx.doi.org/10.2168/LMCS-2(5:1)2006).
- [16] L. Birkedal, B. Reus, J. Schwinghammer, and H. Yang. A Simple Model of Separation Logic for Higher-order Store. In *Proceedings of ICALP 2008*, 2008.
- [17] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations. Technical Report ITU-2009-119, IT University of Copenhagen, 2009.
- [18] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In *Proceedings of FOSSACS*, number 5504 in LNCS, pages 456–470, 2009.
- [19] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Proceedings of APLAS*, number 4279 in LNCS, pages 79–96, 2006.
- [20] F. Cardone. Relational semantics for recursive types and bounded quantification. In *Proceedings of ICALP*, 1989.
- [21] Jaco de Bakker and Erik de Vink. *Control flow semantics*. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-04154-5.
- [22] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In *Proceedings of POPL’2010*, 2010.
- [23] A. Hobor, A.W. Appel, and F.Z. Nardelli. Oracle semantics for concurrent separation logic. In *Proceedings of ESOP’08*, 2008.
- [24] A. Hobor, R. Dockins, and A.W. Appel. A theory of indirection via approximation. In *Proceedings of POPL’2010*, 2010.
- [25] P. B. Levy. Possible world semantics for general storage in call-by-value. In *CSL: 16th Workshop on Computer Science Logic*, volume 2471 of LNCS.
- [26] David B. MacQueen, Gordon D. Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71(1/2):95–130, 1986.
- [27] A. Nanevski, G. Morrisett, and L. Birkedal. Polymorphism and separation in hoare type theory. In *Proceedings of ICFP’06*, 2006.
- [28] Peter W. O’Hearn and David J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.
- [29] Bernhard Reus and Jan Schwinghammer. Separation logic for higher-order store. In *Proc. CSL’06*, volume 4207 of LNCS, pages 575–590, 2006.
- [30] Bernhard Reus and Thomas Streicher. Semantics and logic of object calculi. In *Proceedings of 17th Annual IEEE Symposium Logic in Computer Science*, pages 113–124. IEEE Computer Society Press, 2002.
- [31] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of LICS*, pages 55–74, 2002.
- [32] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In *Proceedings of CSL*, number 5771 in LNCS, pages 440–454, 2009.
- [33] Michael B. Smyth. Topology. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.
- [34] C. Varming. Recursive metric-space equations in coq. Personal Communication, 2009.

A Step-Indexed Model of References

A.1 Language

The language is as in [22], except that we split the context for type variables and term variables in two so that term judgments take the form:

$$\Delta; \Gamma; \Sigma \vdash M : \tau$$

for Δ a context of type variables $\alpha_1, \dots, \alpha_n$, Γ a context of term variables $x_1 : \tau_1, \dots, x_m : \tau_m$, and Σ a context of locations $l_1 : \sigma_1, \dots, l_k : \sigma_k$.

Detailed typing judgments and operational semantics can be found in the online appendix to [22].

A.2 Model

Let V denote the set of closed syntactic values and let $UPred(V)$ be the set

$$\{P \subseteq \mathbb{N} \times V \mid \forall (k, v) \in P. \forall j \leq k. (j, v) \in P\}.$$

We can define a distance function on $UPred(V)$, which measures “up-to-what-level” two uniform predicates agree, as follows: First, for $P \in UPred(V)$, let \overline{P}^k denote $\{(m, v) \in P \mid m < k\}$, and then define distance function d by:

$$d(P, Q) = \begin{cases} 2^{-n} & \text{if } P \neq Q \text{ and } n = \max\{k \mid \overline{P}^k = \overline{Q}^k\} \\ 0 & \text{if } P = Q. \end{cases}$$

Lemma A.1. ($UPred(V), d$) is a well-defined object in CBUlt_{ne} .

In the same manner, we let E denote the set of closed syntactic expressions and define $UPred(E)$ to the corresponding set of uniform predicates on E .

Let $\text{PreCBUlt}_{\text{ne}}$ denote the category with objects (A, \leq) where A an object of CBUlt_{ne} and \leq is a preorder on the underlying set of A such that the following condition holds: if $(a_n)_{n \in \mathbb{N}}$ and $(b_n)_{n \in \mathbb{N}}$ are converging sequences in A with $a_n \leq b_n$ for all n , then also $\lim_{n \rightarrow \infty} a_n \leq \lim_{n \rightarrow \infty} b_n$. Morphisms are functions that are both monotone and non-expansive. By Proposition 5.4 in [17] we then immediately get:

Theorem A.2. There exists a preordered non-empty complete bounded ultra-metric space W with an isomorphism

$$W \cong \mathbb{N} \xrightarrow{\text{fin}} \frac{1}{2} (W \xrightarrow{\text{mon}} UPred(V))$$

in $\text{PreCBUlt}_{\text{ne}}$. One member of the right hand side is less than another if the domain of the first is included in the domain of the second and they agree on the former.

Semantic value types will be modeled as elements of

$$T = W \xrightarrow{\text{mon}} UPred(V),$$

and the semantic computation types will be modeled as elements of

$$T_E = W \xrightarrow{\text{mon}} UPred(E).$$

For Δ a context of type variables, we use φ to range over the product space $T^{|\Delta|}$ in CBUlt_{ne} .

We now define the interpretation of types in context as a function

$$\mathcal{V}[\Delta \vdash \tau] : T^{|\Delta|} \rightarrow T$$

in CBUlt_{ne} (i.e., note that the function space consists of non-expansive functions). The function is defined like this:

$$\begin{aligned} \mathcal{V}[\Delta \vdash 1] \varphi &= \lambda w. \{(k, *) \mid k \in \mathbb{N}\} \\ \mathcal{V}[\Delta \vdash \alpha] \varphi &= \varphi(\alpha) \\ \mathcal{V}[\Delta \vdash \tau_1 \rightarrow \tau_2] \varphi &= \lambda w. \{(k, v) \mid \forall w' \geq w. \forall j \leq k. \\ &\quad \forall v_1. (j, v_1) \in \mathcal{V}[\Delta \vdash \tau_1] \varphi w' \\ &\quad \Rightarrow (j, v v_1) \in \mathcal{E}[\Delta \vdash \tau_2] \varphi w'\} \\ \mathcal{V}[\Delta \vdash \text{ref } \tau] \varphi &= \lambda w. \{(k, l) \mid \\ &\quad (l \in \text{dom}(w) \wedge w(l) \stackrel{k}{=} \mathcal{V}[\Delta \vdash \tau] \varphi)\} \\ \mathcal{V}[\Delta \vdash \mu \alpha. \tau] \varphi &= \text{fix}(\lambda r. \lambda w. \{(k, \text{fold } v) \mid k > 0 \Rightarrow \\ &\quad (k-1, v) \in \mathcal{V}[\Delta, \alpha \vdash \tau] \varphi[\alpha \mapsto r] w\}) \\ \mathcal{V}[\Delta \vdash \forall \alpha. \tau] \varphi &= \lambda w. \{(k, v) \mid \forall \tau_0 \in \text{Type}. \forall r \in T. \\ &\quad \forall w' \geq w. \forall j \leq k. \\ &\quad (j, v[\tau_0]) \in \mathcal{E}[\Delta, \alpha \vdash \tau] \varphi[\alpha \mapsto r] w'\} \\ \mathcal{E}[\Delta \vdash \tau] \varphi &= \lambda w. \{(k, e) \mid \forall j \leq k. \forall s, v, s'. \\ &\quad ((e, s) \Downarrow^j (v, s') \wedge s :_k w) \\ &\quad \Rightarrow (\exists w' \geq w. s' :_{k-j} w' \wedge (k-j, v) \in \mathcal{V}[\Delta \vdash \tau] \varphi w')\} \end{aligned}$$

$$s :_k w \iff \forall j < k. \text{dom}(s) = \text{dom}(w) \wedge \forall l \in \text{dom}(w). (j, s(l)) \in w(l)(w)$$

Remark A.3.

- Note that in the case for $\mathcal{V}[\Delta \vdash \text{ref } \tau]$, we use k -equality in the space T .

Lemma A.4. If $w \stackrel{n}{=} w'$ and $w_0 \geq w$ then there exists w'_0 with

$$w'_0(l) = \begin{cases} w'(l) & \text{if } l \in \text{dom}(w') \\ w_0(l) & \text{otherwise} \end{cases}$$

and $w'_0 \in W$ and $w'_0 \stackrel{n}{=} w_0$.

Lemma A.5. If $s :_k w$ and $w \stackrel{n}{=} w'$ and $k < n$, then also $s :_k w'$.

Proof. We are to show that $\forall j < k. \text{dom}(s) = \text{dom}(w') \wedge \forall l \in \text{dom}(w'). (j, s(l)) \in w'(l)(w')$. It holds vacuously if $k = 0$, so assume $k > 0$. Then also $n > 0$. Let $j < k$ be arbitrary. By the assumption that $w \stackrel{n}{=} w'$, we get that

$\text{dom}(w) = \text{dom}(w') \wedge \forall l \in \text{dom}(w). \forall w_0. w(l)(w_0) \stackrel{n-1}{=} w'(l)(w_0)$. Since $\text{dom}(s) = \text{dom}(w)$ by the assumption that $s :_k w$ (using $k > 0$), we get $\text{dom}(s) = \text{dom}(w')$, as required. Moreover, we find that

$$w(l)(w) \stackrel{n}{=} w(l)(w') \stackrel{n-1}{=} w'(l)(w')$$

with the first equality since $w(l)$ is non-expansive, and the second equality by the assumption that $w \stackrel{n}{=} w'$. Thus, as $(j, s(l)) \in w(l)(w)$ by assumption, and since $j < k \leq n-1$, we also get $(j, s(l)) \in w'(l)(w')$, as desired. \square

Lemma A.6. $\mathcal{V}[\Delta \vdash \tau]$ and $\mathcal{E}[\Delta \vdash \tau]$ are well-defined. In particular,

- $\mathcal{V}[\Delta \vdash \tau]\varphi \in T$ (so non-expansive and monotone),
- $\mathcal{E}[\Delta \vdash \tau]\varphi \in T_E$ (so non-expansive),
- $\mathcal{V}[\Delta \vdash \tau]$ is a non-expansive map,
- The function $r \mapsto \lambda w. \{(k, \text{fold } v) \mid k > 0 \Rightarrow (k-1, v) \in \mathcal{V}[\Delta, \alpha \vdash \tau]\varphi[\alpha \mapsto r]w\}$ is contractive so the fixed point taken in $\mathcal{V}'[\Delta \vdash \mu\alpha.\tau]$ exists (this boils down to the use of $k-1$).

We now define interpretations of contexts and the logical relation interpretation of well-typed expressions:

$$\begin{aligned} \mathcal{D}[\Delta] &: T^{|\Delta|} \\ \mathcal{D}[\emptyset] &= \emptyset \\ \mathcal{D}[\Delta, \alpha] &= \{\varphi[\alpha \mapsto r] \mid \\ &\varphi \in \mathcal{D}[\Delta] \wedge r \in T\} \end{aligned}$$

$$\begin{aligned} \mathcal{G}[\Delta \vdash \Gamma] &: T^{|\Delta|} \rightarrow W \rightarrow \text{UPred}(V^{|\Gamma|}) \\ \mathcal{G}[\Delta \vdash \emptyset]\varphi &= \emptyset \\ \mathcal{G}[\Delta \vdash \Gamma, x : \tau]\varphi &= \lambda w. \{(k, \gamma[x \mapsto v]) \mid \\ &(k, \gamma) \in \mathcal{G}[\Gamma]\varphi w \wedge (k, v) \in \mathcal{V}[\Delta \vdash \tau]\varphi w\} \end{aligned}$$

$$\begin{aligned} \mathcal{S}[\Sigma] &= \{(k, w) \mid \forall (l : \tau) \in \Sigma. \\ &(k, l) \in \mathcal{V}[\emptyset \vdash \text{ref } \tau]\emptyset w\} \end{aligned}$$

$$\begin{aligned} \Delta; \Gamma; \Sigma \vdash e :^{\log} \tau &\iff \\ \exists \alpha_1, \dots, \alpha_n. \Delta &= \alpha_1, \dots, \alpha_n \wedge \forall \tau_1, \dots, \tau_n. \\ \forall k \geq 0. \forall \varphi. \forall \gamma. \forall w. \\ (\varphi \in \mathcal{D}[\Delta] \wedge (k, \gamma) &\in \mathcal{G}[\Delta \vdash \Gamma]\varphi w \wedge (k, w) \in \mathcal{S}[\Sigma]) \\ \Rightarrow ((k, [\alpha_1 \mapsto \tau_1, \dots, \alpha_n &\mapsto \tau_n](\gamma(e))) \in \mathcal{E}[\Delta \vdash \tau]\varphi w) \end{aligned}$$

Theorem A.7 (Fundamental Theorem of Logical Relations). If $\Delta; \Gamma; \Sigma \vdash e : \tau$, then $\Delta; \Gamma; \Sigma \vdash e :^{\log} \tau$.

B Specialization to Indirection Theory, Three Proofs

Proof of Proposition 3.3. It is immediate that any non-expansive φ has the stated property. Assume, on the other

hand, that we need to show φ non-expansive. Let $x, y \in X$, we must show that $d(\varphi(x), \varphi(y)) \leq d(x, y)$, where d is the metric on X . We may without loss of generality assume that $d(x, y) \neq 0$. But then there is $m \in \mathbb{N}$ with $d(x, y) = 2^{-m}$, in particular we have $d(x, y) \leq 2^{-m}$ which we usually write $x =_m y$. From the assumption we get that $\varphi(x) =_m \varphi(y)$, i.e., that $d(\varphi(x), \varphi(y)) \leq 2^{-m}$ and we are done. \square

Proof of Theorem 3.6. Let X and Φ be the result of applying Theorem 3.5 to \hat{F} . Note initially that X must be bisected. This is by definition the case for $\text{UPred}(O)$ and hence any two elements of $X \rightarrow_{ne} \text{UPred}(O)$ have a distance that is the supremum of a nonempty subset of $\{0\} \cup \{2^{-m} \mid m \in \mathbb{N}\}$. But this set is closed under nonempty suprema and so $X \rightarrow_{ne} \text{UPred}(O)$ is bisected too. Both of the functors $\frac{1}{2}(-)$ and \hat{F} preserve the property of being bisected, the former by construction and the latter by assumption. And so X , which is isometric to $\hat{F}(\frac{1}{2}(X \rightarrow_{ne} \text{UPred}(O)))$, must be bisected.

Without further ado, let us plunge into the construction. For every $m \in \mathbb{N}$ we know that $=_m$ is an equivalence relation on X , for $x \in X$ we denote by $[x]_m$ the equivalence class containing x . We let K be the sum of all but the first of the sets of equivalence classes:

$$K = \sum_{m \geq 1} X / =_m$$

Furthermore we let $K \times O \rightarrow_{her} 2$ consist of the set-theoretic maps $\psi : K \times O \rightarrow 2$ such that for any $(m, [x]_m) \in K$, any $o \in O$ and any $0 < n < m$ we have

$$\psi((m, [x]_m), o) \Rightarrow \psi((n, [x]_n), o).$$

To build squash and unsquash we need auxiliary maps:

$$\frac{1}{2}(X \rightarrow_{ne} \text{UPred}(O)) \xrightleftharpoons[B]{H} K \times O \rightarrow_{her} 2$$

defined by

$$H(\varphi) = \lambda((m, [x]_m), o) \in K \times O. \varphi(x) \ni (m-1, o)$$

respectively by

$$B(\psi) = \lambda x \in X. \{(m, o) \mid \psi((m+1, [x]_{m+1}), o)\}.$$

These are well-defined. To verify this for H take $\varphi \in \frac{1}{2}(X \rightarrow_{ne} \text{UPred}(O))$, $(m, [x]_m) \in K$ and $o \in O$. Notice initially that the choice of the representative x does not matter for if $x =_m y$ holds for two $x, y \in X$ we have $\varphi(x) =_m \varphi(y)$ too, in particular $(m-1, o) \in \varphi(x)$ if and only if $(m-1, o) \in \varphi(y)$. To prove $H(\varphi) \in K \times O \rightarrow_{her} 2$ we furthermore take $0 < n < m$ and assume that $H(\varphi)((m, [x]_m), o)$ holds, i.e., that $(m-1, o) \in$

$\varphi(x)$. Proving $H(\varphi)((n, [x]_n), o)$ comes down to showing $(n-1, o) \in \varphi(x)$ which is true by uniformity of $\varphi(x)$.

To verify that B is well-defined we take $\psi \in K \times O \rightarrow_{her} 2$. First we take $x \in X$ and must prove $\{(m, o) \mid \psi((m+1, [x]_{m+1}), o)\}$ uniform. So assume that we have $n < m \in \mathbb{N}$ and $o \in O$ with $\psi((m+1, [x]_{m+1}), o)$, we immediately get $\psi((n+1, [x]_{n+1}), o)$. Second we take $x, y \in X$ with $x =_m y$ for some $m \in \mathbb{N}$, we must show that $B(\psi)(x) =_m B(\psi)(y)$, i.e., that for all $n < m \in \mathbb{N}$ and all $o \in O$ we have $\psi((n+1, [x]_{n+1}), o)$ iff $\psi((n+1, [y]_{n+1}), o)$ but this is immediate since $[x]_{n+1} = [y]_{n+1}$. Here we used Proposition 3.3 to prove non-expansiveness of $B(\psi)$.

Going back and forth with H and B gets you nowhere. For $\psi \in K \times O \rightarrow_{her} 2$ we get that

$$\begin{aligned} H(B(\psi)) &= H(\lambda x. \{(m, o) \mid \psi((m+1, [x]_{m+1}), o)\}) \\ &= \lambda((m, [x]_m), o). \psi((m, [x]_m), o) \\ &= \psi \end{aligned}$$

and for $\varphi \in \frac{1}{2}(X \rightarrow_{ne} UPred(O))$ we get

$$\begin{aligned} B(H(\varphi)) &= B(\lambda((m, [x]_m), o). \varphi(x) \ni (m-1, o)) \\ &= \lambda x. \{(m, o) \mid \varphi(x) \ni (m, o)\} \\ &= \varphi. \end{aligned}$$

Up until this point, the maps H and B have been merely set-theoretic and not morphisms in CBUlt_{ne} , indeed, $K \times O \rightarrow_{her} 2$ is itself just a set. But now we equip it with the metric induced by the bijection with $\frac{1}{2}(X \rightarrow_{ne} UPred(O))$, i.e., the distance between to elements is the distance between the images of these elements under application of B . With this metric we obviously get an object of CBUlt_{ne} and the maps H and B are morphisms of CBUlt_{ne} , indeed, they are isometries. We need this to be able to apply \hat{F} to them.

Also we need, for each $m \in \mathbb{N}$, to define π_m on $\frac{1}{2}(X \rightarrow_{ne} UPred(O))$ by pointwise application of the restriction map, i.e., for $\varphi \in \frac{1}{2}(X \rightarrow_{ne} UPred(O))$ we define

$$\pi_m(\varphi)(x) = \varphi(x)|_m.$$

We should verify that this is a non-expansive map. It has been argued above that $\frac{1}{2}(X \rightarrow_{ne} UPred(O))$ is bisected so by Proposition 3.3 we take $\varphi_0, \varphi_1 \in \frac{1}{2}(X \rightarrow_{ne} UPred(O))$, $n \in \mathbb{N}$, assume $\varphi_0 =_n \varphi_1$ and aim to prove $\pi_m(\varphi_0) =_n \pi_m(\varphi_1)$. We may without loss of generality assume $n > 0$. For $x \in X$ we get by assumption that $\varphi_0(x) =_{n-1} \varphi_1(x)$ which implies that $\varphi_0(x)|_m =_{n-1} \varphi_1(x)|_m$ too and we are done. Really we would like to talk about the maps $(\text{approx}_m)_{m \in \mathbb{N}}$ on $K \times O \rightarrow_{her} 2$ but we cannot since squash and unsquash have not been defined yet; instead we deal in $(\pi_m)_{m \in \mathbb{N}}$ on $\frac{1}{2}(X \rightarrow_{ne} UPred(O))$. We shall need and prove a close correspondence between the two below.

We are now ready to construct the promised set-theoretic maps squash and unsquash. For $(m, [x]_m) \in K$ we define

$$\text{unsquash}(m, [x]_m) = \left(m-1, (\hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x) \right)$$

and for $(m, \nu) \in \mathbb{N} \times \hat{F}(K \times O \rightarrow_{her} 2)$ we set

$$\text{squash}(m, \nu) = \left(m+1, [(\Phi^{-1} \circ \hat{F}(B))(\nu)]_{m+1} \right).$$

Our first aim is to verify that unsquash is indeed well-defined, i.e., that the choice of the representative x does not matter. For $x, y \in X$ with $x =_m y$ for some $m > 0$ we get $\Phi(x) =_m \Phi(y)$ too. For any two $\varphi_0, \varphi_1 \in \frac{1}{2}(X \rightarrow_{ne} UPred(O))$ we get that if $\varphi_0 =_m \varphi_1$ then for any $z \in X$ we have $\varphi_0(z) =_{m-1} \varphi_1(z)$. But then

$$\begin{aligned} \pi_{m-1}(\varphi_0)(z) &= \varphi_0(z)|_{m-1} \\ &= \varphi_1(z)|_{m-1} \\ &= \pi_{m-1}(\varphi_1)(z) \end{aligned}$$

so we have $\pi_{m-1}(\varphi_0) = \pi_{m-1}(\varphi_1)$. As \hat{F} was assumed non-shrinking, we can now conclude that $\hat{F}(\pi_{m-1})(x) = \hat{F}(\pi_{m-1})(y)$ and we know that unsquash is well defined.

Before we go on, we need a quick comment on an easily overlooked issue. The maps squash and unsquash are both set-theoretic as desired but really they go between K and $\mathbb{N} \times U(\hat{F}(K \times O \rightarrow_{her} 2))$ where $U : \text{CBUlt}_{ne} \rightarrow \text{Set}$ is the forgetful functor. But we assumed \hat{F} a lift of F so

$$U(\hat{F}(K \times O \rightarrow_{her} 2)) = F(U(K \times O \rightarrow_{her} 2))$$

and the latter is what we usually just write $F(K \times O \rightarrow_{her} 2)$. So the domain respectively codomain of squash and unsquash really are what they are supposed to be.

With the issues of well-definedness taken care of, we now pursue the promised equalities. For $(m, [x]_m) \in K$ we calculate as follows:

$$\begin{aligned} (\text{squash} \circ \text{unsquash})(m, [x]_m) &= \text{squash} \left(m-1, (\hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x) \right) \\ &= (m, [(\Phi^{-1} \circ \hat{F}(B) \circ \hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)]_m) \\ &= (m, [(\Phi^{-1} \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)]_m). \end{aligned}$$

A bit of reasoning remains to show that this is indeed $(m, [x]_m)$. Notice first that we may rewrite $x = (\Phi^{-1} \circ \hat{F}(1_{\frac{1}{2}(X \rightarrow_{ne} UPred(O))}) \circ \Phi)(x)$. This means that if we can prove

$$\pi_{m-1} =_m 1_{\frac{1}{2}(X \rightarrow_{ne} UPred(O))}$$

then we are done as \hat{F} was assumed locally non-expansive. So take $\varphi \in \frac{1}{2}(X \rightarrow_{ne} UPred(O))$. For any $y \in X$ we get that

$$\pi_{m-1}(\varphi)(y) = \varphi(y)|_{m-1} =_{m-1} \varphi(y)$$

so in $X \rightarrow_{ne} UPred(O)$ we have $\pi_{n-1}(\varphi) =_{m-1} \varphi$ and we are done because of the shrinking factor.

For $(m, \nu) \in \mathbb{N} \times \hat{F}(K \times O \rightarrow_{her} 2)$ we get

$$\begin{aligned} & (\text{unsquash} \circ \text{squash})(m, \nu) \\ &= \text{unsquash} \left(m + 1, [(\Phi^{-1} \circ \hat{F}(B))(\nu)]_{m+1} \right) \\ &= (m, (\hat{F}(H) \circ \hat{F}(\pi_m) \circ \Phi \circ \Phi^{-1} \circ \hat{F}(B))(\nu)) \\ &= (m, (\hat{F}(H) \circ \hat{F}(\pi_m) \circ \hat{F}(B))(\nu)). \end{aligned}$$

To finish this we need to look into the relationship between π_m and the map approx_m . Take $\psi \in K \times O \rightarrow_{her} 2$, we start from one end and get that

$$\begin{aligned} & (\pi_m \circ B)(\psi) \\ &= \pi_m(\lambda x. \{ (n, o) \mid \psi((n+1, [x]_{n+1}), o) \}) \\ &= \lambda x. \{ (n, o) \mid \psi((n+1, [x]_{n+1}), o) \wedge n < m \} \\ &= \lambda x. \{ (n, o) \mid \text{approx}_m(\psi)((n+1, [x]_{n+1}), o) \} \\ &= (B \circ \text{approx}_m)(\psi) \end{aligned}$$

where we remember that $\text{level}(n+1, [x]_{n+1}) = n$ since level is the composite of the first projection and unsquash. Summing up we have proved that

$$\begin{aligned} & (\text{unsquash} \circ \text{squash})(m, \nu) \\ &= (m, (F(H) \circ F(B) \circ F(\text{approx}_m))(\nu)) \\ &= (m, F(\text{approx}_m)(\nu)) \end{aligned}$$

as desired – again we applied that \hat{F} is a lift of F .

We now consider the third property; we need to prove that the subset $K \times O \rightarrow_{her} 2$ of the full function space $K \times O \rightarrow 2$ coincides with the functions that are hereditary in the sense that they are fixed under application of \square . Take initially $(m, [x]_m)$ and $(n, [y]_n)$ in K , we get that $(m, [x]_m) \mathbf{A} (n, [y]_n)$ holds iff we have

$$\begin{aligned} & \text{unsquash}(m, [x]_m) = (l+1, \nu) \wedge (n, [y]_n) = \text{squash}(l, \nu) \\ & \Leftrightarrow (m-1, (\hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)) = (l+1, \nu) \wedge \\ & \quad (n, [y]_n) = (l+1, [(\Phi^{-1} \circ \hat{F}(B))(\nu)]_{l+1}) \\ & \Leftrightarrow m = n + 1 \wedge \\ & \quad [y]_n = [(\Phi^{-1} \circ \hat{F}(B) \circ \hat{F}(H) \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)]_n \\ & \Leftrightarrow m = n + 1 \wedge [y]_n = [x]_n. \end{aligned}$$

Here the last biimplication is a consequence of the fact that $x =_m (\Phi^{-1} \circ \hat{F}(\pi_{m-1}) \circ \Phi)(x)$ by arguments used to prove $\text{squash} \circ \text{unsquash} = 1_K$ above. It is immediate from this that for the closure \mathbf{A}^* of \mathbf{A} we have

$$(m, [x]_m) \mathbf{A}^* (n, [y]_n) \iff m \geq n \wedge [y]_n = [x]_n.$$

We know by definition that for $\psi \in K \times O \rightarrow 2$ we have $\psi = \square \psi$ iff for all $(m, [x]_m) \in K$ and all $o \in O$ we have

$$\psi((m, [x]_m), o) = (\square \psi)((m, [x]_m), o).$$

But by our characterization of \mathbf{A}^* we have that the right hand side again equals

$$\begin{aligned} & \forall (n, [y]_n) \in K. (m, [x]_m) \mathbf{A}^* (n, [y]_n) \Rightarrow \psi((n, [y]_n), o) \\ &= \forall n \leq m. \psi((n, [x]_n), o). \end{aligned}$$

From this it is immediate that $\psi = \square \psi$ holds iff we have that $\psi \in K \times O \rightarrow_{her} 2$ and we are done. \square

Proof of Proposition 3.7. We shall consider only three of the cases.

Constant Non-empty Sets Let X be some fixed non-empty set. Let $F : \text{Set} \rightarrow \text{Set}$ be the constant functor mapping any set to X and any function to the identity map 1_X . We need to come up with a plain lift $\hat{F} : \text{CBUlt}_{ne} \rightarrow \text{CBUlt}_{ne}$ of F . We naturally choose \hat{F} to be the constant functor mapping any object of CBUlt_{ne} to X equipped with the discrete metric d_1 and any morphism of CBUlt_{ne} to the identity map 1_X . This easily constitutes a locally non-expansive functor $\hat{F} : \text{CBUlt}_{ne} \rightarrow \text{CBUlt}_{ne}$ and obviously is a lift of F . For any $\varphi : (Y, d) \rightarrow (Z, e)$ whatsoever we get that for any $m > 0$ and any two $x, y \in \hat{F}(Y, d) = (X, d_1)$ with $x =_m y$ we have $x = y$, in particular we have that $\hat{F}(\varphi)(x) = 1_X(x) = 1_X(y) = \hat{F}(\varphi)(y)$. Hence \hat{F} is non-shrinking. Finally note that since (X, d_1) is bisected we have that \hat{F} maps all objects to bisected objects, in particular those that were bisected already.

Products Let us consider the case of products; we shall work with binary products only but the construction generalizes to any finite product. Take two functors $F, G : \text{Set} \rightarrow \text{Set}$ and define $H : \text{Set} \rightarrow \text{Set}$ by mapping a set X to the set $F(X) \times G(X)$ and a map $\varphi : X \rightarrow Y$ to $F(\varphi) \times G(\varphi) : F(X) \times G(X) \rightarrow F(Y) \times G(Y)$. Under the assumption that we have plain lifts $\hat{F}, \hat{G} : \text{CBUlt}_{ne} \rightarrow \text{CBUlt}_{ne}$ of F and G , we have to build a plain lift \hat{H} of H .

For an object $(X, d) \in \text{CBUlt}_{ne}$ we write $(Y, e) = \hat{F}(X, d)$ and $(Z, f) = \hat{G}(X, d)$ and assign

$$\hat{H}(X, d) = (Y \times Z, e \times f),$$

where the product metric $e \times f$ on $Y \times Z$ is defined by $(e \times f)((y_0, z_0), (y_1, z_1)) = \max(e(y_0, y_1), f(z_0, z_1))$ for any two $(y_0, z_0), (y_1, z_1) \in Y \times Z$. For a morphism $\varphi : (X_0, d_0) \rightarrow (X_1, d_1) \in \text{CBUlt}_{ne}$ we write $\hat{F}(\varphi) = (Y_0, e_0) \rightarrow (Y_1, e_1)$ and $\hat{G}(\varphi) = (Z_0, f_0) \rightarrow (Z_1, f_1)$ and assign

$$\hat{H}(\varphi) = \hat{F}(\varphi) \times \hat{G}(\varphi) : Y_0 \times Z_0 \rightarrow Y_1 \times Z_1.$$

It is well known that this yields a well-defined and locally non-expansive functor $\hat{H} : \text{CBUlt}_{ne} \rightarrow \text{CBUlt}_{ne}$. For the action on objects, this is spelled out in Lemmas 1.24 and 1.28 of [21].

We now proceed to prove that the functor \hat{H} is indeed a plain lift of H . First up is the property of being a lift, take an object (X, d) of CBUlt_{ne} . We write $(Y, e) = \hat{F}(X, d)$ and $(Z, f) = \hat{G}(X, d)$ and get that

$$\begin{aligned} U(\hat{H}(X, d)) &= U(Y \times Z, e \times f) \\ &= Y \times Z \\ &= U(\hat{F}(X, d)) \times U(\hat{G}(X, d)) \\ &= F(U(X, d)) \times G(U(X, d)) \\ &= H(U(X, d)). \end{aligned}$$

For a morphism $\varphi : (X_0, d_0) \rightarrow (X_1, d_1) \in \text{CBUlt}_{\text{ne}}$ we get the weirdly easy calculation $U(\hat{H}(\varphi)) = \hat{H}(\varphi) = \hat{F}(\varphi) \times \hat{G}(\varphi) = F(\varphi) \times G(\varphi) = H(\varphi)$ since the forgetful functor has no action on morphisms.

Next up is proof that \hat{H} is non-shrinking. Take a morphism $\varphi : (X_0, d_0) \rightarrow (X_1, d_1) \in \text{CBUlt}_{\text{ne}}$, we write $\hat{F}(\varphi) = (Y_0, e_0) \rightarrow (Y_1, e_1)$ and $\hat{G}(\varphi) = (Z_0, f_0) \rightarrow (Z_1, f_1)$. Assume that for some $m > 0$ we have that

$$\forall x, y \in X_0. x =_m y \Rightarrow \varphi(x) = \varphi(y).$$

Now take $(y_0, z_0), (y_1, z_1) \in Y_0 \times Z_0$ and assume that we have $(y_0, z_0) =_m (y_1, z_1)$. But then $y_0 =_m y_1$ and $z_0 =_m z_1$ by the definition of the metric $e_0 \times f_0$. And so we have

$$\begin{aligned} \hat{H}(\varphi)(y_0, z_0) &= \left(\hat{F}(\varphi) \times \hat{G}(\varphi) \right) (y_0, z_0) \\ &= \left(\hat{F}(\varphi)(y_0), \hat{G}(\varphi)(z_0) \right) \\ &= \left(\hat{F}(\varphi)(y_1), \hat{G}(\varphi)(z_1) \right) \\ &= \left(\hat{F}(\varphi) \times \hat{G}(\varphi) \right) (y_1, z_1) \\ &= \hat{H}(\varphi)(y_1, z_1) \end{aligned}$$

since both \hat{F} and \hat{G} were assumed non-shrinking. Finally we remark that \hat{H} preserves the property of being bisected since that holds by assumption for \hat{F} and \hat{G} and because the product metric introduces no new distances.

Finite, Partial Maps from a Constant Set Now on to finite, partial maps from a constant set. Take a set X and a functor $F : \text{Set} \rightarrow \text{Set}$, define $G : \text{Set} \rightarrow \text{Set}$ by mapping a set Y to the set $X \rightarrow_{\text{fin}} F(Y)$ of partial maps with a finite domain. A map $\varphi : Y \rightarrow Z$ is mapped to $\lambda\psi : X \rightarrow_{\text{fin}} F(Y)$. $F(\varphi) \circ \psi$. Under the assumption that we have a plain lift $\hat{F} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$ of F , we have to build a plain lift \hat{G} of G .

For an object $(Y, d) \in \text{CBUlt}_{\text{ne}}$ we write $(Z, e) = \hat{F}(Y, d)$ and assign

$$\hat{G}(Y, d) = (X \rightarrow_{\text{fin}} Z, e_{X \rightarrow_{\text{fin}}}),$$

where $e_{X \rightarrow_{\text{fin}}}(\psi_0, \psi_1)$ is $\max_{x \in \text{dom}(\varphi)} e(\psi_0(x), \psi_1(x))$ for any two $\psi_0, \psi_1 : X \rightarrow_{\text{fin}} Z$ with identical domain, otherwise the distance is 1. For a morphism $\varphi : (Y_0, d_0) \rightarrow (Y_1, d_1)$ in CBUlt_{ne} we write $\hat{F}(\varphi) : (Z_0, e_0) \rightarrow (Z_1, e_1)$ and employ that \hat{F} is a lift of F to simply assign

$$\hat{G}(\varphi) = G(\varphi) : (X \rightarrow_{\text{fin}} Z_0) \rightarrow (X \rightarrow_{\text{fin}} Z_1).$$

It is easily verifiable – if not exactly well known – that this yields a well-defined and locally non-expansive functor $\hat{G} : \text{CBUlt}_{\text{ne}} \rightarrow \text{CBUlt}_{\text{ne}}$, a high level argument is given in the proof of Proposition 22 of [18].

We now proceed to prove that the functor \hat{G} is a plain lift of G . First we verify that it is a lift, take an object (Y, d) of CBUlt_{ne} . We write $(Z, e) = \hat{F}(Y, d)$ and get that

$$\begin{aligned} U(\hat{G}(Y, d)) &= U(X \rightarrow_{\text{fin}} Z, e_{X \rightarrow_{\text{fin}}}) \\ &= X \rightarrow_{\text{fin}} Z \\ &= X \rightarrow_{\text{fin}} U(\hat{F}(Y, d)) \\ &= X \rightarrow_{\text{fin}} F(U(Y, d)) \\ &= G(U(Y, d)). \end{aligned}$$

The case of morphisms holds by definition.

Next up is proof that \hat{G} is non-shrinking. Take a morphism $\varphi : (Y_0, d_0) \rightarrow (Y_1, d_1) \in \text{CBUlt}_{\text{ne}}$, we write $\hat{F}(\varphi) = (Z_0, e_0) \rightarrow (Z_1, e_1)$. Assume that for some $m > 0$ we have that

$$\forall x, y \in Y_0. x =_m y \Rightarrow \varphi(x) = \varphi(y).$$

Now take $\psi_0, \psi_1 \in X \rightarrow_{\text{fin}} Z_0$ and assume that we have $\psi_0 =_m \psi_1$. We have $\text{dom}(\psi_0) = \text{dom}(\psi_1)$ and furthermore know that for all $x \in \text{dom}(\psi_0)$ we have $\psi_0(x) =_m \psi_1(x)$. We obviously have $\text{dom}(\hat{G}(\varphi)(\psi_0)) = \text{dom}(\psi_0) = \text{dom}(\psi_1) = \text{dom}(\hat{G}(\varphi)(\psi_1))$ and for any x in this domain we get

$$\begin{aligned} \hat{G}(\varphi)(\psi_0)(x) &= G(\varphi)(\psi_0)(x) \\ &= (F(\varphi) \circ \psi_0)(x) \\ &= F(\varphi)(\psi_0(x)) \\ &= F(\varphi)(\psi_1(x)) \\ &= (F(\varphi) \circ \psi_1)(x) \\ &= G(\varphi)(\psi_1)(x) \\ &= \hat{G}(\varphi)(\psi_1)(x) \end{aligned}$$

Finally we remark that \hat{G} preserves the property of being bisected since that holds by assumption for \hat{F} and because we introduce no new distances by taking a maximum of finitely many existing distances. \square

$$\begin{array}{c}
\frac{\Gamma, x \vdash \{P * e \mapsto x\}'C'\{Q\}}{\Gamma \vdash \{\exists x. P * e \mapsto x\}'\text{let } x = [e] \text{ in } C'\{Q\}} \quad (x \notin \text{fv}(e, Q)) \\
\text{(DEREF)} \\
\\
\frac{}{\Gamma \vdash \{e \mapsto _ * P\}'[e] := e_0'\{e \mapsto e_0 * P\}} \quad \text{(UPDATE)} \\
\\
\frac{\Gamma, x \vdash \{P * x \mapsto e\}'C'\{Q\}}{\Gamma \vdash \{P\}'\text{let } x = \text{new}(e) \text{ in } C'\{Q\}} \quad (x \notin \text{fv}(P, e, Q)) \quad \text{(NEW)} \\
\\
\frac{}{\Gamma \vdash \{e \mapsto _ * P\}'\text{free } e'\{P\}} \quad \text{(FREE)} \\
\\
\frac{}{\Gamma \vdash \{P\}'\text{skip}\{P\}} \quad \text{(SKIP)} \\
\\
\frac{\Gamma \vdash \{P\}'C_1'\{R\} \quad \Gamma \vdash \{R\}'C_2'\{Q\}}{\Gamma \vdash \{P\}'C_1; C_2'\{Q\}} \quad \text{(SEQ)} \\
\\
\frac{\Gamma \vdash \{P \wedge e_1 = e_2\}'C_1'\{Q\} \quad \Gamma \vdash \{P \wedge e_1 \neq e_2\}'C_2'\{Q\}}{\Gamma \vdash \{P\}'\text{if } (e_1 = e_2) \text{ then } C_1 \text{ else } C_2'\{Q\}} \quad \text{(IF)} \\
\\
\frac{\Gamma \vdash P' \Rightarrow P \quad \Gamma \vdash Q \Rightarrow Q'}{\Gamma \vdash \{P\}e\{Q\} \Rightarrow \{P'\}e\{Q'\}} \quad \text{(CONSEQ)} \\
\\
\frac{\Gamma \vdash P}{\Gamma \vdash P \otimes Q} \quad \text{(\otimes-FRAME)} \\
\\
\frac{}{\Gamma \vdash \{P\}e\{Q\} \Rightarrow \{P * R\}e\{Q * R\}} \quad \text{(*-FRAME)} \\
\\
\frac{\Gamma, k \vdash R[k] \Rightarrow \{P * e \mapsto R[_]\}k\{Q\}}{\Gamma \vdash \{P * e \mapsto R[_]\}'\text{eval } [e]'\{Q\}} \quad \text{(EVAL)}
\end{array}$$

Figure 5. Proof rules for Hoare triples.

C Step-Indexed Model of Separation Logic for Nested Hoare Triples

In this section we include additional definitions and proofs for the step-indexed model of separation logic for nested Hoare triples.

The operational semantics is specified in Figure 4.

See Figure 5 for additional separation logic proof rules.

The interpretation of assertions is written out in full in Figure 6.

| | |
|---|--|
| $([e_1] := e_2, h) \rightsquigarrow (\text{skip}, h[n \mapsto [e_2]])$ | $\text{if } [e_1] = n \text{ and } n \in \text{dom}(h)$ |
| $(\text{let } x = [e] \text{ in } C, h) \rightsquigarrow (C[v/x], h)$ | $\text{if } [e] = n \text{ and } h(n) = v$ |
| $(\text{eval } [e], h) \rightsquigarrow (C, h)$ | $\text{if } [e] = n \text{ and } h(n) = 'C'$ |
| $(\text{let } x = \text{new}(e) \text{ in } C, h) \rightsquigarrow (C[n/x], h * [n \mapsto [e]])$ | $\text{if } n \notin \text{dom}(h)$ |
| $(\text{free } e, h) \rightsquigarrow (\text{skip}, h')$ | $\text{if } [e] = n \text{ and } h = h' * [n \mapsto v]$ |
| $(C_1; C_2, h) \rightsquigarrow (C'_1; C_2, h')$ | $\text{if } (C_1, h) \rightsquigarrow (C'_1, h')$ |
| $(\text{skip}; C_2, h) \rightsquigarrow (C_2, h)$ | |
| $(\text{if } (e_1 = e_2) \text{ then } C_1 \text{ else } C_2, h) \rightsquigarrow (C_1, h)$ | $\text{if } [e_1] = [e_2]$ |
| $(\text{if } (e_1 = e_2) \text{ then } C_1 \text{ else } C_2, h) \rightsquigarrow (C_2, h)$ | $\text{if } [e_1] \neq [e_2]$ |
| $([e_1] := e_2, h) \rightsquigarrow \text{abort}$ | $\text{if } [e_1] = 'C' \text{ or } [e_1] = n \text{ where } n \notin \text{dom}(h)$ |
| $(\text{let } x = [e] \text{ in } C, h) \rightsquigarrow \text{abort}$ | $\text{if } [e] = 'C' \text{ or } [e] = n \text{ where } n \notin \text{dom}(h)$ |
| $(\text{eval } [e], h) \rightsquigarrow \text{abort}$ | $\text{if } [e] = 'C' \text{ or } [e] = n \text{ where } n \notin \text{dom}(h)$ |
| $(\text{eval } [e], h) \rightsquigarrow \text{abort}$ | $\text{if } [e] = n \text{ where } h(n) = m$ |
| $(\text{free } e, h) \rightsquigarrow \text{abort}$ | $\text{if } [e] = 'C' \text{ or } [e] = n \text{ where } n \notin \text{dom}(h)$ |
| $(C_1; C_2, h) \rightsquigarrow \text{abort}$ | $\text{if } (C_1, h) \rightsquigarrow \text{abort}$ |

Figure 4. Operational semantics.

| |
|---|
| $\llbracket \text{false} \rrbracket_\eta w = \emptyset$ |
| $\llbracket \text{true} \rrbracket_\eta w = \mathbb{N} \times H$ |
| $\llbracket P \wedge Q \rrbracket_\eta w = \llbracket P \rrbracket_\eta w \cap \llbracket Q \rrbracket_\eta w$ |
| $\llbracket P \vee Q \rrbracket_\eta w = \llbracket P \rrbracket_\eta w \cup \llbracket Q \rrbracket_\eta w$ |
| $\llbracket P \Rightarrow Q \rrbracket_\eta w = \{(n, h) \mid \forall m \leq n. (m, h) \in \llbracket P \rrbracket_\eta w \Rightarrow (m, h) \in \llbracket Q \rrbracket_\eta w\}$ |
| $\llbracket \forall x. P \rrbracket_\eta w = \bigcap_{v \in V} \llbracket P \rrbracket_{\eta[x \mapsto v]} w$ |
| $\llbracket \exists x. P \rrbracket_\eta w = \bigcup_{v \in V} \llbracket P \rrbracket_{\eta[x \mapsto v]} w$ |
| $\llbracket \text{int}(e) \rrbracket_\eta w = \begin{cases} \mathbb{N} \times H & \text{if } [e]_\eta = m \text{ for some } m \in \mathbb{Z} \\ \emptyset & \text{otherwise} \end{cases}$ |
| $\llbracket e_1 = e_2 \rrbracket_\eta w = \begin{cases} \mathbb{N} \times H & \text{if } [e_1]_\eta = [e_2]_\eta \\ \emptyset & \text{otherwise} \end{cases}$ |
| $\llbracket e_1 \leq e_2 \rrbracket_\eta w = \begin{cases} \mathbb{N} \times H & \text{if } [e_1]_\eta = m_1 \text{ and } [e_2]_\eta = m_2 \text{ where } m_1 \leq m_2 \\ \emptyset & \text{otherwise} \end{cases}$ |
| $\llbracket e_1 \mapsto e_2 \rrbracket_\eta w = \begin{cases} \{(n, [m \mapsto [e_2]_\eta]) \mid n \in \mathbb{N}\} & \text{if } [e_1]_\eta = m \text{ for some } m \in \mathbb{Z} \\ \emptyset & \text{otherwise} \end{cases}$ |
| $\llbracket \text{emp} \rrbracket_\eta w = \mathbb{N} \times \{\{\}\}$ |
| $\llbracket P * Q \rrbracket_\eta w = \llbracket P \rrbracket_\eta w * \llbracket Q \rrbracket_\eta w$ |
| $\llbracket P \multimap Q \rrbracket_\eta w = \llbracket P \rrbracket_\eta w \multimap \llbracket Q \rrbracket_\eta w$ |
| $\llbracket \{P\}e\{Q\} \rrbracket_\eta w = \begin{cases} \{(n, h) \mid w \models_n (\llbracket P \rrbracket_\eta, C, \llbracket Q \rrbracket_\eta)\} & \text{if } [e]_\eta = 'C' \\ \emptyset & \text{otherwise} \end{cases}$ |
| $\llbracket P \otimes Q \rrbracket_\eta w = (\llbracket P \rrbracket_\eta \otimes i(\llbracket Q \rrbracket_\eta)) w$ |
| $\llbracket (\mu\alpha(x).P)(e) \rrbracket_\eta w = \dots$ |
| $\llbracket \alpha(e) \rrbracket_\eta w = \dots$ |

Figure 6. Interpretation of assertions.