A Relational Realizability Model for Higher-Order Stateful ADTs

Lars Birkedal, Kristian Støvring, Jacob Thamsborg*

IT University of Copenhagen

Abstract

We present a realizability model for reasoning about contextual equivalence of higher-order programs with impredicative polymorphism, recursive types, and higher-order mutable state.

The model combines the virtues of two recent earlier models: (1) Ahmed, Dreyer, and Rossberg's step-indexed logical relations model, which was designed to facilitate proofs of representation independence for "state-dependent" ADTs and (2) Birkedal, Støvring, and Thamsborg's realizability logical relations model, which was designed to facilitate abstract proofs without tedious step-index arithmetic. The resulting model can be used to give abstract proofs of representation independence for "state-dependent" ADTs.

Keywords: Abstract Data Types, Logical Relations, Local State, Parametricity

1. Introduction

Reynolds [31] proposed to use *logical relations* for reasoning about polymorphic programs, in particular, to show equivalence of polymorphic programs and to show representation independence for abstract data types. Reynolds' work focused on System F, a core calculus for polymorphic functional programming. In recent years, there has been a lot of work on giving

*Corresponding Author

Email addresses: birkedal@itu.dk (Lars Birkedal), kss@itu.dk (Kristian Støvring), thamsborg@itu.dk (Jacob Thamsborg)

URL: www.itu.dk/~birkedal (Lars Birkedal), www.itu.dk/~kss (Kristian Støvring), www.itu.dk/~thamsborg (Jacob Thamsborg)

[1]Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark

logical relations models for reasoning about contextual equivalence and representation independence in increasingly realistic programming languages with effects [27, 29, 24, 10, 4, 17].

For programming languages involving recursive types and general references there are two main technical challenges:

Well-definedness Show that the logical relation is well-defined (that is exists); traditionally logical relations have been defined by induction on the structure of types but that is not possible in the presence of recursive types (and/or references).

Mutable Abstract Data Types Define the logical relation in such a way that one can use it to show equivalences of programs using local state for implementing mutable abstract data types in different ways

Recently (in 2009) two logical relation models, developed in parallel, were proposed for reasoning about a call-by-value language with impredicative polymorphism, recursive types, and general references: one was developed by Ahmed, Dreyer, and Rossberg (hereafter ADR) [3] and one was developed by the current authors (hereafter BST) [12]. Both models use *Kripke* logical relations to capture that the meaning of types depends on how many references have been allocated

We now highlight some features of the ADR and BST models to situate the present paper.

The ADR model is a step-indexed model over the operational semantics in which the logical relation is indexed by natural numbers, following ideas of Appel and McAllester [8]. Step-indexing is used to address the challenge of showing well-definedness of the logical relation. The main technical innovation in the ADR model is an advanced definition of worlds, which makes it possible to show contextual equivalences of many examples involving local state. In particular, it is possible to reason about programs using local state invariants that *evolve* over time.

The indexing over natural numbers makes reasoning directly in the model fairly low-level and cumbersome, however, since one has to keep explicit account of the indeces.

This led Dreyer et al. to develop logics for reasoning more abstractly about step-indexed logical relation models, first for a language without references [20] and then, most recently, for reasoning about the ADR model [22]. The latter logic, called LADR, is a modal relational logic in which one can

reason about ADR style contextual equivalences at a higher level of abstraction avoiding low-level details about steps and worlds.

The focus of the BST model was to obtain a relatively abstract logical relations model, without any step-indexing, by constructing the logical relations over a simple adequate domain-theoretic model of the programming language. Thence the well-definedness of the model was more complicated to establish and the main technical innovations in the BST model were (i) the observation that one can solve the naturally occurring recursive world equation in a category of ultrametric spaces and (ii) a novel modeling of locations with a domain-theoretic codification of approximation information, crucially used for establishing the well-definedness of the model. The model is indeed more abstract than the ADR model in the sense that, e.g., two functions f_1 and f_2 are related if they map related arguments to related results and there is no reasoning about steps. On the other hand, the BST model used a simple form of world, which only allowed to prove equivalences of programs that used local state in simple ways.

In this paper we extend the BST model with more refined worlds similar to those from the ADR model (specifically, we use the world description of LADR, which is a slight simplification of the one in ADR). Thus we show that the semantic techniques used in the BST model scale to state-of-the-art world descriptions and the resulting model can be used to show equivalences like those that can be shown using the ADR model, but with more abstract reasoning without any step-indexing. We compare reasoning in the resulting model to reasoning using the ADR model and the LADR logic.

2. Overview of the Technical Development

The present paper is a lengthy and somewhat technical one. To navigate safely the many details, we provide a quick, informal overview of the development and give extended textual explanations of some high points.

The language in question, including typing rules, is introduced in Section 3. It is a quite standard call-by-value language with universal, recursive and reference types.

An untyped denotational semantics is given in Section 4. The semantics is adequate and is given in monadic style by means of a universal predomain; this again, is obtained as the solution to a recursive domain equation. The semantics is quite standard with the exception of approximate locations, see Subsection 2.1 below. This section also defines the crucial domain-theoretic notions of uniform predomains and domains.

Some basic metric space theory is recalled in Section 5, in particular we discuss the notion of ultrametrics. Also we introduce a category of certain ordered metric spaces with an associated fixed-point theorem to be used in Section 7.

Bohr relations on uniform predomains and domains are defined and also equipped with a metric in Section 6. These are the kind of relations on states and values we will work with; the definition is motivated in Subsection 2.2 below.

The possible worlds of our Kripke logical relations are built in Section 7. These mimic the worlds of ADR. They are obtained as the fixed point of a functor on a certain category of ordered metric spaces; we laboriously build this functor and verify that it meets the requirements of the fixed point theorem. See also Subsection 2.3 below for a short, informal description of the worlds and some considerations on the choice of categories and fixed-point theorem.

The world-indexed logical relation is finally built in Section 8. The relation on states induced by a world corresponds to the approach taken in ADR, the interpretation of reference types does not, rather we take a more extensional approach. The remaining types are interpreted much as in BST, in particular we rely on our metric setup and Banach's fixed-point theorem in the case of recursive types. Also we rely on the approximate locations discussed in Subsection 2.1 below to ensure that the interpretation of reference types is well-defined.

The fundamental theorem of logical relations and proof resides in Section 9 after a definition of semantic relatedness; that the latter implies contextual approximation is an immediate corollary. The proof is lengthy, but it is a simple matter of verification in light of the definitions of the previous sections, and we only include some of the proof cases.

A worked-out example is the last Section 10 of the paper. We introduce some necessary syntactic sugar and prove the equivalence of Example 5.1 in ADR. This particular example is spelled out in ADR too, and so one can compare reasoning in the two models. Indeed, we conclude this section with some general considerations on this, also taking into account the recent LADR logic [22]. This serves as conclusion to the entire paper as well and has directions for future work.

2.1. Approximate Locations

As mentioned in the introduction, it is not, in general, trivial to prove the existence of logical relations in the presence of recursive types; a simple definition by induction on the types will not do. Minimal invariance as proposed by Pitts [28] and others is, arguably, the method of choice to tackle this issue, but it is not readily applicable because of the general reference types. In some sense, the standard, flat modelling of locations as integers does not provide enough foothold to get the iterative machinery of minimal invariance going.

Faced with this issue, the authors coined the idea of *approximate* locations in earlier work [13]. A location, say l, is modelled by an element λ_l that is the least upper bound of an ascending chain $\lambda_l^1 \sqsubseteq \lambda_l^2 \sqsubseteq \cdots \sqsubseteq \lambda_l$ of so-called approximate locations. The interpretation of references to a type ν then has 'proper' semantic locations such as λ_l as well as approximate semantic locations such as λ_k^{n+1}; the latter intuitively signifies that ν and the type of values stored at location k might agree only up to the nth approximation. These approximate locations were the crucial ingredient in a minimal invariance proof of existence of a logical relation indexed over syntactic worlds.

In BST, the minimal invariance was wrapped in metrics; this permitted the solution of a definitional circularity involving semantic worlds and semantic types. The approximate locations were still necessary, however; the interpretation of reference types simply would not be non-expansive without them. Here, we copy that usage, apart from a minor technical change to the

interpretation of lookup and assignment due to the more refined worlds. See Section 4 for details.

The approximate locations are required for technical reasons as sketched above. On the other hand, they do not mirror anything in the language and are, as such, junk. Some implications of their presence in the model is discussed at the end of Section 10.

2.2. Bohr Relations

One novelty of this paper is the particular choice of conditions we impose on our relations.

We carve our relations out of a universal predomain V that loosely corresponds to the set of closed, syntactic values. V is essentially obtained as the solution to a recursive domain equation as prescribed by Smyth and Plotkin [33]. But we must impose some restrictions – it will not do to allow all relations on V. The presence of recursive terms requires that relations respect the denotational construction of fixed points. And recursive types renders the existence of the logical relation non-trivial and the relations must accommodate that.

In BST we worked with *complete*, *uniform* relations. Complete means chain-complete, i.e., if we have an ascending chain of pairs in a relation, then the pair of the least upper bounds also must be in the relation. Uniform loosely means closed under the projections that come with solutions to recursive domain equations. For each $n \in \omega$ we have a projection $\pi_n : V \to V_\perp$; a relation $R \subseteq V \times V$ is uniform if for all $(v_1, v_2) \in R$ and all $n \in \omega$ we have that

$$(\pi_n(v_1), \pi_n(v_2)) \in \{\perp, \perp\} \cup \{\lfloor w_1 \rfloor, \lfloor w_2 \rfloor \mid (w_1, w_2) \in R\},$$

where $\lfloor - \rfloor : V \to V_\perp$ is the standard inclusion. Completeness and uniformity deal with the issues that arise from recursive terms and types respectively. Indeed, they are both well-known approaches, completeness is present, e.g., in work by Reynolds [31] and uniformity is found, e.g., in work by Abadi and Plotkin [2] and by Amadio [5].

Restricting to uniform and complete relations comes at a price, however: we are, e.g., unable to relate an integer to a pair of integers since the latter but not the former 'bottom out' under application of $\pi_1 : V \to V_\perp$. Similarly, one cannot relate, say, a list of integers, to it length; indeed, most non-trivial relations are not uniform. It is not just a question of taking the appropriate closure: if a relation has a pair (v_1, v_2) for which there is $n \in \omega$ such that, say,

$\pi_n(v_1) = \bot \neq \pi_n(v_2)$ then obviously the same is the case for any superset. This is a shortcoming because the conceptual relations that one 'plugs into' universal (and existential) types must be complete and uniform too, which limits the use of relational parametricity. None of the proofs of example equivalences of ADR appear to fail on these grounds, but it is easy to build equivalences that would: relating, say, a standard imperative counter to one that stores its count as the sum of a pair of integers cannot be done. Note that the restriction to uniform and complete relations does have some intuitive merit; we do, after all, approximate contextual equivalence with our relations and thus relating bottom to non-bottom seems inappropriate.

It is this shortcoming we address with *Bohr relations*, which we formally introduce in Section 6. Conceptually, we aim for relations that approximate contextual approximation rather than contextual equivalence. Technically, Bohr relations only restrict the left hand side: Bohr relations are chain-complete and downwards closed in the left coordinate. The former means, that if we have a sequence of pairs in the relation such that the left coordinates form an ascending chain and the right coordinates are identical, then the pair of the least upper bound of the left coordinates and the right coordinate must be in the relation too. The latter means, that if we have a pair in the relation, then any pair with a smaller left coordinate and identical right coordinate must be in the relation too. Being uniform instead of downwards closed in the left coordinate would work as well, but we stick to the latter for simplicity.

While not all relations on V are Bohr relations, they all have a least Bohr relation that contain them; this closure can be 'plugged into' universal types. Thus the overall idea is to remove the artificial 'synchronization' restriction imposed by (two-sided) uniformity and so be free to apply relational parametric reasoning at will.

Going for contextual approximation instead of contextual equivalence seems standard in recent step-indexed models of recursive types. There is an analogy to Bohr relations here: step-indexed models, e.g. [4], do not require expressions to terminate in the same number of steps in order for them to be related. Rather they allot a number of steps for the left hand side to terminate, and if this happens then the right hand side is required to terminate in any number of steps. Requiring the expressions to march in step would, most likely, not invalidate the soundness of the reasoning but rather prove fewer (albeit stronger) equivalences.

For expository reasons, we have focused on relations between values and reasoning by relational parametricity in the explanations above. It is worth-

while, however, to note, that the restrictions on relations apply to relations between states too. In particular, we would have been unable to relate, say, the empty state to any non-empty state containing a pair with the (two-sided) uniformity requirement of BST. This posed no problem in BST because of the simple notion of worlds, but it would have been a severe limitation here.

We finally remark that the idea of approximating contextual approximation rather than contextual equivalence is present in the 4-tuples of Bohr and Birkedal [17], hence the nomenclature. Their setup handled any kind of relation, whether complete or not, uniform or not. We think we have distilled this ability: 4-tuples are – roughly and in retrospect – just two Bohr relations grouped together to be able to argue both ways of contextual approximation in one go.

2.3. Solving Recursive World Equations

Definitional circularities arise when modelling higher-order store phenomena; the main accomplishment of BST is the use of metric space theory to solve one such circularity. That particular circularity involves both the space of types and the space of worlds and so one has the choice of solving for either. This is not, however, an immaterial choice. Types come with no particular order and we can make do with a classic fixed-point result for functors on the category of ultrametric spaces by America and Rutten [6]; this was the approach taken in BST. Worlds, on the other hand, come with an extension ordering that corresponds to further allocation. Hence we arrive at a functor on certain ordered metric spaces and the cited result no longer suffices.

In ADR, the notion of world is far more refined than in BST. A world is a series of islands, each managing separate parts of the store. Islands themselves are dynamic, they have a population that may grow according to a population law. Also each island has a heap law that regulates the part of the store managed by the island; the heap law is indexed by the population and hence may vary over time. We refer the reader to Section 7 and in particular to ADR for further motivation and explanation; here it shall suffice to state that the heap laws are indexed also over worlds themselves and so the definition of worlds is circular. But unlike the circularity solved in BST, there seems to be no way of 'cycling' this circularity to arrive at point where the fixed-point result of America and Rutten is applicable.

Faced with this challenge, the authors proved a generalized fixed-point theorem [15] that allows for additional structure on the metric spaces, in

particular certain orderings. And it is a special case of this theorem that we shall apply in Section 7 to build our space of worlds.

Recently, Dreyer et al. have developed the logic LADR [22] to facilitate reasoning in the ADR model. In the process, they simplified the ADR model somewhat and it is the notion of worlds from this, simpler model, that we have chosen to settle on in this paper. We believe that this simplification has removed the obstacles that prevented the use of the fixed-point result of America and Rutten in our adaptation of the original ADR model. In other words, we probably could do without the aforementioned generalized fixed-point result. It would, however, take some amount of 'hacking' to do so and the development would be more complicated.

A natural question to conclude this subsection is this: why do we use metric space theory instead of domain theory to solve the definitional circularity; after all, the latter is arguably the more standard tool for computer scientists. The answer to this is somewhat vague: we probably could have used the standard solution to recursive domain equations [33], but the orderings on the domains would be less natural than the metrics we use here. Indeed, we have results that translates back and forth between the two approaches, but just writing out the circularity and equipping the domains with the standard orderings does not work out well. You could hammer a nail into the wall with a screwdriver, but using a hammer is the natural choice.

3. Programming Language

We consider the same programming language as the one used in the BST model [12]. It is a standard call-by-value language with universal types, iso-recursive types, ML-style reference types, and a ground type of integers.

The language is sketched in Figures 1 and 2. The typing rules are standard [26]. In the figure, Ξ and Γ range over contexts of type variables and term variables, respectively. As we do not consider operational semantics in this article, there is no need for location constants, and hence no need for store typings.

4. Untyped semantics

The terms of the language above are not intrinsically typed. In other words, the language consists of an untyped term language and a set of rules

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

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

Typing rules:

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

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

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

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

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

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

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

(Continued in Figure 2.)

Figure 1: Programming language

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

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

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

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

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

Figure 2: Programming language (ctd.)

for assigning types to untyped terms. We now take advantage of this distinction and give a semantics of the untyped term language. This "untyped semantics" is almost identical to the one used in the BST model [12, 16] (we point out some minor differences below), but we include a description here in order to keep the article self-contained. See the cited papers and earlier work [13] by the authors for connections, i.e., adequacy, to a standard operational semantics.

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

The only non-standard aspect of the semantics is the treatment of store locations. As explained in Section 2.1, the model includes *approximate* locations. This means that locations are modeled as elements of the cpo $Loc = \mathbb{N} \times \overline{\omega}$ where $\overline{\omega}$ is the "vertical natural numbers" cpo: $1 \sqsubset 2 \sqsubset \cdots \sqsubset n \sqsubset \cdots \sqsubset \infty$. (For notational reasons it is convenient to call the least element 1 rather than 0.) The intuitive idea is that locations can be approximated: the element $(l, \infty) \in Loc$ is the "ideal" location numbered l, while the elements of the form (l, n) for $n < \infty$ are its approximations. As already mentioned, these approximate locations are included in order to ensure that the logical relation we construct is well-defined.

4.1. Domain-theoretic preliminaries

We assume that the reader is familiar with basic denotational semantics, as presented for example in Winskel [35], and with semantics in monadic style [25].

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

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

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

Having now specified the kinds of partial orders we use, we follow common practice and introduce some more abstract terminology: in this article, a *predomain* simply means a cpo, and a *domain* means a cppo.

The semantics below is presented in monadic style [25], i.e., structured using a monad that models the effects of the language. It is most convenient to define this monad by means of a Kleisli triple: for every predomain S and every domain Ans, the continuation-and-state monad $T_{S,\mathit{Ans}} : \mathsf{Cpo} \to \mathsf{Cpo}$ over S and Ans is given by

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

where $\eta_A : A \to T_{S,\mathit{Ans}} A$ and $\star_{A,B} : T_{S,\mathit{Ans}} A \to (A \to T_{S,\mathit{Ans}} B) \to T_{S,\mathit{Ans}} B$. In the following we omit the type subscripts on η and $\star$. (Continuations are included for a technical reason, namely to ensure chain-completeness of the relations that will be used to model computations.)

4.2. A universal uniform predomain

The standard methods for solving recursive domain equations give solutions that satisfy certain induction principles [33, 28]. One way of formulating this property is that one obtains as a solution not only a domain D, but also a sequence of "projection" functions ϖ_n on D such that each element d of D is the limit of its projections $\varpi_0(d)$, $\varpi_1(d)$, etc. These functions therefore provide a handle for proving properties about D by induction on n.

Definition 4.1.

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

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \cdots \sqsubseteq \varpi_n \sqsubseteq \ldots \tag{1}$$

$$\bigsqcup_{n\in\omega} \varpi_n = \lambda a.\lfloor a \rfloor \tag{2}$$

$$\varpi_m \mathbin{\overline{\circ}} \varpi_n = \varpi_n \mathbin{\overline{\circ}} \varpi_m = \varpi_{\min(m,n)} \tag{3}$$

$$\varpi_0 = \lambda e.\, \perp\,. \tag{4}$$

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

$$\varpi_0 \sqsubseteq \varpi_1 \sqsubseteq \cdots \sqsubseteq \varpi_n \sqsubseteq \ldots \tag{5}$$

$$\bigsqcup_{n\in\omega} \varpi_n = id_D \tag{6}$$

$$\varpi_m \circ \varpi_n = \varpi_n \circ \varpi_m = \varpi_{\min(m,n)} \tag{7}$$

$$\varpi_0 = \lambda e.\, \perp\,. \tag{8}$$

Uniform domains are called *rank-ordered cpos* in earlier work by Baier and Majster-Cederbaum [9].

Proposition 4.2. *There exists a uniform predomain* $(V, (\pi_n)_{n\in\omega})$ *satisfying the following two properties:*

1. *The following isomorphism holds in* Cpo*:*

$$V \;\cong\; \mathbb{Z} + \mathit{Loc} + 1 + (V \times V) + (V + V) + V + T_{S,Ans}V + (V \to T_{S,Ans}V) \tag{9}$$

where

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

and

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

2. *Abbreviate* $TV = T_{S,Ans}V$ *and* $K = V \to S \to Ans$. *Define the following injection functions corresponding to the summands on the right-hand side of the isomorphism* (9)*:*

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

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

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

Proof (sketch). Proposition 3.2 of Birkedal et al. [14] gives a uniform predomain $(V, (\varpi_n)_{n\in\omega})$ where V satisfies (9). The proposition furthermore gives a uniform predomain $(S, (\varpi^S_n)_{n\in\omega})$ as well as uniform domains $(K, (\varpi^K_n)_{n\in\omega})$ and $(TV, (\varpi^T_n)_{n\in\omega})$ where S, K, and TV are as above. Now define the functions π_n as shown in Figure 3, by induction on n. We must show that $(V, (\pi_n)_{n\in\omega})$ is a uniform predomain.

One can show the following inequalities by mutual induction:

$$\begin{aligned} \varpi_n &\sqsubseteq \pi_{n+1} \sqsubseteq \varpi_{n+1} \\ \varpi^S_n &\sqsubseteq \pi^S_n \sqsubseteq \varpi^S_{n+1} \\ \varpi^K_n &\sqsubseteq \pi^K_n \sqsubseteq \varpi^K_{n+1} \\ \varpi^T_n &\sqsubseteq \pi^T_n \sqsubseteq \varpi^T_{n+1}\,. \end{aligned}$$

It follows from the first inequality that $(\pi_n)_{n\in\omega}$ is increasing. Furthermore, the same inequality gives that $\bigsqcup_{n\in\omega} \pi_n = \lambda v.\lfloor v \rfloor$ since $\bigsqcup_{n\in\omega} \varpi_n = \bigsqcup_{n\in\omega} \varpi_{n+1} =$

$$\pi_0 = \lambda v.\bot \tag{10}$$

$$\pi_{n+1}(in_{\mathbb{Z}}(m)) = \lfloor in_{\mathbb{Z}}(m) \rfloor \tag{11}$$

$$\pi_{n+1}(in_1(*)) = \lfloor in_1(*) \rfloor \tag{12}$$

$$\pi_{n+1}(in_{Loc}(l,\infty)) = \lfloor in_{Loc}(l, n+1) \rfloor \tag{13}$$

$$\pi_{n+1}(in_{Loc}(l,m)) = \lfloor in_{Loc}(l, \min(n+1, m)) \rfloor \tag{14}$$

$$\pi_{n+1}(in_{\times}(v_1, v_2)) = \begin{cases} \lfloor in_{\times}(v_1', v_2') \rfloor & \text{if } \pi_n\, v_1 = \lfloor v_1' \rfloor \text{ and } \pi_n\, v_2 = \lfloor v_2' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{15}$$

$$\pi_{n+1}(in_{+}(\iota_i\, v)) = \begin{cases} \lfloor in_{+}(\iota_i\, v') \rfloor & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \qquad (i = 1, 2) \tag{16}$$

$$\pi_{n+1}(in_{\mu}\, v) = \begin{cases} \lfloor in_{\mu}\, v' \rfloor & \text{if } \pi_n v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{17}$$

$$\pi_{n+1}(in_{\forall}\, c) = \lfloor in_{\forall}(\pi_n^T\, c) \rfloor \tag{18}$$

$$\pi_{n+1}(in_{\to}\, f) = \left\lfloor in_{\to}\left(\lambda v.\, \begin{cases} \pi_n^T\,(f\, v') & \text{if } \pi_n\, v = \lfloor v' \rfloor \\ \bot & \text{otherwise} \end{cases}\right)\right\rfloor \tag{19}$$

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

$$\pi_0^S = \lambda s.\bot \qquad \pi_0^K = \lambda k.\bot \qquad \pi_0^T = \lambda c.\bot \tag{20}$$

$$\pi_{n+1}^S(s) = \begin{cases} \lfloor s' \rfloor & \text{if } \pi_{n+1} \circ s = \lambda l.\lfloor s'(l) \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{21}$$

$$\pi_{n+1}^K(k) = \lambda v.\lambda s.\begin{cases} k\, v'\, s' & \text{if } \pi_{n+1}\, v = \lfloor v' \rfloor \text{ and } \pi_{n+1}^S\, s = \lfloor s' \rfloor \\ \bot & \text{otherwise} \end{cases} \tag{22}$$

$$\pi_{n+1}^T(c) = \lambda k.\lambda s.\begin{cases} c\,(\pi_{n+1}^K\, k)\, s' & \text{if } \pi_{n+1}^S\, s = \lfloor s' \rfloor \\ \bot & \text{otherwise}\,. \end{cases} \tag{23}$$

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

$\lambda v.\lfloor v \rfloor$. The remaining requirements in the definition of a uniform predomain are easy to check. □

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

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

The proof of the proposition above gives:

Proposition 4.3.

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

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

Lemma 4.4. *The functions alloc, lookup, and assign are continuous.*

Notice that the definitions of *lookup* and *assign* depend on the projection functions π_n^S. Intuitively, if one for example looks up the approximate location $(l,\, n+1)$ in a store s, one only obtains the approximate element $\pi_{n+1}^S(s)(l)$ as result. It would not suffice to define, e.g., $lookup(\lambda_l^{n+1})(k)(s) = \bot$ for $l \in \mathrm{dom}(s)$, and hence avoid mentioning the projection functions: *lookup* would then not be continuous.

We are now ready to define the untyped semantics.

Definition 4.5. *Let t be a term and let X be a set of variables such that* $\mathrm{FV}(t) \subseteq X$. *The* untyped semantics of t with respect to X *is the continuous function $[\![t]\!]_X : V^X \to TV$ defined by induction on t in Figures 5 and 6.*

Definition 4.6. *Let t be a term with no free term variables or type variables. The* program semantics of t *is the element $[\![t]\!]^{\mathrm{p}}$ of Ans defined by*

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

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

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

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

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

Figure 4: Functions used for interpreting reference operations.

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

$$
\begin{aligned}
[\![x]\!]_X\rho &= \eta(\rho(x)) \\
[\![\overline{m}]\!]_X\rho &= \eta(in_{\mathbb{Z}}\, m) \\
[\![\mathsf{ifz}\, t_0\, t_1\, t_2]\!]_X\rho &= [\![t_0]\!]_X\rho \star \lambda v_0. \begin{cases} [\![t_1]\!]_X\rho & \text{if } v_0 = in_{\mathbb{Z}}\, 0 \\ [\![t_2]\!]_X\rho & \text{if } v_0 = in_{\mathbb{Z}}\, m \text{ where } m \neq 0 \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![t_1 \pm t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2. \begin{cases} \eta(in_{\mathbb{Z}}(m_1 \pm m_2)) \\ \qquad \text{if } v_1 = in_{\mathbb{Z}}\, m_1 \\ \qquad \text{and } v_2 = in_{\mathbb{Z}}\, m_2 \\ \mathit{error} \quad \text{otherwise} \end{cases} \\
[\![()]\!]_X\rho &= \eta(in_1\, *) \\
[\![(t_1,\, t_2)]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\, [\![t_2]\!]_X\rho \star \lambda v_2.\, \eta(in_{\times}(v_1, v_2)) \\
[\![\mathsf{fst}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_1) & \text{if } v = in_{\times}(v_1, v_2) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{snd}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v. \begin{cases} \eta(v_2) & \text{if } v = in_{\times}(v_1, v_2) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{void}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \mathit{error} \\
[\![\mathsf{inl}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{+}(\iota_1\, v)) \\
[\![\mathsf{inr}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\, \eta(in_{+}(\iota_2\, v))
\end{aligned}
$$

(Continued in Figure 6.)

Figure 5: Untyped semantics of terms.

where

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

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

Remark. The model in this section differs slightly from the BST model. First, the projection functions have been modified in order to ease calculations. Second, the semantic functions *lookup* and *assign* depend on projections of entire stores, not just projections of the individual values to be looked up or stored. This latter modification seems necessary when relations on stores

$$
\begin{aligned}
[\![\mathsf{case}\, t_0\ x_1.t_1\ x_2.t_2]\!]_X\rho &= [\![t_0]\!]_X\rho \star \lambda v_0.\ \begin{cases} [\![t_1]\!]_{X,x_1}(\rho[x_1 \mapsto v]) & \text{if } v_0 = in_+(\iota_1\, v) \\ [\![t_2]\!]_{X,x_2}(\rho[x_2 \mapsto v]) & \text{if } v_0 = in_+(\iota_2\, v) \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\lambda x.t]\!]_X\rho &= \eta(in_\to(\lambda v.\ [\![t]\!]_{X,x}(\rho[x \mapsto v]))) \\
[\![t_1\, t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\ [\![t_2]\!]_X\rho \star \lambda v_2.\ \begin{cases} g\, v_2 & \text{if } v_1 = in_\to g \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{fix}\, f.\lambda x.t]\!]_X\rho &= \eta(in_\to(\mathit{fix}(\lambda g^{V\to TV}.\ \lambda v.\ [\![t]\!]_{X,f,x}(\rho[f \mapsto in_\to g,\ x \mapsto v])))) \\
[\![\mathsf{fold}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \eta(in_\mu\, v) \\
[\![\mathsf{unfold}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \begin{cases} \eta(v_0) & \text{if } v = in_\mu\, v_0 \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\Lambda\alpha.t]\!]_X\rho &= \eta(in_\forall\, ([\![t]\!]_X\rho)) \\
[\![t\, [\tau]]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \begin{cases} c & \text{if } v = in_\forall\, c \\ \mathit{error} & \text{otherwise} \end{cases} \\
[\![\mathsf{ref}\, t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \mathit{alloc}\, v \\
[\![!t]\!]_X\rho &= [\![t]\!]_X\rho \star \lambda v.\ \mathit{lookup}\, v \\
[\![t_1 := t_2]\!]_X\rho &= [\![t_1]\!]_X\rho \star \lambda v_1.\ [\![t_2]\!]_X\rho \star \lambda v_2.\ \mathit{assign}\, v_1\, v_2
\end{aligned}
$$

Figure 6: Untyped semantics of terms (ctd.)

must be described by the more refined "worlds" in this article. Intuitively, the refined worlds allow binary relations on stores that are not simply composed from binary relations on the individual values in the stores.

5. Ultrametric spaces

We recall some basic definitions and properties about metric spaces. For more details, see for example de Bakker and de Vink [18] or the long version of the article about the BST model [16].

A metric space (X, d) is 1-*bounded* if $d(x, y) \leq 1$ for all x and y in X. An *ultrametric space* is a metric space that satisfies the 'ultrametric inequality,'

$$d(x, z) \leq \max(d(x, y), d(y, z)),$$

and not just the weaker triangle inequality (where one has + instead of max on the right-hand side). It might be helpful to think of the function d of an ultrametric space (X, d) not as a measure of (euclidean) distance between elements, but rather as a measure of the degree of similarity between elements.

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

A metric space is *complete* if every Cauchy sequence has a limit. By Banach's fixed-point theorem, every contractive function from a non-empty, complete metric space to itself has a unique fixed point.

For a given complete metric space, consider the function *fix* that maps every contractive operator to its unique fixed-point. On complete ultrametric spaces, *fix* is non-expansive in the following sense [5]:

Proposition 5.1. *Let (X, d) be a non-empty, complete ultrametric space. For all contractive functions f and g from (X, d) to itself, $d(\mathit{fix}\, f, \mathit{fix}\, g) \leq d(f, g)$.*

All the metric spaces we consider satisfy the following property:

Definition 5.2. *A metric space is* bisected *if all non-zero distances are of the form 2^{-n} for some natural number $n \geq 0$.*

The following notation is convenient when working with bisected metric spaces: in such a space, $x =_n y$ means that $d(x, y) \leq 2^{-n}$. Notice that

each relation $=_n$ is an equivalence relation. Here transitivity follows from the ultrametric inequality. Also, notice that a bisected metric space is one-bounded. In other words, the relation $x =_0 y$ always holds.

Proposition 5.3. *Let (X_1, d_1) and (X_2, d_2) be bisected metric spaces. A function $f : X_1 \to X_2$ is non-expansive if and only if*

$$x_1 =_n x_1' \implies f(x_1) =_n f(x_1')$$

holds for all $x_1, x_1' \in X_1$ and all natural numbers $n > 0$.

5.1. Categories of ultrametric spaces

Let $\mathsf{CBUlt_{ne}}$ be the category with non-empty, complete, 1-bounded ultrametric spaces as objects and non-expansive functions as morphisms. This category is cartesian closed [32, 16]; here one needs the ultrametric inequality. The terminal object is the one-point metric space. Binary products are defined in the natural way: $(X_1, d_1) \times (X_2, d_2) = (X_1 \times X_2,\ d_{X_1 \times X_2})$ where

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

The exponential $(X_1, d_1) \to (X_2, d_2)$ has the set of non-expansive functions from (X_1, d_1) to (X_2, d_2) as the underlying set, and the 'sup'-metric $d_{X_1 \to X_2}$ as distance function: $d_{X_1 \to X_2}(f, g) = \sup\{d_2(f(x), g(x)) \mid x \in X_1\}$. For both products and exponentials, limits are pointwise.

Let $\mathsf{PreCBUlt_{ne}}$ be the category of *pre-ordered*, non-empty, complete, 1-bounded ultrametric spaces. Objects of this category are pairs $(A, \leq)$ consisting of an object A of $\mathsf{CBUlt_{ne}}$ and a preorder $\leq$ on the underlying set of A such that the following condition holds: if $(a_n)_{n\in\omega}$ and $(b_n)_{n\in\omega}$ are converging sequences in A with $a_n \leq b_n$ for all n, then also $\lim_{n\to\infty} a_n \leq \lim_{n\to\infty} b_n$. The morphisms of the category are the non-expansive and monotone functions between such objects. We refer to the objects of this category as 'continuous preorders'.

Birkedal et al. [14] generalize the standard construction of solutions to recursive metric-space equations [6, 19] to a large class of categories with metric-space structure on each set of morphisms. In particular, one can solve recursive equations in the category $\mathsf{PreCBUlt_{ne}}$:

Definition 5.4. *A functor $F : \mathsf{PreCBUlt_{ne}}^{\mathrm{op}} \times \mathsf{PreCBUlt_{ne}} \to \mathsf{PreCBUlt_{ne}}$ is* locally non-expansive *if $d(F(f, g), F(f', g')) \leq \max(d(f, f'), d(g, g'))$ for all*

f,*f*′, *g*, *and g*′ *with appropriate domains and codomains. Stronger, F is* locally contractive *if there exists some* $c < 1$ *such that* $d(F(f,g), F(f',g')) \leq c \cdot \max(d(f,f'), d(g,g'))$ *for all f*,*f*′, *g*, *and g*′.

Theorem 5.5 ([14])**.** *Every locally contractive functor* $F : \mathsf{PreCBUlt_{ne}}^{\mathrm{op}} \times \mathsf{PreCBUlt_{ne}} \to \mathsf{PreCBUlt_{ne}}$ *has a unique fixed point: there exists an object Z of* $\mathsf{PreCBUlt_{ne}}$ *such that* $Z \cong F(Z,Z)$*, and if Z*′ *is another such object then* $Z \cong Z'$.

6. Bohr Relations on Uniform Domains and Predomains

We introduce the notion of Bohr relations on domains and predomains. And we equip spaces of such with complete bisected ultrametrics. To do this, we need additional structure, we require uniform domains and predomains.

First up, we introduce a Hausdorff metric on the admissible, downwards closed subsets of a uniform domain. This buys us the metric on Bohr relations on a uniform domain. Then we show that there is a simple bijective correspondence between chain-complete, downwards closed subsets of a uniform predomain and the admissible, downwards closed subsets of the uniform domain obtained by lifting. We define a metric on the former by means of this bijection and this gives us the metric on Bohr relations on a uniform predomain.

We apply standard metric space constructions such as Hausdorff distance and carving closed subsets out of complete metric spaces. As such, we save some mileage by appeal to standard (mostly completeness) results. But the route is sufficiently indirect that going directly for the Theorems 6.9 and 6.16 by brute force is a viable alternative; indeed, this was the approach of the authors in [12].

6.1. Distance on ADSub(D)

In the following subsection $(D, (\pi_n)_{n\in\omega})$ denotes an arbitrary uniform domain.

Based on the additional structure on the domain D given by the projections, we build a metric on D:

Proposition 6.1. *There is a (unique) complete, bisected, ultrametric* d_π *on D such that for any* $n \in \omega$ *and any two* $d, e \in D$ *we have*

$$d =_n e \iff \pi_n(d) = \pi_n(e).$$

Proof. We define the map $d_\pi : D \times D \to \mathbb{R}$ by mapping any two $d, e \in D$ to

$$d_\pi(d,e) = \begin{cases} 0 & \text{if } d = e \\ 2^{-\max\{n \in \omega \mid \pi_n(d) = \pi_n(e)\}} & \text{if } d \neq e. \end{cases}$$

Let us initially verify that this is well-defined, we need to show that for $d \neq e$ we have that the set $\{n \in \omega \mid \pi_n(d) = \pi_n(e)\}$ is non-empty and finite. The former is a consequence of having $\pi_0(d) = \bot = \pi_0(e)$. Note now that for $m \leq n$ we have that $\pi_n(d) = \pi_n(e)$ implies $\pi_m(d) = \pi_m(e)$ since we have $\pi_m(d) = \pi_{\min(m,n)}(d) = \pi_m(\pi_n(d)) = \pi_m(\pi_n(e)) = \pi_{\min(m,n)}(e) = \pi_m(e)$. If now the set in question was infinite, then all projections of d and e would agree and they would be equal, contradicting our assumption.

By similar reasoning we easily show that for any $n \in \omega$ and any two $d, e \in D$ we have $d =_n e$ iff $\pi_n(d) = \pi_n(e)$. The map d_π is bisected by construction and for any two $d, e \in A$ we easily have that $d = e$ iff $d_\pi(d,e) = 0$ and that $d_\pi(d,e) = d_\pi(e,d)$. It remains to prove the strong triangle inequality and completeness. In search for the former, we pick $d, e, f \in D$ and aim to prove

$$d_\pi(d,f) \leq \max(d_\pi(d,e), d_\pi(e,f)).$$

Without loss of generality we may assume $d \neq f$, $d \neq e$ and $e \neq f$. There are $n, m \in \omega$ such that $d_\pi(d,e) = 2^{-n}$ and $d_\pi(e,f) = 2^{-m}$, let $l = \min(n,m)$. But then $d =_l e$ and $e =_l f$ and so $\pi_l(d) = \pi_l(e) = \pi_l(f)$ and we have

$$d_\pi(d,f) \leq 2^{-l} = 2^{-\min(n,m)} = \max(2^{-n}, 2^{-m}) = \max(d_\pi(d,e), d_\pi(e,f)).$$

To prove completeness we take an arbitrary Cauchy sequence $(d_n)_{n \in \omega}$ in D, we must build an $d \in D$ such that $\lim_n d_n = d$. For each $m \in \omega$ we pick an $M_m \in \omega$ such that we for any $n \geq M_m$ have that $d_n =_m d_{M_m}$. We may without loss of generality assume that $M_m \leq M_{m+1}$ for all $m \in \omega$. Our candidate for the limit now is

$$d = \bigsqcup_{m \in \omega} \pi_m(d_{M_m}).$$

To verify that this least upper bound actually exists, we remark that for any $m \in \omega$ we have

$$\pi_m(d_{M_m}) = \pi_m(d_{M_{m+1}}) \sqsubseteq \pi_{m+1}(d_{M_{m+1}}).$$

To finally prove that d is the limit we take any $m \in \omega$ and note that for any $n \geq M_m$ we have that

$$\begin{aligned}
\pi_m(d) &= \pi_m\left(\bigsqcup_{o\in\omega} \pi_o(d_{M_o})\right) \\
&= \bigsqcup_{o\in\omega} \pi_{\min(m,o)}(d_{M_o}) \\
&= \bigsqcup_{o\geq m} \pi_m(d_{M_o}) \\
&= \bigsqcup_{o\geq m} \pi_m(d_{M_m}) \\
&= \pi_m(d_{M_m}) \\
&= \pi_m(d_n)
\end{aligned}$$

which as noted implies that $d =_m d_n$ and we are done. □

We recollect the notion of Hausdorff distance:

Definition 6.2. *The* Hausdorff *distance d_H between two non-empty subsets $X, Y \subseteq M$ of a 1-bounded metric space (M, d) is defined as follows:*

$$d_H(X,Y) = \max\left(\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\right)$$

The notion of Hausdorff distance is standard, cf. Definitions 2.2 and 2.6 and Lemma 2.7 of [18] for an expository presentation. We restrict to 1-bounded metric spaces to avoid dealing with unbounded suprema and also focus on non-empty sets to simplify the presentation.

The Hausdorff distance is not a metric on the entire set of non-empty subsets of M as a distance of zero may fail to imply equality. But if we restrict ourselves to the *closed*, non-empty subsets we get a proper metric and completeness carries over:

Proposition 6.3. *The Hausdorff distance is a 1-bounded metric on the set $\mathcal{P}_{ncl}(X)$ of non-empty and closed subsets of a 1-bounded metric space (X, d). $(\mathcal{P}_{ncl}(X), d_H)$ is ultrametric and complete if (X, d) is ultrametric and complete, respectively.*

These are textbook result, cf. Theorems 2.3 and 2.10 of [18].

Intuitively, the Hausdorff distance between X and Y is the least distance r such that for any $x \in X$ we can find $y \in Y$ with mutual distance no greater than r and vice versa. This intuition is captured in the following proposition under the assumption that the the underlying metric space is bisected:

Proposition 6.4. *Let (M, d) be a bisected metric space. Then for any non-empty $X, Y \subseteq M$ we have that $d_H(X, Y)$ is zero or of the form 2^{-n} for some $n \in \omega$ and for any $n \in \omega$ we get*

$$X =_n Y \iff \forall x \in X \exists y \in Y.\ x =_n y \wedge \forall y \in Y \exists x \in X.\ x =_n y.$$

Proof. To prove that for two non-empty $X, Y \subseteq M$ we have $d_H(X, Y) \in \{0\} \cup \{\, 2^{-m} \mid m \in \omega \,\}$ we simply observe that this set is closed under non-empty suprema and infima.

We now proceed to prove the biimplication. Pick non-empty $X, Y \subseteq M$ and $n \in \omega$ arbitrarily. To prove that the left hand side implies the right hand side we assume that $X =_n Y$, take arbitrary $x \in X$ and need to find $y \in Y$ with $x =_n y$. For the sake of deriving a contradiction we assume that this cannot be done, i.e., that for every $y \in Y$ we have $d(x, y) > 2^{-n}$. By the assumption of the proposition this would mean that $d(x, y) \geq 2^{-n+1}$ for all $y \in Y$ which would imply that

$$d_H(X, Y) \geq \sup_{z \in X} \inf_{y \in Y} d(z, y) \geq \inf_{y \in Y} d(x, y) \geq 2^{-n+1}$$

which contradicts our assumption that $X =_n Y$. Proving the other conjunct proceeds similarly and the reverse implication is standard and does not rely on the special form of the metric. □

We are now in a position to define a distance on the set $ADSub(D)$ of admissible and downwards closed subsets of D as the Hausdorff distance on top of the distance d_π on D.

Proposition 6.5. *Any downwards closed and chain-complete subset of D is a closed subset of the metric space (D, d_π).*

Proof. Let $X \subseteq D$ be a downwards closed and chain-complete subset of D. Let $(x_m)_{m \in \omega}$ be a sequence in X with $\lim_m x_m = x$ for some $x \in D$. We must prove that $x \in X$ too. We know that

$$x = \bigsqcup_m \pi_m(x)$$

so by chain-completeness of X it suffices to show that $\pi_m(x) \in X$ holds for any $m \in \omega$. But this is a consequence of X being downwards closed since for any $m \in \omega$ there is $M_m \in \omega$ with $x =_m x_{M_m}$ which implies that $\pi_m(x) = \pi_m(x_{M_m}) \sqsubseteq x_{M_m} \in X$. □

Any admissible subset of D is non-empty as it contains the least element and so $ADSub(D) \subseteq \mathcal{P}_{ncl}(D)$. By Propositions 6.3 and Proposition 6.4 we know that d_H is a complete, bisected ultrametric on $\mathcal{P}_{ncl}(D)$. Hence d_H is a bisected ultrametric on $ADSub(D)$ too, where we overload d_H to mean both the Hausdorff distance on $\mathcal{P}_{ncl}(D)$ and its restriction to $ADSub(D)$. To obtain completeness we need the following:

Proposition 6.6. *The set $ADSub(D)$ is a closed subset of the metric space $(\mathcal{P}_{ncl}(D), d_H)$.*

Proof. Take some sequence $(X_m)_{m\in\omega}$ in $ADSub(D)$ and assume that $\lim_m X_m = X$ for some $X \in \mathcal{P}_{ncl}(D)$, we must prove that $X \in ADSub(D)$ too.

Let us initially prove that the least element $\bot \in D$ is in X. For any $m \in \omega$ there is M_m such that $X_{M_m} =_m X$. And as $\bot \in X_{M_m}$ we know that there is a member, x_m say, of X with $\bot =_m x_m$ by Proposition 6.4. But then clearly $\lim_m x_m = \bot$ and since X was closed we have $\bot \in X$.

We now prove X chain-complete. We take an increasing chain $(x_m)_{m\in\omega}$ in X and aim to show that $x = \sqcup_m x_m \in X$. Take any $n \in \omega$, there is M_n such that $X_{M_n} =_n X$ and so the increasing chain $(\pi_n(x_m))_{m\in\omega}$ is in X_{M_n} as X_{M_n} was downwards closed. But X_{M_n} was chain-complete too and hence $\pi_n(x) \in X_{M_n}$ and so we may find $y_n \in X$ with $y_n =_n \pi_n(x) =_n x$. Clearly $\lim_n y_n = x$ and since X was closed we have $x \in X$.

Finally take $x, y \in D$ with $x \sqsubseteq y$ and $y \in X$, we need to show $x \in X$. For any $m \in \omega$, there is M_m such that $X_{M_m} =_m X$ and hence $\pi_m(y) \in X_{M_m}$ as X_{M_m} was downwards closed. But then $\pi_m(x) \in X_{M_m}$ too as X_{M_m} was downwards closed and we proceed as above. □

In summa, we have the following:

Corollary 6.7. *There is a (unique) complete, bisected ultrametric d_H on $ADSub(D)$ such that for any two $X, Y \in ADSub(D)$ and any $n \in \omega$ we have*

$$X =_n Y \iff \pi_n(X) \subseteq Y \wedge \pi_n(Y) \subseteq X.$$

6.2. Bohr Relations on Uniform Domains

Definition 6.8 (Bohr Relation)**.** *A relation $R \subseteq D \times D$ on a domain D is called a* Bohr relation *if for any $e \in D$ we have that*

$$R(-, e) = \{d \mid (d, e) \in R\}$$

is admissible and downwards closed.

Theorem 6.9. *Let $(D, (\pi_n)_{n\in\omega})$ be a uniform domain. There is a (unique) complete, bisected ultrametric d_B on $BohrRel(D)$ such that for any two $R, S \in BohrRel(D)$ and any $n \in \omega$ we have*

$$R =_n S \iff \forall e \in D.\ \pi_n(R(-, e)) \subseteq S(-, e) \wedge \pi_n(S(-, e)) \subseteq R(-, e)$$

The proof proceeds along the lines of the proof of Theorem 6.16 only we appeal to Corollary 6.7 instead of Proposition 6.13.

It is not hard to prove the following:

Proposition 6.10. *Let $(D, (\pi_n)_{n\in\omega})$ be a uniform domain. Let $(R_n)_{n\in\omega}$ and $(S_n)_{n\in\omega}$ be sequences in $BohrRel(D)$ such that $\lim_n R_n = R$ and $\lim_n S_n = S$ for R and S also in $BohrRel(D)$. We then have that*

$$(\forall n \in \omega.\ R_n \subseteq S_n) \implies R \subseteq S.$$

Summing up, we have that the Bohr relations on a uniform domain equipped with the metric from Theorem 6.9 above and ordered by set-theoretic inclusion is an object of $\mathsf{PreCBUlt}_{\mathsf{ne}}$; see also Subsection 5.1.

6.3. Distance on $CDSub(A)$

In the following subsection, $(A, (\pi_n)_{n\in\omega})$ denotes an arbitrary uniform predomain.

Now let us return to uniform predomains. Recall our goal of obtaining a metric on the set of chain-complete and downwards closed subsets of a uniform predomain. We employ lifting to build a a uniform domain from a given uniform predomain and then apply the above theory.

It is well known that we may lift a predomain A to a domain $A_\perp$ by introducing a least element. This idea extends naturally to build uniform domains from uniform predomains:

Proposition 6.11. *Define, for $m \in \omega$, a new projection $\pi'_m : A_\perp \to A_\perp$ by*

$$\pi'_m(d) = \begin{cases} \pi_m(a) & \text{if } d = \lfloor a \rfloor \\ \perp & \text{if } d = \perp \end{cases}$$

for each $d \in A_\perp$. Then $(A_\perp, (\pi'_m)_{m\in\omega})$ is a uniform domain.

Proof. That $A_\perp$ is a domain and the new projections continuous are basic results of domain theory, see, e.g., section 8.3.4 of [35]. As the projections are strict by definition, it remains to verify the four defining axioms of uniform domains under the assumption of the axioms of uniform predomains:

For any $m \in \omega$ we need initially to show $\pi'_m \leq \pi'_{m+1}$. We prove this pointwise so we take $d \in A_\perp$ arbitrary. We may without loss of generality assume $d = \lfloor a \rfloor$ for some $a \in A$ and we have $\pi'_m(d) = \pi_m(a) \leq \pi_{m+1}(a) = \pi'_{m+1}(d)$.

We need to show $\sqcup_m \pi'_m = id_{A_\perp}$. As above, we take $d \in A_\perp$ arbitrary and discharge the case $d = \perp$ easily. So assume $d = \lfloor a \rfloor$ for some $a \in A$ we get that

$$\left(\bigsqcup_m \pi'_m\right)(d) = \bigsqcup_m \pi_m(a) = \left(\bigsqcup_m \pi_m\right)(a) = \lfloor a \rfloor = d.$$

For the third axiom we pick $m, n \in \omega$ and must show that $\pi'_m \circ \pi'_n = \pi'_n \circ \pi'_m = \pi'_{\min(m,n)}$. We prove this pointwise, so we take take $d \in A_\perp$ and may without loss of generality assume that $d = \lfloor a \rfloor$ for some $a \in A$. But then we need to show that

$$\pi'_m(\pi_n(a)) = \pi'_n(\pi_m(a)) = \pi_{\min(m,n)}(a)$$

which coincides with the third axiom of uniform predomains.

The fourth and final axiom requires π'_0 to be constant bottom which is obviously true as it holds for π_0 by assumption. □

We now give a bijective correspondence between the set $CDSub(A)$ of chain-complete, downwards closed subsets of A and the set $ADSub(A_\perp)$. For $X \subseteq A$ we let $X_\perp$ denote $\{\lfloor x \rfloor \mid x \in X\} \cup \{\perp\}$; this provides the bijection:

Proposition 6.12. *The map $(-)_\perp : \mathcal{P}(A) \to \mathcal{P}(A_\perp)$ establishes a bijective correspondence between $CDSub(A)$ and $ADSub(A_\perp)$.*

Proof. Take $X \in CDSub(A)$, we must prove that $X_\perp \in ADSub(A_\perp)$. To prove chain-completeness we take an increasing chain $(d_n)_{n\in\omega}$ in $X_\perp$ and we must show $\bigsqcup_n d_n \in X_\perp$ too. We may without loss of generality assume that no elements of the chain are bottom and hence we can choose an $x_n \in X$ with $d_n = \lfloor x_n \rfloor$ for all $n \in \omega$. But then $(x_n)_{n\in\omega}$ is an increasing chain too and we have $\bigsqcup_n x_n \in X$ by assumption. By continuity we get

$$\bigsqcup_n d_n = \bigsqcup_n \lfloor x_n \rfloor = \left\lfloor \bigsqcup_n x_n \right\rfloor \in X_\perp$$

and since $\perp \in X_\perp$ by definition we have proved admissibility. Downwards closure is simple, take $d, e \in A_\perp$ with $d \sqsubseteq e$ and $e \in X_\perp$, we must show $d \in X_\perp$ too. If $d = \perp$ we are done, otherwise there is $x \in A$ and $y \in X$ with $d = \lfloor x \rfloor$ and $e = \lfloor y \rfloor$ and hence $x \leq y$ which means that $x \in X$ too.

For any two $X, Y \in \mathcal{P}(A)$ we have that $X_\perp = Y_\perp$ readily implies $X = Y$. It remains to show that for any $X \in ADSub(A_\perp)$ there is an $Y \in CDSub(A)$ with $X = Y_\perp$. Unsurprisingly, we aim for

$$Y = \{a \in A \mid \lfloor a \rfloor \in X\}$$

which obviously has $Y_\perp = X$ since we must have $\perp \in X$. Continuity of $\lfloor - \rfloor$ immediately yields that $Y \in CDSub(A)$ and we are done. □

Proposition 6.13. *There is a (unique) complete, bisected ultrametric $d_\perp$ on $CDSub(A)$ such that for any two $X, Y \in CDSub(A)$ and any $n \in \omega$ we have*

$$X =_n Y \iff \pi_n(X) \subseteq Y_\perp \wedge \pi_n(Y) \subseteq X_\perp.$$

Proof. Given the preceding development, it should come as no surprise that we lift the uniform predomain to obtain the uniform domain $(A_\perp, (\pi'_m)_{m\in\omega})$ by Proposition 6.11. $A_\perp$ is endowed with the complete, bisected ultrametric d_π of Proposition 6.1 and $ADSub(A_\perp)$ with the complete, bisected ultrametric d_H of Corollary 6.7. For any two $X, Y \in CDSub(A)$ we now define

$$d_\perp(X, Y) = d_H(X_\perp, Y_\perp),$$

which yields a complete, bisected ultrametric on $CDSub(A)$ by Proposition 6.12. Now take any two $X, Y \in CDSub(A)$ and any $n \in \omega$, we must prove that

$$X =_n Y \iff \pi_n(X) \subseteq Y_\perp \wedge \pi_n(Y) \subseteq X_\perp.$$

Assume that we have $X =_n Y$, i.e., that $X_\perp =_n Y_\perp$. We take $x \in X$ and must prove that $\pi_n(x) \in Y_\perp$. We have $\lfloor x \rfloor \in X_\perp$ and hence there is $y \in Y_\perp$ such that $\lfloor x \rfloor =_n y$. As $Y_\perp$ is downwards closed we have $\pi'_n(y) \in Y_\perp$ and so

$$\pi_n(x) = \pi'_n(\lfloor x \rfloor) = \pi'_n(y)$$

and we have proved the desired; proving the other conjunct proceeds similarly.

Going for the other implication, we assume that $\pi_n(X) \subseteq Y_\perp \wedge \pi_n(Y) \subseteq X_\perp$ and must prove $X =_n Y$, i.e., that $X_\perp =_n Y_\perp$. So take $x \in X_\perp$, we must produce $y \in Y_\perp$ with $x =_n y$. We may without loss of generality assume $x \neq \perp$. So there is $x' \in X$ with $x = \lfloor x' \rfloor$ and our assumption buys us that $\pi'_n(x) = \pi_n(x') \in Y_\perp$. But we obviously have $\pi'_n(x) =_n x$ and are done; the symmetric property is proved similarly. □

6.4. Bohr Relations on Uniform Predomains

Definition 6.14 (Bohr Relation)**.** *A relation $R \subseteq A \times A$ on a predomain A is called a* Bohr relation *if for any $b \in A$ we have that*

$$R(-, b) = \{a \mid (a, b) \in R\}$$

is chain-complete and downwards closed.

As the defining property of Bohr relations is preserved by set-theoretic intersection, we easily get the following closure operator:

Proposition 6.15. *For any relation $R \subseteq A \times A$ we have that*

$$\overline{R} = \bigcap_{R \subseteq S \subseteq A \times A,\ S \text{ Bohr}} S$$

is a Bohr relation, furthermore it is least such that contain R.

Theorem 6.16. *Let $(A, (\pi_n)_{n \in \omega})$ be a uniform predomain. There is a (unique) complete, bisected ultrametric d_B on $BohrRel(A)$ such that for any two $R, S \in BohrRel(A)$ and any $n \in \omega$*

$$\begin{aligned} R =_n S \iff & \\ & \big[\forall (a,b) \in R.\ \pi_n(a) = \perp \vee (\exists a' \in A.\ \pi_n(a) = \lfloor a' \rfloor \wedge (a', b) \in S)\big] \wedge \\ & \big[\forall (a,b) \in S.\ \pi_n(a) = \perp \vee (\exists a' \in A.\ \pi_n(a) = \lfloor a' \rfloor \wedge (a', b) \in R)\big]. \end{aligned}$$

Proof. Consider the space of all functions $A \to CDSub(A)$. We may define a distance d_F between any two members $f, g \in A \to CDSub(A)$ of this set by setting

$$d_F(f,g) = \sup_{b \in A} d_\perp(f(b), g(b))$$

and it is a textbook result that this constitutes a complete ultrametric as this is the case for $CDSub(D)$ by Proposition 6.13. See, e.g., Lemmas 1.24 and 1.28 of [18] for details. As $d_\perp$ is bisected and the set $\{0\} \cup \{2^{-n} \mid n \in \omega\}$ is closed under non-empty suprema we have that d_F is bisected as well, and we may replace the supremum by the maximum in the above definition. We now define the map $\Phi : BohrRel(A) \to (A \to CDSub(A))$ by setting

$$\Phi(R)(b) = R(-, b)$$

for any $R \in BohrRel(A)$ and any $b \in A$. This is well-defined by the definition of Bohr relations and furthermore a bijection. We define the distance d_B between two $R, S \in BohrRel(A)$ by setting

$$d_B(R,S) = d_F(\Phi(R), \Phi(S))$$

and by a bijection argument we have that d_B is a complete, bisected ultrametric on $BohrRel(A)$.

Take now two $R, S \in BohrRel(A)$ and any $n \in \omega$ and assume that we have $R =_n S$. Take $(a, b) \in R$, assume that $\pi_n(a) = \lfloor a' \rfloor$ for some $a' \in A$, we must prove that $(a', b) \in S$. By definition we have $\Phi(R) =_n \Phi(S)$ which means that

$$R(-, b) = \Phi(R)(b) =_n \Phi(S)(b) = S(-, b)$$

and since $a \in R(-, b)$ we have $\pi_n(a) \in (S(-, b))_\perp$ by Proposition 6.13. But since $\pi_n(a) = \lfloor a' \rfloor$ we must have $a' \in S(-, b)$, i.e., $S(a', b)$. Proving the second conjunct of the right hand side of the biimplication proceeds similarly.

So assume now that the right hand side of the desired biimplication holds, we must prove that $R =_n S$. This means proving $\Phi(R) =_n \Phi(S)$ which again comes down to proving that for any $b \in A$ we have

$$R(-, b) = \Phi(R)(b) =_n \Phi(S)(b) = S(-, b)$$

So take $a \in R(-, b)$, i.e., $R(a, b)$ holds. We must by Proposition 6.13 prove that $\pi_n(a) \in (S(-, b))_\perp$ but this is exactly what the first disjunct of the right hand side gives us. And the second disjunct similarly buys us the converse implication. □

As was the case for uniform domains, we can prove the following:

Proposition 6.17. *Let $(A, (\pi_n)_{n\in\omega})$ be a uniform predomain. Let $(R_n)_{n\in\omega}$ and $(S_n)_{n\in\omega}$ be sequences in $BohrRel(A)$ such that $\lim_n R_n = R$ and $\lim_n S_n = S$ for R and S also in $BohrRel(A)$. We then have that*

$$(\forall n \in \omega.\, R_n \subseteq S_n) \implies R \subseteq S.$$

As concluded for Bohr Relations on uniform domains, we also have that the Bohr relations on a uniform predomain equipped with the metric from Theorem 6.16 above and ordered by set-theoretic inclusion is an object of $\mathsf{PreCBUlt}_{\mathsf{ne}}$; we refer to Subsection 5.1 for a definition of this category.

7. Building Worlds

In this section we build the space of worlds to be used in our Kripke logical relation. The space of worlds is obtained using Theorem 5.5, i.e., as the fixed point of a functor on certain pre-ordered metric spaces.

7.1. M-categories

Since we aim to apply Theorem 5.5 we need to keep track of whether the functors we build are locally contractive. To that end, it is most convenient to introduce the general M-categories of Birkedal et al. [14]; these are categories such as $\mathsf{CBUlt}_{\mathsf{ne}}$ or $\mathsf{PreCBUlt}_{\mathsf{ne}}$ that have a metric-space structure on each hom-set. This subsection can be skipped on a first reading: one can then read the definitions of the functors in the following sections while taking for granted that they do satisfy the required technical conditions.

Definition 7.1. *An M-*category *is a category $\mathcal{C}$ where each hom-set $\mathcal{C}(A, B)$ is equipped with a distance function turning it into a non-empty, complete, 1-bounded ultrametric space, and where each composition function*

$$\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$$

is non-expansive with respect to these metrics. (Here the domain of such a composition function is given the product metric.)

In other words, an M-category is a category where each hom-set is equipped with a metric which turns it into an object in $\mathsf{CBUlt}_{\mathsf{ne}}$; furthermore, each composition function must be a morphism in $\mathsf{CBUlt}_{\mathsf{ne}}$. We observe that if $\mathcal{C}$ is an M-category, then so are $\mathcal{C}^{\mathrm{op}}$ (with the same metric on each hom-set as in $\mathcal{C}$) and $\mathcal{C}^{\mathrm{op}} \times \mathcal{C}$ (with the product metric on each hom-set)

Proposition 7.2 ([14]). $\mathsf{CBUlt_{ne}}$ *and* $\mathsf{PreCBUlt_{ne}}$ *are M-categories when each hom-set is given the 'sup'-metric:*

$$d_{\mathcal{C}(X_1,X_2)}(f,g) = \sup\{d_{X_2}(f(x),g(x)) \mid x \in X_1\}$$

Definition 7.3. *A functor $F : \mathcal{C} \to \mathcal{D}$ between M-categories $\mathcal{C}$ and $\mathcal{D}$ is called* locally ε-Lipschitz *for some $\varepsilon \geq 0$ if, for all morphisms $f, g : A \to B$ of $\mathcal{C}$, we have*

$$d(F(f),F(g)) \leq \varepsilon \cdot d(f,g),$$

where the leftmost distance is in the hom-set $\mathcal{D}(F(A),F(B))$ and the rightmost is in the hom-set $\mathcal{C}(A,B)$.

We also say that the functor has the *local Lipschitz constant* ε. Notice that being locally contractive and locally non-expansive comes down to having a local Lipschitz constant strictly less than one and less than or equal to one, respectively.

The following are compositional rules for computing the local Lipschitz constant. They are stated in their most general form, notice in particular that the shrinking functor of Proposition 7.9 cannot readily be generalized to arbitrary M-categories. We omit all proofs as they are quite simple.

Proposition 7.4 (Identity Functor). *Let $\mathcal{C}$ be an M-category. The identity functor on $\mathcal{C}$ is locally 1-Lipschitz.*

Proposition 7.5 (Constant Functor). *Let $\mathcal{C}$ and $\mathcal{D}$ be M-categories and let D be a fixed object of $\mathcal{D}$. The constant functor that maps objects and morphisms of $\mathcal{C}$ to D and 1_D respectively is locally 0-Lipschitz.*

Proposition 7.6 (Functor Pairing). *Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ be M-categories and let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{E}$ be locally ε-Lipschitz and locally δ-Lipschitz respectively. Then $\langle F, G\rangle : \mathcal{C} \to \mathcal{D} \times \mathcal{E}$ is locally* $\max\{\delta,\varepsilon\}$*-Lipschitz.*

Proposition 7.7 (Hom Functor). *Let $\mathcal{C}$ be an M-category. The hom functor $(-) \to (-) : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathsf{CBUlt_{ne}}$ defined in the standard way is locally 1-Lipschitz.*

Note that this is not, in general, an exponential. Rather, we just return the set of morphisms equipped with the metric structure it has according to the definition of M-categories.

Proposition 7.8 (Functor Composition)**.** *Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ be M-categories and let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{E}$ be locally ε-Lipschitz and locally δ-Lipschitz respectively. Then $G \circ F : \mathcal{C} \to \mathcal{E}$ is locally $\delta\varepsilon$-Lipschitz.*

Proposition 7.9 (Shrinking Functor)**.** *For any $0 < \varepsilon \leq 1$ we have that the functor $\varepsilon \cdot (-) : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ that multiplies all distances by ε is locally ε-Lipschitz.*

Proposition 7.10 (Product Functor)**.** *The standard metric product functor $(-) \times (-) : \mathsf{CBUlt}_{\mathsf{ne}} \times \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is locally 1-Lipschitz.*

Proposition 7.11 (Finite Maps Functor)**.** *Let X be an arbitrary set. The functor $X \rightharpoonup_{fin} (-) : \mathsf{CBUlt}_{\mathsf{ne}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ is locally 1-Lipschitz. It assigns the distance 1 to maps with different domains and the pointwise maximum otherwise, the action on morphisms is the obvious.*

7.2. The Space $\mathcal{W}$ of Worlds

We now turn to constructing the space of worlds. First, for any set X we let $\mathcal{L}(X)$ denote the set $\{(x, L) \in X \times \mathcal{P}(X) \mid x \in L\}$ of pairs of elements of X and subsets of X such that the former belongs to the latter. It is obviously non-empty provided that X is.

Proposition 7.12. *The functor $I : \mathsf{PreCBUlt}_{\mathsf{ne}}{}^{\mathsf{op}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ defined by*

$$\mathcal{L}(\mathcal{P}(V)) \times \left(\mathcal{P}(V) \to \frac{1}{2} \left[- \to \mathit{BohrRel}(S) \right] \right)$$

is locally $\frac{1}{2}$-Lipschitz. And so is the functor $W : \mathsf{PreCBUlt}_{\mathsf{ne}}{}^{\mathsf{op}} \to \mathsf{CBUlt}_{\mathsf{ne}}$ defined by

$$\mathcal{P}(\mathit{Loc}) \times \mathcal{P}(\mathit{Loc}) \times (\mathbb{N} \rightharpoonup_{\mathit{fin}} I(-)) .$$

The proof is a simple application of the above propositions. But the definitional one liners call for a few comments: We implicitly equip $\mathcal{L}(\mathcal{P}(V))$, $\mathcal{P}(V)$ and $\mathcal{P}(\mathit{Loc})$ with the discrete metric and, as such, consider them objects of $\mathsf{CBUlt}_{\mathsf{ne}}$. The rightmost arrow in the definition of the functor I is the standard hom functor on $\mathsf{PreCBUlt}_{\mathsf{ne}}$, i.e., it is the set of all non-expansive and monotone functions equipped with the supremum metric (see Proposition 7.7). And the arrow preceding that is the standard hom functor on $\mathsf{CBUlt}_{\mathsf{ne}}$ but reduces to the full function space because of the discrete metric on $\mathcal{P}(V)$. S is just the uniform predomain of states as defined earlier.

We need some notation to work with the output of the functors; we strive for compatibility with the nomenclature of LADR [22]. Let A be an object of $\mathsf{PreCBUlt_{ne}}$ and let $\Delta \in W(A)$, we write

$$\Delta = (\Delta.\varsigma_1, \Delta.\varsigma_2, \Delta.\mathcal{I})$$

for $\Delta.\varsigma_1, \Delta.\varsigma_2 \subseteq Loc$ and $\Delta.\mathcal{I} \in \mathbb{N} \rightharpoonup_{fin} I(A)$. Intuitively, a *world*[2] Δ oversees pairs of stores. It has a set of *left locations* $\Delta.\varsigma_1$ and *right locations* $\Delta.\varsigma_2$ that keep track of the allocated locations in the left and right hand side stores, respectively. Also it has an *island map* $\Delta.I$ that holds islands, each of which manages separate parts of the stores.

For $\Theta \in I(A)$ we write

$$\Theta = (\Theta.CP, \Theta.PL, \Theta.HL)$$

for $\Theta.CP \subseteq V$, $\Theta.PL \subseteq \mathcal{P}(V)$ and $\Theta.HL \in \mathcal{P}(V) \to \frac{1}{2}\,[A \to BohrRel(S)]$. An *island*[3] Θ has three components, the *current population* $\Theta.CP$, the *population law* $\Theta.PL$ and the *heap law* $\Theta.HL$. The population captures the current state of the island, it may vary over time, but only within the bonds given by the population law: because of our use of $\mathcal{L}(\mathcal{P}(V))$ instead of $\mathcal{P}(V) \times \mathcal{P}(\mathcal{P}(V))$ in the definition of the functor I we get that $\Theta.CP \in \Theta.PL$ always holds. The heap law provides the set of pairs of heaps that the island accepts; the idea is to feed it the current population and the current world.

Returning to the technical development, we have a locally $\frac{1}{2}$-Lipschitz functor $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ and are almost ready to apply the fixed-point existence theorem that will give us the space of worlds $\mathcal{W}$. First we must remedy one shortcoming, though: the functor maps into $\mathsf{CBUlt_{ne}}$ so we must equip the images under W of objects with continuous preorders; this will give us a functor that maps into $\mathsf{PreCBUlt_{ne}}$.

Definition 7.13. *Let A be an object of $\mathsf{PreCBUlt_{ne}}$. For any two $\Delta_1, \Delta_2 \in W(A)$ we say that Δ_2* extends *Δ_1 and we write $\Delta_1 \sqsubseteq \Delta_2$ if we have*

$$\Delta_1.\varsigma_1 \subseteq \Delta_2.\varsigma_1 \wedge \Delta_1.\varsigma_2 \subseteq \Delta_2.\varsigma_2 \wedge \forall n \in \mathrm{dom}(\Delta_1.\mathcal{I}).\ \Delta_1.\mathcal{I}(n) \sqsubseteq \Delta_2.\mathcal{I}(n),$$

[2]Outside of this subsection, we speak of worlds only as the results of applying W to the specific fixed-point $\hat{\mathcal{W}}$ that we produce below, not to an arbitrary object of $\mathsf{PreCBUlt_{ne}}$.

[3]As for worlds, an island, in general, belongs to the result of applying I to the specific fixed point $\hat{\mathcal{W}}$ built below.

where we write $\Theta_1 \sqsubseteq \Theta_2$ *for any two* $\Theta_1, \Theta_2 \in I(A)$ *if we have*

$$\Theta_1.CP \subseteq \Theta_2.CP \wedge \Theta_1.PL = \Theta_2.PL \wedge \Theta_1.HL = \Theta_2.HL.$$

On the conceptual level, world extension has two separate components. We may add new islands to the island map, often to manage newly allocated store; there are no restrictions on these new islands with respect to the old world. This is known as *width extension* in LADR. But the existing islands can also change: their populations may grow within the bounds of the population law. The population and heap laws are themselves immutable, but as we apply the heap law to the current population, it may permit different pairs of stores in the old and new worlds. Such population growth loosely corresponds to a state change of some existing object in the store; it is termed *depth extension* in LADR.

Proposition 7.14. *For any object* A *of* $\mathsf{PreCBUlt}_{\mathsf{ne}}$ *we have that the above ordering on* $W(A)$ *is a continuous preorder; for any morphism* $f : B \to A$ *of* $\mathsf{PreCBUlt}_{\mathsf{ne}}$ *we have that* $W(f) : W(A) \to W(B)$ *is monotone with respect to this ordering.*

Proof. The ordering is easily a preorder. To show that it is a continuous preorder we take sequences $(\Delta_n)_{n\in\omega}$ and $(\Gamma_n)_{n\in\omega}$ in $W(A)$ with limits $\lim_n \Delta_n = \Delta$ and $\lim_n \Gamma_n = \Gamma$ such that $\Delta_n \sqsubseteq \Gamma_n$ for all $n \in \omega$; we must show that we have $\Delta \sqsubseteq \Gamma$ too. We now pick an $m \in \omega$ such that $\Delta_m =_1 \Delta$ and that $\Gamma_m =_1 \Gamma$. But then by our construction we get

$$\Delta.\varsigma_1 = \Delta_m.\varsigma_1 \subseteq \Gamma_m.\varsigma_1 = \Gamma.\varsigma_1,$$

and by a similar argument we get that $\Delta.\varsigma_2 \subseteq \Gamma.\varsigma_2$ and $\mathrm{dom}(\Delta.\mathcal{I}) \subseteq \mathrm{dom}(\Gamma.\mathcal{I})$. Also, for any $n \in \mathrm{dom}(\Delta.\mathcal{I})$ we have that $\Delta_m.\mathcal{I}(n) =_1 \Delta.\mathcal{I}(n)$ and $\Gamma_m.\mathcal{I}(n) =_1 \Gamma.\mathcal{I}(n)$. But then

$$\Delta.\mathcal{I}(n).CP = \Delta_m.\mathcal{I}(n).CP \subseteq \Gamma_m.\mathcal{I}(n).CP = \Gamma.\mathcal{I}(n).CP$$

and also $\Delta.\mathcal{I}(n).PL = \Gamma.\mathcal{I}(n).PL$. Assume now that we have $\Delta.\mathcal{I}(n).HL \neq \Gamma.\mathcal{I}(n).HL$ for some $n \in \mathrm{dom}(\Delta.\mathcal{I})$, this means that we can pick $l \in \omega$ such that $\Delta.\mathcal{I}(n).HL \neq_l \Gamma.\mathcal{I}(n).HL$. Pick $k \in \omega$ such that $\Delta_k =_l \Delta$ and that $\Gamma_k =_l \Gamma$. But then

$$\Delta.\mathcal{I}(n).HL =_l \Delta_k.\mathcal{I}(n).HL = \Gamma_k.\mathcal{I}(n).HL =_l \Gamma.\mathcal{I}(n).HL$$

which is a contradiction.

We proceed to prove the second property. For $f : B \to A$ in $\mathsf{PreCBUlt_{ne}}$ and $\Delta \in W(A)$ arbitrary we can write out the action of the functor W on the morphism f as follows:

$$W(f)(\Delta.\varsigma_1, \Delta.\varsigma_2, \Delta.\mathcal{I}) = \big(\Delta.\varsigma_1, \Delta.\varsigma_2, \lambda n \in \mathrm{dom}(\Delta.\mathcal{I}).\ I(f)(\Delta.\mathcal{I}(n))\big).$$

For $\Theta \in I(A)$ arbitrary we can similarly write out the action of the functor I on the morphism f as follows:

$$I(f)(\Theta.CP, \Theta.PL, \Theta.HL) = \big(\Theta.CP, \Theta.PL, F(f)(\Theta.HL)\big),$$

where $F : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ is a shorthand for the component functor $\mathcal{P}(V) \to \frac{1}{2}(- \to \mathit{BohrRel}(S))$. From these observations it is immediate that $W(f)$ is monotone with respect to the above ordering. □

Corollary 7.15. *We may extend $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{CBUlt_{ne}}$ to a locally $\frac{1}{2}$-Lipschitz functor $W : \mathsf{PreCBUlt_{ne}}^{\mathsf{op}} \to \mathsf{PreCBUlt_{ne}}$ by equipping the images of objects with the continuous preorder from Definition 7.13.*

Definition 7.16 (Worlds)**.** *Let $\hat{\mathcal{W}}$ be an object such that $Sq : W(\hat{\mathcal{W}}) \cong \hat{\mathcal{W}}$ holds in $\mathsf{PreCBUlt_{ne}}$; existence (and uniqueness up to isomorphism) is guaranteed by Theorem 5.5. We write $\mathcal{W}$ for $W(\hat{\mathcal{W}})$.*

We conclude with a remark on the metric on worlds. Reasoning in the finished model often require us to define non-expansive maps out of the space $\mathcal{W}$ of worlds, this is the case, e.g., when we build new types as well as heap laws for new islands. An example of this is found in Section 10. It is worthwhile to note, that in many cases we have non-expansiveness for free.

For any two worlds $\Delta_1, \Delta_2 \in \mathcal{W}$ with $\Delta_1 =_1 \Delta_2$, it is immediate by our use of discrete metric spaces and the finite maps functor in the construction of the functor W that we have

$$\Delta_1.\varsigma_1 = \Delta_2.\varsigma_1, \quad \Delta_1.\varsigma_2 = \Delta_2.\varsigma_2, \quad \mathrm{dom}(\Delta_1.\mathcal{I}) = \mathrm{dom}(\Delta_2.\mathcal{I})$$

and for any n in the shared domain of the island maps we have

$$\Delta_1.\mathcal{I}(n).CP = \Delta_2.\mathcal{I}(n).CP, \quad \Delta_1.\mathcal{I}(n).PL = \Delta_2.\mathcal{I}(n).PL,$$

again because of our use of the discrete metric on $\mathcal{L}(\mathcal{P}(V))$. This means that we cannot invalidate non-expansiveness by inspection of these components, or, phrased differently, if we never 'project out' any heap laws then we have non-expansiveness automatically. If, however, we make use of the heap laws, then we must proceed with caution; see, e.g., the proof Proposition 8.2 for an example of this.

8. Logical Relation

We now construct a Kripke logical relation that uses the space of worlds $\mathcal{W}$ obtained above. First up is the definition of types:

Definition 8.1 (Types)**.** *The space of types is* $\mathcal{T} = \mathcal{W} \to_{mon} \mathit{BohrRel}(V)$*, i.e., the set of non-expansive and monotone functions from* $\mathcal{W}$ *to* $\mathit{BohrRel}(V)$*. It comes equipped with the supremum metric, i.e., for* $\mu, \nu \in \mathcal{T}$ *and* $n \in \omega$ *we have*

$$\mu =_n \nu \iff \forall \Delta \in \mathcal{W}.\, \mu(\Delta) =_n \nu(\Delta).$$

This is well defined and the metric a complete, bisected ultrametric by Proposition 7.2.

We need quite a few different function spaces and introduce some section-specific notation to help out. An arrow between metric spaces denotes the set of non-expansive maps as, e.g., in $\mathcal{T} \to \mathcal{T}$. If the metric spaces are ordered and the arrow has the monotonicity subscript then we restrict attention to functions that are both non-expansive and monotone; an example is the definition of the space $\mathcal{T}$ of types above. A superscript 1 on the arrow, on the other hand, indicate that we only require the maps to be one-expansive, i.e., Lipschitz continuous with Lipschitz constant 2. This is a weaker requirement than non-expansive; in the context of bisected metric spaces it means that elements that are $(n+1)$-equal are mapped to elements that are n-equal, for all $n \in \omega$. An example is $\mathcal{W} \to^1 \mathit{BohrRel}(S)$ which we shall meet soon. (In general, a one-expansive function from X to Y is the same as a non-expansive function from X to $\frac{1}{2}Y$.)

There is some room for variation here. If we modified the functor I that builds the islands of worlds by replacing $\frac{1}{2}[- \to_{mon} \mathit{BohrRel}(S)]$ with $\frac{1}{2}(-) \to_{mon} \mathit{BohrRel}(S)$, i.e., by requiring *one-contractive* heap laws, then the operations *states*, *cont* and *comp* defined below would be non-expansive. But then the types would be one-contractive too, *and* we would rely on that to prove the allocation case of the fundamental theorem of logical relations. Similar considerations apply to the slight change of the projection functions compared to BST, see also the discussion at the end of Section 4; none of the variations appear superior to the other, however.

The full definition of the logical relation is shown in Figures 7 and 8. In the rest of this section we show that the logical relation is indeed well-defined. This essentially amounts to checking that all relations involved in the definition are Bohr relations, and that all functions involved in the definition

are non-expansive or one-expansive and possibly monotone. In particular, the clause for recursive types is then well-defined by Banach's fixed-point Theorem.

In many (but not all) of the cases where we prove non-expansiveness it is actually possible to prove the stronger property of contractiveness. But this would clutter the picture, and so we skip it as we do not need this in the overall development. Remember also that the sets of values and states, V and S, are uniform predomains whereas the sets of computations and continuations, TV and K, are uniform domains, see Propositions 4.2 and 4.3. In particular, we have Bohr relations with metric on the former two according to Definition 6.14 and Theorem 6.16 whereas the Bohr relations with metric on the latter two follow Definition 6.8 and Theorem 6.9.

We focus on the cases involving states and references. The remaining cases are essentially as in Birkedal et al. [16], where more details can be found.

8.1. Relations on states, continuations and computations

Proposition 8.2. *The operator states defined in Figure 8 satisfies* $states \in \mathcal{W} \to^1 BohrRel(S)$.

Proof. We must show that for $\Delta \in \mathcal{W}$ we have that $states(\Delta) \in BohrRel(S)$, and we must also show that $states : \mathcal{W} \to BohrRel(S)$ is one-expansive. The first property is a consequence of the definition of order on the set of states S, the finiteness of $\mathrm{dom}(\Delta.\mathcal{I})$ and the fact that for any $n \in \mathrm{dom}(\Delta.\mathcal{I})$ we have that $\Delta.\mathcal{I}(n).HL(\Delta.\mathcal{I}(n).CP)$ maps into $BohrRel(S)$. As for one-expansiveness, assume that $\Delta_1 =_{n+1} \Delta_2$. We must show that $states(\Delta_1) =_n states(\Delta_2)$. By the construction of worlds, we have

$$\Delta_1.\varsigma_1 = \Delta_2.\varsigma_1, \quad \Delta_1.\varsigma_2 = \Delta_2.\varsigma_2, \quad \text{and } \mathrm{dom}(\Delta_1.\mathcal{I}) = \mathrm{dom}(\Delta_2.\mathcal{I})\,.$$

Also we get for all $m \in \mathrm{dom}(\Delta_1.\mathcal{I}) = \mathrm{dom}(\Delta_2.\mathcal{I})$ that $\Delta_1.\mathcal{I}(m).CP = \Delta_2.\mathcal{I}(m).CP$ and hence that

$$\Delta_1.\mathcal{I}(m).HL(\Delta_1.\mathcal{I}(m).CP) =_{n+1} \Delta_2.\mathcal{I}(m).HL(\Delta_2.\mathcal{I}(m).CP),$$

in the space $\frac{1}{2}(\hat{\mathcal{W}} \to BohrRel(S))$. But this means that we only have n-equality in the space $\hat{\mathcal{W}} \to BohrRel(S)$, and as $Sq(\Delta_1) =_{n+1} Sq(\Delta_2)$ holds too, we get

$$\Delta_1.\mathcal{I}(m).HL(\Delta_1.\mathcal{I}(m).CP)(Sq(\Delta_1)) =_n \Delta_2.\mathcal{I}(m).HL(\Delta_2.\mathcal{I}(m).CP)(Sq(\Delta_2))\,.$$

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

$$
\begin{aligned}
[\![\alpha]\!]_\Xi\varphi &= \varphi(\alpha) \\
[\![\texttt{int}]\!]_\Xi\varphi &= \lambda\Delta.\ \{(in_{\mathbb{Z}}n, in_{\mathbb{Z}}n) \mid n \in \mathbb{Z}\} \\
[\![1]\!]_\Xi\varphi &= \lambda\Delta.\ \{(in_1{*}, in_1{*})\} \\
[\![\tau_1 \times \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi \times [\![\tau_2]\!]_\Xi\varphi \\
[\![0]\!]_\Xi\varphi &= \lambda\Delta.\ \emptyset \\
[\![\tau_1 + \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi + [\![\tau_2]\!]_\Xi\varphi \\
[\![\texttt{ref}\ \tau]\!]_\Xi\varphi &= \mathit{ref}([\![\tau]\!]_\Xi\varphi) \\
[\![\forall\alpha.\tau]\!]_\Xi\varphi &= \lambda\Delta.\ \{(in_\forall c_1, in_\forall c_2) \mid \forall\nu \in \mathcal{T}.\ \forall\Delta' \sqsupseteq \Delta. \\
&\qquad\qquad (c_1, c_2) \in \mathit{comp}([\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu])(\Delta')\} \\
[\![\mu\alpha.\tau]\!]_\Xi\varphi &= \mathit{fix}\,(\lambda\nu.\ \lambda\Delta.\ \{(in_\mu v_1, in_\mu v_2) \mid (v_1, v_2) \in [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu](\Delta)\}) \\
[\![\tau_1 \to \tau_2]\!]_\Xi\varphi &= [\![\tau_1]\!]_\Xi\varphi \to [\![\tau_2]\!]_\Xi\varphi
\end{aligned}
$$

The following operators and elements are used above:

$$
\begin{aligned}
\times &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & \mathit{comp} &: \mathcal{T} \to (\mathcal{W} \to^1 \mathit{BohrRel}(TV)) \\
+ &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & \mathit{cont} &: \mathcal{T} \to (\mathcal{W} \to^1_{mon} \mathit{BohrRel}(K)) \\
\mathit{ref} &: \mathcal{T} \to \mathcal{T} & \mathit{states} &: \mathcal{W} \to^1 \mathit{BohrRel}(S) \\
\to &: \mathcal{T} \times \mathcal{T} \to \mathcal{T} & R_{Ans} &\in \mathit{BohrRel}(Ans)
\end{aligned}
$$

$$
\begin{aligned}
(\nu_1 \times \nu_2)(\Delta) &= \{\, (in_\times(v_1, v_2), in_\times(v_1', v_2')) \mid (v_1, v_1') \in \nu_1(\Delta) \wedge (v_2, v_2') \in \nu_2(\Delta) \,\} \\
(\nu_1 + \nu_2)(\Delta) &= \{\, (in_+(\iota_1\, v_1), in_+(\iota_1\, v_1')) \mid (v_1, v_1') \in \nu_1(\Delta) \,\} \cup \\
&\quad \{\, (in_+(\iota_2\, v_2), in_+(\iota_2\, v_2')) \mid (v_2, v_2') \in \nu_2(\Delta) \,\} \\
(\nu_1 \to \nu_2)(\Delta) &= \{\, (in_\to f, in_\to f') \mid \forall\Delta' \sqsupseteq \Delta.\, \forall(v, v') \in \nu(\Delta')\,.(f\, v, f'\, v') \in \mathit{comp}(\nu_2)(\Delta') \,\} \\
\mathit{comp}(\nu)(\Delta) &= \{\, (c, c') \mid \forall(k, k') \in \mathit{cont}(\nu)(\Delta). \\
&\qquad\qquad \forall(s, s') \in \mathit{states}(\Delta).\, (c\, k\, s, c'\, k'\, s') \in R_{Ans} \,\} \\
\mathit{cont}(\nu)(\Delta) &= \{\, (k, k') \mid \forall\Delta' \sqsupseteq \Delta.\, \forall(v, v') \in \nu(\Delta'). \\
&\qquad\qquad \forall(s, s') \in \mathit{states}(\Delta').\, (k\, v\, s, k'\, v'\, s') \in R_{Ans} \,\} \\
R_{Ans} &= \{\, (\bot, a) \mid a \in Ans\} \cup \{\, (\lfloor\iota_1\, m\rfloor, \lfloor\iota_1\, m\rfloor) \mid m \in \mathbb{Z} \,\}
\end{aligned}
$$

(Continued in Figure 8.)

Figure 7: Logical relation.

Definition of $ref : \mathcal{T} \to \mathcal{T}$

$$
\begin{aligned}
ref(\nu)(\Delta) = \big\{ (\lambda_{l_1}, \lambda_{l_2}) \mid \forall \Delta' \sqsupseteq \Delta.\, l_1 \in \Delta'.\varsigma_1 \wedge l_2 \in \Delta'.\varsigma_2 \wedge \\
\forall (s_1, s_2) \in states(\Delta'). \\
(s_1(l_1), s_2(l_2)) \in \nu(\Delta') \wedge \\
\forall (v_1, v_2) \in \nu(\Delta'). \\
(s_1[l_1 \mapsto v_1], s_2[l_2 \mapsto v_2]) \in states(\Delta') \big\} \cup \\
\big\{ (\lambda^{n+1}_{l_1}, \lambda_{l_2}) \mid \forall \Delta' \sqsupseteq \Delta.\, l_1 \in \Delta'.\varsigma_1 \wedge l_2 \in \Delta'.\varsigma_2 \wedge \\
\forall (s_1, s_2) \in states(\Delta').\, \pi_n(s_1) = \lfloor s_1' \rfloor \implies \\
(s_1'(l_1), s_2(l_2)) \in \nu(\Delta') \wedge \\
\forall (v_1, v_2) \in \nu(\Delta').\, \pi_n(v_1) = \lfloor v_1' \rfloor \implies \\
(s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in states(\Delta') \big\}.
\end{aligned}
$$

Definition of $states : \mathcal{W} \to^1 BohrRel(S)$

$$states(\Delta) = \{(s_1, s_2) \mid \mathrm{dom}(s_1) = \Delta.\varsigma_1 \wedge \mathrm{dom}(s_2) = \Delta.\varsigma_2 \wedge (s_1, s_2) \in sep(\Delta)\}$$

where $sep(\Delta)$ is an auxiliary relation on S defined by

$$
\begin{aligned}
(s_1, s_2) \in sep(\Delta) \iff \exists \sigma_1, \sigma_2 : \mathrm{dom}(\Delta.\mathcal{I}) \to S. \\
s_1 = \biguplus_{n \in \mathrm{dom}(\Delta.\mathcal{I})} \sigma_1(n) \quad \wedge \quad s_2 = \biguplus_{n \in \mathrm{dom}(\Delta.\mathcal{I})} \sigma_2(n) \quad \wedge \\
\forall n \in \mathrm{dom}(\Delta.\mathcal{I}). \\
(\sigma_1(n), \sigma_2(n)) \in \Delta.\mathcal{I}(n).HL(\Delta.\mathcal{I}(n).CP)(Sq(\Delta))
\end{aligned}
$$

Figure 8: Logical relation (ctd.)

Now let $(s_1, s_2) \in \mathit{states}(\Delta_1)$ and assume that $\pi^S_n(s_1) = \lfloor s_1' \rfloor \neq \bot$. We must show that $(s_1', s_2) \in \mathit{states}(\Delta_2)$. But this follows easily from the above equation. □

It is worthwhile to note that it is the (necessary) use of the shrinking factor $\frac{1}{2}$ in the construction of worlds in Section 7 that prevents us from proving non-expansiveness. This will haunt us throughout this subsection.

Lemma 8.3. *For all $n \in \omega$ and all $\Delta_1, \Delta_2, \Delta_1' \in \mathcal{W}$ with $\Delta_1 =_n \Delta_2$ and $\Delta_1 \sqsubseteq \Delta_1'$ there is $\Delta_2' \in \mathcal{W}$ with $\Delta_1' =_n \Delta_2'$ and $\Delta_2 \sqsubseteq \Delta_2'$.*

Proposition 8.4. *The operator cont defined in Figure 7 satisfies $\mathit{cont} \in \mathcal{T} \to (\mathcal{W} \to^1_{mon} \mathit{BohrRel}(K))$.*

Proof. We must show that for all $\nu \in \mathcal{T}$ and all $\Delta \in \mathcal{W}$ we have that $\mathit{cont}(\nu)(\Delta) \in \mathit{BohrRel}(K)$. We must furthermore show that $\mathit{cont} : \mathcal{T} \to (\mathcal{W} \to \mathit{BohrRel}(S))$ is non-expansive in the first argument and one-expansive in the second argument, and that for all $\nu \in \mathcal{T}$ and all $\Delta_1, \Delta_2 \in \mathcal{W}$ we have that

$$\Delta_1 \sqsubseteq \Delta_2 \implies \mathit{cont}(\nu)(\Delta_1) \subseteq \mathit{cont}(\nu)(\Delta_2).$$

The first property is an immediate consequence of the fact that R_{Ans} is itself a Bohr relation on the domain $\mathbb{Z}_\bot$. The expansiveness properties follow from Proposition 8.2, Lemma 8.3, and the definition of π^K_n in Figure 3. Monotonicity is immediate from the quantification over future worlds. □

Proposition 8.5. *The operator comp defined in Figure 7 satisfies $\mathit{comp} \in \mathcal{T} \to (\mathcal{W} \to^1 \mathit{BohrRel}(TV))$.*

The proof proceeds just as the proof of Proposition 8.4, except that one does not need to check monotonicity. This definition is, by the way, the exact point where we benefit from a continuation passing style semantics. The obvious direct style definition would not have continuations but rather call for some future world in which the results of the computations should be suitably related; this, however, is inherently chain-incomplete, and we would have a hard time producing relations in $\mathit{BohrRel}(TV)$.

8.2. Some Type Constructors

Proposition 8.6. *The operator ref defined in Figure 8 satisfies $\mathit{ref} \in \mathcal{T} \to \mathcal{T}$.*

Proof. Note first that, in both clauses, we quantify over pairs of states $(s_1, s_2) \in \mathit{states}(\Delta')$; in particular we that $l_1 \in \Delta'.\varsigma_1 = \mathrm{dom}(s_1)$ and $l_2 \in \Delta'.\varsigma_2 = \mathrm{dom}(s_2)$ by the definition of $\mathit{states}(\Delta')$ and so we only read and write allocated locations.

We must now show that for all $\nu \in \mathcal{T}$ and all $\Delta \in \mathcal{W}$ we have that $\mathit{ref}(\nu)(\Delta) \in \mathit{BohrRel}(V)$. Furthermore, we must show that $\mathit{ref} : \mathcal{T} \to \mathcal{W} \to \mathit{BohrRel}(V)$ is non-expansive in both arguments, and that for all $\nu \in \mathcal{T}$ and all $\Delta_1, \Delta_2 \in \mathcal{W}$ we have that

$$\Delta_1 \sqsubseteq \Delta_2 \implies \mathit{ref}(\nu)(\Delta_1) \subseteq \mathit{ref}(\nu)(\Delta_2).$$

The first property is a consequence of the fact that $\mathit{states}(\Delta)$ and $\nu(\Delta)$ are themselves Bohr relations for all $\Delta \in \mathcal{W}$. Monotonicity is immediate from the quantification over future worlds. Let us, however, prove non-expansiveness in some detail.

Let $n \in \omega$, $\nu_1, \nu_2 \in \mathcal{T}$ and $\Delta_1, \Delta_2 \in \mathcal{W}$ be given and assume that $\nu_1 =_{n+1} \nu_2$ and $\Delta_1 =_{n+1} \Delta_2$. We aim to show that

$$\mathit{ref}(\nu_1)(\Delta_1) =_{n+1} \mathit{ref}(\nu_2)(\Delta_2)\,.$$

Take $(v_1, v_2) \in \mathit{ref}(\nu_1)(\Delta_1)$ and assume that $\pi_{n+1}(v_1) = \lfloor v_1' \rfloor$ holds, we must prove $(v_1', v_2) \in \mathit{ref}(\nu_2)(\Delta_2)$. There must be $l_2 \in \mathit{Loc}$ such that $v_2 = \lambda_{l_2}$ and there must be $l_1 \in \mathit{Loc}$ and $m \leq n$ such that $v_1' = \lambda_{l_1}^{m+1}$. Since $\mathit{ref}(\nu_1)(\Delta_1) \in \mathit{BohrRel}(V)$ we have $(\lambda_{l_1}^{m+1}, \lambda_{l_2}) \in \mathit{ref}(\nu_1)(\Delta_1)$. We set forth to prove that we have $(\lambda_{l_1}^{m+1}, \lambda_{l_2}) \in \mathit{ref}(\nu_2)(\Delta_2)$ too.

According to definition, we take $\Delta_2' \sqsupseteq \Delta_2$ and must show that $l_1 \in \Delta_2'.\varsigma_1$ and $l_2 \in \Delta_2'.\varsigma_2$. By Lemma 8.3 we pick $\Delta_1' \sqsupseteq \Delta_1$ with $\Delta_1' =_{n+1} \Delta_2'$ and get that $l_1 \in \Delta_1'.\varsigma_1 = \Delta_2'.\varsigma_1$ and $l_2 \in \Delta_1'.\varsigma_2 = \Delta_2'.\varsigma_2$. We now pick $(s_1, s_2) \in \mathit{states}(\Delta_2')$ and assume that $\pi_m(s_1) = \lfloor s_1' \rfloor$ and get that $(s_1', s_2) \in \mathit{states}(\Delta_1')$ since $\mathit{states}(\Delta_1') =_m \mathit{states}(\Delta_2')$. Since $\pi_m(s_1') = \lfloor s_1' \rfloor$ we furthermore get $(s_1'(l_1), s_2(l_2)) \in \nu_1(\Delta_1')$ and

$$\begin{aligned}\forall (v_1, v_2) \in \nu_1(\Delta_1').\ \pi_m(v_1) = \lfloor v_1' \rfloor \implies \\ (s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta_1').\end{aligned}$$

Now $\pi_m(s_1'(l_1)) = \lfloor s_1'(l_1) \rfloor$ and so $(s_1'(l_1), s_2(l_2)) \in \nu_2(\Delta_2')$ as $\nu_1(\Delta_1') =_{n+1} \nu_2(\Delta_2')$. Take now $(v_1, v_2) \in \nu_2(\Delta_2')$ and assume that $\pi_m(v_1) = \lfloor v_1' \rfloor$, we must show that $(s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta_2')$. But since $(v_1', v_2) \in \nu_1(\Delta_1')$ and $\pi_m(v_1') = \lfloor v_1' \rfloor$ we get $(s_1'[l_1 \mapsto v_1'], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta_1')$ which in combination with the fact that $\pi_m(s_1'[l_1 \mapsto v_1']) = \lfloor s_1'[l_1 \mapsto v_1'] \rfloor$ gives us the desired. □

This interpretation of reference types differs markedly from ADR. The interpretation above is extensional whereas the one in ADR is intensional: it requires that the world must have an island that looks exactly as if it had been added according to the proof of the case of allocation in the proof of the fundamental theorem of logical relations. The intensional definition in ADR means that we may fail to recognize values as having reference type even though they, for some reason, behave just as references. The extensional definition above does, on the other hand, only support lookup and assignment. It would not suffice to model a language with equality testing on references such as the language in ADR. We conjecture that some notion of bijective bookkeeping could be added to remedy this, but we have not pursued the matter.

Proposition 8.7. *The operator $\to$ defined in Figure 7 belongs to $\mathcal{T}\times\mathcal{T} \to \mathcal{T}$.*

We omit a detailed proof but note that the one-expansiveness (in the second argument) of the operator *comp* is cancelled out by the index-shift in projections, see Equation 19 in Figure 3. A similar story can be told about the interpretation of universal types and reference types; in the latter case we do not, however, rely on the projections but rather on the index-shift from $\lambda_{l_1}^{n+1}$ to π_n in the second clause of the definition of *ref*. In some sense, this is as far as the one-expansiveness caused by the shrinking factor gets, confer the comment following the proof of Proposition 8.2.

Proposition 8.8. *The operators $\times$ and $+$ defined in Figure 7 belong to $\mathcal{T} \times \mathcal{T} \to \mathcal{T}$.*

8.3. Interpretation of Types

Theorem 8.9. *For all $\Xi \vdash \tau$ we have that $[\![\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ defined by induction on τ according to Figure 7 is well-defined and non-expansive. Here $\mathcal{T}^\Xi$ is equipped with the product metric.*

Proof. This is immediate from Propositions 8.6, 8.7, and 8.8 for all except universal and recursive types. And verifying the claim for $\Xi \vdash \forall\alpha.\tau$ under the assumption that it holds for $\Xi, \alpha \vdash \tau$ is not hard.

Consider now the case of $\Xi \vdash \mu\alpha.\tau$. We assume that $[\![\tau]\!]_{\Xi,\alpha} : \mathcal{T}^{\Xi,\alpha} \to \mathcal{T}$ is well defined and non-expansive. For $\varphi \in \mathcal{T}^\Xi$ we define

$$\Phi_\varphi = \lambda\nu \in \mathcal{T}.\ \lambda\Delta \in \mathcal{W}.\ \{(in_\mu v_1, in_\mu v_2) \mid (v_1, v_2) \in [\![\tau]\!]_{\Xi,\alpha}\varphi[\alpha \mapsto \nu](\Delta)\}$$

and it is not hard to see that this constitutes a contractive map $\Phi_\varphi : \mathcal{T} \to \mathcal{T}$. This means that $\mathit{fix}(\Phi_\varphi)$ is well defined by Banach's fixed point theorem. Furthermore we have that for any two $\varphi_1, \varphi_2 \in \mathcal{T}^\Xi$ with $\varphi_1 =_n \varphi_2$ for some $n \in \omega$ we get $\Phi_{\varphi_1} =_{n+1} \Phi_{\varphi_2}$. It then follows from Proposition 5.1 that $\mathit{fix}(\Phi_{\varphi_1}) =_{n+1} \mathit{fix}(\Phi_{\varphi_2})$. In summa, $[\![\mu\alpha.\tau]\!]_\Xi : \mathcal{T}^\Xi \to \mathcal{T}$ is well-defined and contractive. □

9. Fundamental Theorem of Logical Relations

This definition with ensuing lemma will do much of the bookkeeping for us in the proofs to come:

Definition 9.1. *For $\mu, \nu \in \mathcal{T}$ and $\Delta \in \mathcal{W}$ we define a relation on $V \to TV$ by*

$$\mu \to_\Delta \nu = \{(f_1, f_2) \mid \forall \Delta' \sqsupseteq \Delta.\ \forall (v_1, v_2) \in \mu(\Delta').\ (f_1 v_2, f_2 v_2) \in \mathit{comp}(\nu)(\Delta').$$

Lemma 9.2. *For $\nu \in \mathcal{T}$, $\Delta \in \mathcal{W}$ and $(v_1, v_2) \in \nu(\Delta)$ we have*

$$(\eta\, v_1, \eta\, v_2) \in \mathit{comp}(\nu)(\Delta),$$

and for $\mu, \nu \in \mathcal{T}$, $\Delta \in \mathcal{W}$, $(c_1, c_2) \in \mathit{comp}(\mu)(\Delta)$, $(f_1, f_2) \in \mu \to_\Delta \nu$ we have

$$(c_1 \star f_1, c_2 \star f_2) \in \mathit{comp}(\nu)(\Delta).$$

Proof. To prove the first, we take related pairs $(k_1, k_2) \in \mathit{cont}(\nu)(\Delta)$ and $(s_1, s_2) \in \mathit{states}(\Delta)$ and get that

$$((\eta\, v_1)\, k_1 s_2, (\eta\, v_2)\, k_2 s_2) = (k_1 v_1 s_1, k_2, v_2, s_2) \in R_{Ans}$$

by the definition of $\eta : V \to TV$ and $\mathit{cont}(\nu)(\Delta)$.

To prove the second, we similarly take related pairs $(k_1, k_2) \in \mathit{cont}(\nu)(\Delta)$ and $(s_1, s_2) \in \mathit{states}(\Delta)$ and must prove that

$$((c_1 \star f_1)\, k_1 s_1, (c_2 \star f_2)\, k_2 s_2) \in R_{Ans}.$$

By definition of $\star : TV \times (V \to TV) \to TV$ we get that

$$\big((c_1 \star f_1)\, k_1 s_1, (c_2 \star f_2)\, k_2 s_2\big) = \big(c_1(\lambda v_1.\, \lambda t_1.\, f_1 v_1 k_1 t_1) s_1, c_2(\lambda v_2.\, \lambda t_2.\, f_2 v_2 k_2 t_2) s_2\big)$$

and so it remains to prove that

$$(\lambda v_1.\, \lambda s_1'.\, f_1 v_1 k_1 s_1', \lambda v_2.\, \lambda s_2'.\, f_2 v_2 k_2 s_2') \in \mathit{cont}(\mu)(\Delta).$$

So we take $\Delta' \sqsupseteq \Delta$, $(v_1, v_2) \in \mu(\Delta')$, $(s_1', s_2') \in \mathit{states}(\Delta')$ and must prove that we have

$$(f_1 v_1 k_1 s_1', f_2 v_2 k_2 s_2') \in R_{Ans}.$$

But the definition of $\mu \to_\Delta \nu$ gives us that $(f_1 v_1, f_2 v_2) \in \mathit{comp}(\nu)(\Delta')$ and by monotonicity we have $(k_1, k_2) \in \mathit{cont}(\nu)(\Delta')$ and we are done. □

We are now ready to define what it means for two terms of the same type to be *semantically related*. First up is the definition of related environments:

Definition 9.3. *For every term environment $\Xi \vdash \Gamma$, every $\varphi \in \mathcal{T}^\Xi$ and every $\Delta \in \mathcal{W}$ we let $[\![\Gamma]\!]_\Xi \varphi(\Delta)$ be the binary relation on $V^{\mathrm{dom}(\Gamma)}$ defined by*

$$[\![\Gamma]\!]_\Xi \varphi(\Delta) = \{(\rho_1, \rho_2) \mid \forall x \in \mathrm{dom}(\Gamma).\, (\rho_1(x), \rho_2(x)) \in [\![\Gamma(x)]\!]_\Xi \varphi(\Delta)\}.$$

Definition 9.4. *Assume $\Xi \vdash \Gamma$ and terms t_1 and t_2 with free variables in* $\mathrm{dom}(\Gamma)$. *We say that t_1 and t_2 are* semantically related, *written $\Xi \mid \Gamma \models t_1 \sim t_2 : \tau$, if for all $\varphi \in \mathcal{T}^\Xi$, all $\Delta \in \mathcal{W}$, and all $(\rho_1, \rho_2) \in [\![\Gamma]\!]_\Xi \varphi(\Delta)$,*

$$\big([\![t_1]\!]_{\mathrm{dom}(\Gamma)} \rho_1, [\![t_2]\!]_{\mathrm{dom}(\Gamma)} \rho_2\big) \in \mathit{comp}([\![\tau]\!]_\Xi \varphi)(\Delta).$$

Theorem 9.5 (Fundamental Theorem)**.** *Semantic relatedness is preserved by all typing rules. In particular, we have that any typed term is semantically related to itself, i.e, for any $\Xi \mid \Gamma \vdash t : \tau$ we have $\Xi \mid \Gamma \models t \sim t : \tau$.*

Also, and in combination with adequacy, this means that the logical relation approximates contextual approximation; the exact definition of the latter and the details of the argument is standard and we omit them here.

Proof. We provide proofs for only a few interesting cases, and refer to BST [12] with associated technical report [16] for the remaining. The definitions that concern state and references have changed sufficiently that going through the cases of lookup, assignment and allocation in detail is reasonable.

The Case of Lookup

Consider the case of lookup. Assume that $\Xi \mid \Gamma \vdash t_1 \sim t_2 : \mathtt{ref}\ \tau$ holds, we must show that $\Xi \mid \Gamma \vdash !\, t_1 \sim !\, t_2 : \tau$ holds too. We unroll the definition; take $\varphi \in \mathcal{T}^\Xi$, $\Delta \in \mathcal{W}$ and $(\rho_1, \rho_2) \in [\![\Gamma]\!]_\Xi \varphi(\Delta)$ and aim to show that

$$([\![!\, t_1]\!]_X \rho_1, [\![!\, t_2]\!]_X \rho_2) \in \mathit{comp}([\![\tau]\!]_\Xi \varphi)(\Delta),$$

where we for brevity write X for $\mathrm{dom}(\Gamma)$. By definition we have that

$$(\llbracket !\, t_1 \rrbracket_X \rho_1, \llbracket !\, t_2 \rrbracket_X \rho_2) = (\llbracket t_1 \rrbracket_X \rho_1 \star \lambda v_1.\, lookup\, v_1, \llbracket t_2 \rrbracket_X \rho_2 \star \lambda v_2.\, lookup\, v_2),$$

and by Lemma 9.2 we are down to proving

$$(\lambda v_1.\, lookup\, v_1, \lambda v_2.\, lookup\, v_2) \in \llbracket \mathtt{ref}\ \tau \rrbracket_\Xi \varphi \to_\Delta \llbracket \tau \rrbracket_\Xi \varphi.$$

Again we unroll, take $\Delta' \sqsupseteq \Delta$ and related pairs $(v_1, v_2) \in \llbracket \mathtt{ref}\ \tau \rrbracket_\Xi \varphi(\Delta')$, $(k_1, k_2) \in cont(\llbracket \tau \rrbracket_\Xi \varphi)(\Delta')$ and $(s_1, s_2) \in states(\Delta')$; our proof obligation now is

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) \in R_{Ans}.$$

We branch on the possible values of v_1 and v_2 according to the definition of $\llbracket \mathtt{ref}\ \tau \rrbracket_\Xi \varphi(\Delta')$. The first possibility is that there are l_1 and l_2 in Loc such that $v_1 = \lambda_{l_1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$ and $(s_1(l_1), s_2(l_2)) \in \llbracket \tau \rrbracket_\Xi \varphi(\Delta')$. But in that case we have

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) = (k_1\, s_1(l_1)\, s_1, k_2\, s_2(l_2)\, s_2) \in R_{Ans}$$

and are done.

In the second possible branch there are $n \in \omega$ and l_1 and l_2 in Loc such that $v_1 = \lambda_{l_1}^{n+1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$. Furthermore, we know that if $\pi_n(s_1) = \lfloor s_1' \rfloor$ then $(s_1'(l_1), s_2(l_2)) \in \llbracket \tau \rrbracket_\Xi \varphi(\Delta')$. If now $\pi_n(s_1) = \bot$ we get

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) = (\bot, lookup\, v_2\, k_2\, s_2) \in R_{Ans},$$

by the definition of $lookup : V \to TV$. On the other hand, $\pi_n(s_1) = \lfloor s_1' \rfloor$ gives us that

$$(lookup\, v_1\, k_1\, s_1, lookup\, v_2\, k_2\, s_2) = (k_1\, s_1'(l_1)\, s_1, k_2\, s_2(l_2)\, s_2) \in R_{Ans}.$$

The Case of Assignment

We now turn to assignment. Assume that we have $\Xi \mid \Gamma \vdash t_1 \sim t_2 : \mathtt{ref}\ \tau$ and $\Xi \mid \Gamma \vdash u_1 \sim u_2 : \tau$, we must prove that $\Xi \mid \Gamma \vdash t_1 := u_1 \sim t_2 := u_2 : 1$. Take $\varphi \in \mathcal{T}^\Xi$, $\Delta \in \mathcal{W}$ and $(\rho_1, \rho_2) \in \llbracket \Gamma \rrbracket_\Xi \varphi(\Delta)$ and aim to show that

$$(\llbracket t_1 := u_1 \rrbracket_X \rho_1, \llbracket t_2 := u_2 \rrbracket_X \rho_2) \in comp(\llbracket 1 \rrbracket_\Xi \varphi)(\Delta),$$

where we for brevity write X for $\mathrm{dom}(\Gamma)$. As was the case for lookup, we proceed by recalling the interpretation of the terms; we have that

$$[\![t_1 := u_1]\!]_X\rho_1 = [\![t_1]\!]_X\rho_1 \star \lambda v_1.\, [\![u_1]\!]_X\rho_1 \star \lambda w_1.\, assign\, v_1\, w_1$$

and similarly that

$$[\![t_2 := u_2]\!]_X\rho_2 = [\![t_2]\!]_X\rho_2 \star \lambda v_2.\, [\![u_2]\!]_X\rho_2 \star \lambda w_2.\, assign\, v_2\, w_2.$$

By an application of Lemma 9.2 in conjunction with the first assumption of this case we need to prove only that

$$(\lambda v_1.\, [\![u_1]\!]_X\rho_1 \star \lambda w_1.\, assign\, v_1\, w_1, \lambda v_2.\, [\![u_2]\!]_X\rho_2 \star \lambda w_2.\, assign\, v_2\, w_2)$$

is a member of $[\![\mathtt{ref}\ \tau]\!]_\Xi\varphi \to_\Delta [\![1]\!]_\Xi\varphi$. Take $\Delta' \sqsupseteq \Delta$, $(v_1, v_2) \in [\![\mathtt{ref}\ \tau]\!]_\Xi\varphi(\Delta')$ and apply Lemma 9.2 with the second assumption of this case to arrive at the proof obligation

$$(\lambda w_1.\, assign\, v_1\, w_1, \lambda w_2.\, assign\, v_2\, w_2) \in [\![\tau]\!]_\Xi\varphi \to_{\Delta'} [\![1]\!]_\Xi\varphi.$$

We pick $\Delta'' \sqsupseteq \Delta'$ and $(w_1, w_2) \in [\![\tau]\!]_\Xi\varphi(\Delta'')$, $(k_1, k_2) \in cont([\![1]\!]_\Xi\varphi)(\Delta'')$ and $(s_1, s_2) \in states(\Delta'')$ and arrive – finally – at the core of this case, as we plan to show

$$(assign\, v_1\, w_1\, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) \in R_{Ans}.$$

As above, we branch on the possible values of v_1 and v_2 according to the definition of $[\![\mathtt{ref}\ \tau]\!]_\Xi\varphi(\Delta')$. The first possibility is that there are l_1 and l_2 in Loc such that $v_1 = \lambda_{l_1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$ and $(s_1[l_1 \mapsto w_1], s_2[l_2 \mapsto w_2]) \in states(\Delta'')$. This means that

$$(assign\, v_1\, w_1, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) = \\ (k_1\, (in_1 *)\, s_1[l_1 \mapsto w_1], k_2\, (in_1 *)\, s_2[l_2 \mapsto w_2]).$$

and this branch is done.

The second possibility is that there are $n \in \omega$ and l_1 and l_2 in Loc such that $v_1 = \lambda_{l_1}^{n+1}$ and $v_2 = \lambda_{l_2}$ and such that we know $l_1 \in \mathrm{dom}(s_1)$, $l_2 \in \mathrm{dom}(s_2)$. Furthermore, if $\pi_n(s_1) = \lfloor s_1' \rfloor$ we have that $\pi_n(w_1) = \lfloor w_1' \rfloor$ means that we have $(s_1'[l_1 \mapsto w_1'], s_2[l_2 \mapsto w_2]) \in states(\Delta'')$. If either $\pi_n(s_1) = \bot$ or $\pi_n(w_1) = \bot$ we get that

$$(assign\, v_1\, w_1, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) = (\bot, assign\, v_2\, w_2\, k_2\, s_2) \in R_{Ans}$$

by the definition of $assign : V \to V \to TV$. Otherwise we get $\pi_n(s_1) = \lfloor s_1' \rfloor$ and $\pi_n(w_1) = \lfloor w_1' \rfloor$ for some $s_1' \in S$ and $w_1' \in V$. And this buys us

$$(assign\, v_1\, w_1, k_1\, s_1, assign\, v_2\, w_2\, k_2\, s_2) = \\ (k_1\, (in_1 *)\, s_1'[l_1 \mapsto w_1'], k_2\, (in_1 *)\, s_2[l_2 \mapsto w_2])$$

which is an element of R_{Ans}.

The Case of Allocation

We will now go into the allocation of new references. Assume that we have $\Xi \mid \Gamma \vdash t_1 \sim t_2 : \tau$, we must prove $\Xi \mid \Gamma \vdash \texttt{ref}\ t_1 \sim \texttt{ref}\ t_2 : \texttt{ref}\ \tau$. We make the canonical choices of $\varphi \in \mathcal{T}^{\Xi}$, $\Delta \in \mathcal{W}$ and $(\rho_1, \rho_2) \in [\![\Gamma]\!]_{\Xi}\varphi(\Delta)$ and proceed to show

$$([\![\texttt{ref}\ t_1]\!]_X\rho_1, [\![\texttt{ref}\ t_2]\!]_X\rho_2) \in comp([\![\texttt{ref}\ \tau]\!]_{\Xi}\varphi)(\Delta),$$

where we, as usual, write X for $\mathrm{dom}(\Gamma)$. Now, we have by definition that

$$([\![\texttt{ref}\ t_1]\!]_X\rho_1, [\![\texttt{ref}\ t_2]\!]_X\rho_2) = ([\![t_1]\!]_X\rho_1 \star \lambda v_1.\, alloc\, v_1, [\![t_2]\!]_X\rho_2 \star \lambda v_2.\, alloc\, v_2)$$

and so we apply the assumption of the case together with Lemma 9.2, pick $\Delta' \sqsupseteq \Delta$, $(v_1, v_2) \in [\![\tau]\!]_{\Xi}\varphi(\Delta')$, $(k_1, k_2) \in cont([\![\texttt{ref}\ \tau]\!]_{\Xi}\varphi)(\Delta')$ and $(s_1, s_2) \in states(\Delta')$ and are now down to proving

$$(alloc\, v_1\, k_1\, s_1, alloc\, v_2\, k_2\, s_2) \in R_{Ans}.$$

As a first step, we rewrite the above pair according to the definition of $alloc : V \to TV$ to get

$$(alloc\, v_1\, k_1\, s_1, alloc\, v_2\, k_2\, s_2) = (k_1\, \lambda_{l_1}\, s_1[l_1 \mapsto v_1], k_2\, \lambda_{l_2}\, s_2[l_2 \mapsto v_2])$$

where $l_1 \in Loc$ is the least with $l_1 \notin \mathrm{dom}(s_1)$ and $l_2 \in Loc$ is the least with $l_2 \notin \mathrm{dom}(s_2)$. As we have allocated new locations we should extend the world correspondingly. We define for each $\hat{\Delta} \in \hat{\mathcal{W}}$ a relation $\Phi(\hat{\Delta})$ on S by

$$\{(s_1, s_2) \mid l_1 \in \mathrm{dom}(s_1) \wedge l_2 \in \mathrm{dom}(s_2) \wedge (s_1(l_1), s_2(l_2)) \in [\![\tau]\!]_{\Xi}\varphi(Sq^{-1}(\hat{\Delta}))\}$$

and remark that $\Phi : \hat{\mathcal{W}} \to BohrRel(S)$ is well-defined, monotone and non-expansive. But then $\Theta = \{\emptyset, \{\emptyset\}, \lambda\_.\, \Phi\}$ easily is an island, i.e., a member of $I(\hat{\mathcal{W}})$. We define $\Delta'' \in \mathcal{W}$ by

$$\Delta''.\varsigma_1 = \Delta'.\varsigma_1 \cup \{l_1\}, \Delta''.\varsigma_2 = \Delta'.\varsigma_2 \cup \{l_2\}, \Delta''.\mathcal{I} = \Delta'.\mathcal{I}[n \mapsto \Theta]$$

where $n \in \omega$ is the least with $n \notin \mathrm{dom}(\Delta'.\mathcal{I})$. It is immediate by definition that $\Delta'' \sqsupseteq \Delta'$ and so it remains to prove that $(\lambda_{l_1}, \lambda_{l_2}) \in [\![\texttt{ref}\ \tau]\!]_\Xi\varphi(\Delta'')$ and that $(s_1[l_1 \mapsto v_1], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta'')$.

Addressing the first issue, take $\Delta''' \sqsupseteq \Delta''$, we have $l_1 \in \Delta''.\varsigma_1 \subseteq \Delta'''.\varsigma_1$ and $l_2 \in \Delta''.\varsigma_2 \subseteq \Delta'''.\varsigma_2$ by definition of world extension. Assume now that we have $(q_1, q_2) \in \mathit{states}(\Delta''')$. This would imply the existence of subheaps $q_1' \subseteq q_1$ and $q_2' \subseteq q_2$ with $(q_1', q_2') \in \Phi(Sq(\Delta'''))$, also by the definition of world extension. This means that $l_1 \in \mathrm{dom}(q_1')$, $l_2 \in \mathrm{dom}(q_2')$ and that $(q_1(l_1), q_2(l_2)) = (q_1'(l_1), q_2'(l_2)) \in [\![\tau]\!]_\Xi\varphi(\Delta''')$. And if we pick $(w_1, w_2) \in [\![\tau]\!]_\Xi\varphi(\Delta''')$ then we have $(q_1'[l_1 \mapsto w_1], q_2'[l_2 \mapsto w_2]) \in \Phi(Sq(\Delta'''))$ and hence $(q_1[l_1 \mapsto w_1], q_2[l_2 \mapsto w_2]) \in \mathit{states}(\Delta''')$. In conclusion, $(\lambda_{l_1}, \lambda_{l_2}) \in [\![\texttt{ref}\ \tau]\!]_\Xi\varphi(\Delta'')$. Showing that $(s_1[l_1 \mapsto v_1], s_2[l_2 \mapsto v_2]) \in \mathit{states}(\Delta'')$ holds is not hard as we recall that $(s_1, s_2) \in \mathit{states}(\Delta')$ and that for any $m \in \mathrm{dom}(\Delta'.\mathcal{I})$ we have $\Delta''.\mathcal{I}(m) = \Delta'.\mathcal{I}(m)$, but do notice that this where we crucially rely on monotonicity of types and of the heap law of an island. □

10. Examples

10.1. Syntactic Sugar: Existential Types

Our language has universal types with associated term constructs but does not, a priori, come with existential types. But we can apply the standard encoding of existential types as universal types [26, Section 24.3] as follows:

Definition 10.1. *We write $\exists\alpha.\,\tau$ for the type $\forall\beta.\,(\forall\alpha.\,\tau \to \beta) \to \beta$ where β is not in τ. And we write* $\texttt{pack}\ \sigma, t$ *for the term $\Lambda\beta.\,\lambda f.\,f\,[\sigma]\,t$.*

It is easy to show that

$$\frac{\Xi \vdash \sigma \qquad \Xi \mid \Gamma \vdash t : \tau[\sigma/\alpha]}{\Xi \mid \Gamma \vdash \texttt{pack}\ \sigma, t : \exists\alpha.\,\tau}$$

is a derived typing rule. We do not need to unpack existential packages in the examples to come, but this could be encoded too. Instead we provide the following semantic lemma that is both useful and reassuring:

Lemma 10.2. *Define $in_\exists : TV \to V$ by $in_\exists(c) = in_\forall(\eta(in_\to(\psi_c)))$ where*

$$\psi_c = \lambda u.\,\eta(u) \star \lambda v.\,\begin{cases} d & v = in_\forall(d) \\ \mathit{error} & \text{otherwise} \end{cases} \star \lambda w.\,c \star \lambda x.\,\begin{cases} f\,x & w = in_\to(f) \\ \mathit{error} & \text{otherwise} \end{cases}.$$

We then have for $\Xi, \alpha \vdash \tau$, $\varphi \in \mathcal{T}^{\Xi}$, $\Delta \in \mathcal{W}$ *and* $c_1, c_2 \in TV$ *that*

$$\big(\exists \nu \in \mathcal{T}.\, \forall \Delta' \sqsupseteq \Delta.\, (c_1, c_2) \in [\![\tau]\!]_{\Xi,\alpha} \varphi[\alpha \mapsto \nu](\Delta')\big) \implies \big((in_\exists(c_1), in_\exists(c')) \in [\![\exists \alpha.\, \tau]\!]_\Xi \varphi(\Delta)\big).$$

Notice here the similarity with the interpretation of types, only the quantification is different. And that we cannot reason both ways; we do not know whether the reverse implication holds. The map $in_\exists : TV \to V$ was constructed by unrolling the interpretation of $\mathtt{pack}\ \sigma, t$ to the point where no syntax was left; indeed, we have $[\![\mathtt{pack}\ \sigma, t]\!]_X \rho = \eta(in_\exists([\![t]\!]_X \rho))$ whenever all term variables of t are in X.

10.2. More Sugar: Let Bindings and Sequencing

Definition 10.3. *For terms* s *and* t *and a variable* x *we write* $\mathtt{let}\ x = s\ \mathtt{in}\ t$ *for the term* $(\lambda x.\, t)\, s$. *For terms* s *and* t *we write* $s; t$ *for* $\mathtt{let}\ x = s\ \mathtt{in}\ t$ *where* x *is not in* t.

We have the obvious derived typing rules

$$\frac{\Xi \mid \Gamma \vdash s : \tau \qquad \Xi \mid \Gamma, x : \tau \vdash t : \sigma}{\Xi \mid \Gamma \vdash \mathtt{let}\ x = s\ \mathtt{in}\ t : \sigma} \qquad \frac{\Xi \mid \Gamma \vdash s : \tau \qquad \Xi \mid \Gamma \vdash t : \sigma}{\Xi \mid \Gamma \vdash s; t : \sigma}$$

and by using the convenient fact that we have $\eta(v) \star f = f(v)$ for any $v \in V$ and $f \in V \to TV$ we easily have the following lemma:

Lemma 10.4. *We have that* $[\![\mathtt{let}\ x = s\ \mathtt{in}\ t]\!]_X \rho = [\![s]\!]_X \rho \star \lambda v.\, [\![t]\!]_{X,x} \rho[x \mapsto v]$ *and that* $[\![s; t]\!]_X \rho = [\![s]\!]_X \rho \star \lambda_{-}.\, [\![t]\!]_X \rho$.

10.3. Booleans

We need the type $\mathtt{bool}$ of booleans in the example to come. Abbreviate

$$\begin{aligned} \mathtt{bool} &= 1 + 1 \\ \mathtt{true} &= \mathsf{inl}\,() \\ \mathtt{false} &= \mathsf{inr}\,() \end{aligned}$$

We also introduce some convenient notation on the semantic side: Let $\mathbb{B} = \{0, 1\}$ be the discrete two-point predomain, and define $in_{\mathbb{B}} : \mathbb{B} \to V$ by $in_{\mathbb{B}}(1) = in_+(\iota_1(*))$ and $in_{\mathbb{B}}(0) = in_+(\iota_2(*))$. Then $[\![\mathtt{true}]\!]_X \rho = \eta(in_{\mathbb{B}}\, 1)$ and $[\![\mathtt{false}]\!]_X \rho = \eta(in_{\mathbb{B}}\, 0)$.

It is furthermore convenient to add an integer comparison operator $t_1 \leq t_2$ to the language. It has the following typing rule and semantics:

$$\frac{\Xi \mid \Gamma \vdash t_1 : \mathtt{int} \qquad \Xi \mid \Gamma \vdash t_2 : \mathtt{int}}{\Xi \mid \Gamma \vdash t_1 \leq t_2 : \mathtt{bool}}$$

$$[\![t_1 \leq t_2]\!]_X \rho = [\![t_1]\!]_X \rho \star \lambda v_1.\, [\![t_2]\!]_X \rho \star \lambda v_2. \begin{cases} \eta(in_{\mathbb{B}}\, 1) & v_1 = in_{\mathbb{Z}}\, n,\ v_2 = in_{\mathbb{Z}}\, m,\ n \leq m \\ \eta(in_{\mathbb{B}}\, 0) & v_1 = in_{\mathbb{Z}}\, n,\ v_2 = in_{\mathbb{Z}}\, m,\ n > m \\ \mathit{error} & \text{otherwise.} \end{cases}$$

(One can encode this operator using ifz and fix, but the encoding is fairly complicated.)

10.4. Name Generator

Consider the program t_1 given by

$$t_1 = \mathtt{let\ x} = \mathtt{ref\ 0\ in\ pack\ int}, (\lambda \mathtt{z.\, x} := \mathtt{!\,x} + \mathtt{1}; \mathtt{!\,x}, \lambda \mathtt{z.\, z} \leq \mathtt{!\,x}).$$

It is not hard to assign it the type $\exists\alpha.\, (1 \to \alpha) \times (\alpha \to \mathtt{bool})$. The idea is that of a name generator, each call to the first function returns a fresh name of type α by incrementing and then returning the value stored at location x. The second function is a sanity check, it asserts that a supplied value of type α is valid, i.e., does not exceed the largest name supplied so far. Put roughly, it can never return false because there is no way of producing stray values of α. And indeed, we shall prove e_1 contextually equivalent to the program t_2 given by

$$t_2 = \mathtt{let\ x} = \mathtt{ref\ 0\ in\ pack\ int}, (\lambda \mathtt{z.\, x} := \mathtt{!\,x} + \mathtt{1}; \mathtt{!\,x}, \lambda \mathtt{z.\, true}),$$

where we have replaced the second function with a dummy that always returns true. The approach is, of course, to prove the interpretation of e_1 semantically related to the interpretation of e_2 at type $\exists\alpha.\, (1 \to \alpha) \times (\alpha \to \mathtt{bool})$ and the other way round, we shall do only the first.

So, let us take on the task. We must show that $\models t_1 \sim t_2 : \exists\alpha.\, (1 \to \alpha) \times (\alpha \to \mathtt{bool})$ where we note that both the type and term contexts are empty. This means picking $\Delta \in \mathcal{W}$ arbitrary, taking $(k_1, k_2) \in \mathit{cont}(\exists\alpha.\, (1 \to \alpha) \times (\alpha \to \mathtt{bool}))(\Delta)$ and $(s_1, s_2) \in \mathit{states}(\Delta)$ and proving

$$([\![t_1]\!]\, k_1\, s_1, [\![t_2]\!]\, k_2\, s_2) \in R_{Ans}.$$

A few calculations gives us that the left component $[\![t_1]\!]\, k_1\, s_1$ equals

$$[\![\texttt{pack int}, (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{z} \leq !\,\texttt{x})]\!]_x[x \mapsto \lambda_{l_1}]\, k_1\, s_1[l_1 \mapsto 0]$$

where $l_1 \in \omega$ is the least such that $l_1 \notin \mathrm{dom}(s_1)$. Similarly we have that right component $[\![t_2]\!]\, k_2\, s_2$ equals

$$[\![\texttt{pack int}, (\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{true})]\!]_x[x \mapsto \lambda_{l_2}]\, k_2\, s_2[l_2 \mapsto 0]$$

where $l_2 \in \omega$ is the least such that $l_2 \notin \mathrm{dom}(s_2)$. Writing out a few more lines we arrive at

$$k_1\, in_{\exists}([\![(\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{z} \leq !\,\texttt{x})]\!]_x \rho_1)\, s_1[l_1 \mapsto 0],$$

and at

$$k_2\, in_{\exists}([\![(\lambda \texttt{z}.\, \texttt{x} := !\,\texttt{x} + \texttt{1}; !\,\texttt{x}, \lambda \texttt{z}.\, \texttt{true})]\!]_x \rho_2)\, s_2[l_2 \mapsto 0]$$

as our left and right hand side components, respectively. For brevity we write ρ_1 for $[x \mapsto \lambda_{l_1}]$ and ρ_2 for $[x \mapsto \lambda_{l_2}]$.

We are now, so to speak, at a point where allocation has been made by both programs and so we aim to extend the world to reflect this. First up, we define for each $n \in \omega$ a relation on S indexed by $\hat{\Delta} \in \hat{\mathcal{W}}$ as follows:

$$\Phi_n(\hat{\Delta}) = \{(s_1, s_2) \mid l_1 \in \mathrm{dom}(s_1) \wedge l_2 \in \mathrm{dom}(s_2) \wedge s_1(l_1) = s_2(l_2) = in_{\mathbb{Z}}\, n\}$$

It is easy to verify that $\Phi_n : \hat{\mathcal{W}} \to \mathit{BohrRel}(S)$ is well-defined and since it is constant it is monotone and non-expansive too. Let now $P_n = \{1, 2, \ldots, n\}$ for any $n \in \omega$, in particular we have $P_0 = \emptyset$. We then define

$$\Theta = \left(P_0, \{P_n \mid n \in \omega\}, \lambda X.\, \begin{cases} \Phi_n & X = P_n \\ \_ & \text{otherwise} \end{cases} \right)$$

and note that this is island, i.e., $\Theta \in I(\hat{\mathcal{W}})$. The population corresponds to the names generated so far; as the left and right name generators work in lock-step they always have the same set of generated names. Notice that it is initially empty because no names have been generated so far and that we restrict it to values from $\{P_0, P_1, \ldots\}$. The heap law just matches populations with the indexed relations on states; the definition requires us to define images of all of the subsets of V but we shall only ever need images

of $\{P_0, P_1, \ldots\}$ and hence leave the remaining unspecified. We now define $\Delta' \in \mathcal{W}$ by

$$\Delta'.\varsigma_1 = \Delta.\varsigma_1 \cup \{l_1\}, \Delta'.\varsigma_2 = \Delta.\varsigma_2 \cup \{l_2\}, \Delta'.\mathcal{I} = \Delta.\mathcal{I}[n \mapsto \Theta]$$

where $n \in \omega$ is the least with $n \notin \mathrm{dom}(\Delta.\mathcal{I})$. It is immediate that $\Delta' \sqsupseteq \Delta$.

Having extended the world with an island that keeps track of the counters of both name generators we now build the type of generated names. These are exactly the population of the new island, so we just read them off; define a relation on V for $\Delta^* \in \mathcal{W}$ by

$$\nu(\Delta^*) = \begin{cases} \{(in_{\mathbb{Z}}\, v, in_{\mathbb{Z}}\, v) \mid v \in \Delta^*.\mathcal{I}(n).CP\} & n \in \mathrm{dom}(\Delta^*.\mathcal{I}) \wedge \\ & \Delta^*.\mathcal{I}(n).PL = \Theta.PL \\ \emptyset & \text{otherwise.} \end{cases}$$

We shall only apply this type to the world Δ' and possible extensions of this and so the second clause is really unreachable. But we cannot do without it, as the definition of $\mathcal{T}$ requires us to give values to all worlds. It is not hard to prove $\nu : \mathcal{W} \to BohrRel(V)$ well defined, non-expansive and monotone; we rely on the fact that island populations cannot shrink under world extension for the latter.

We now return to the issue at hand. As continuations are required to behave in future worlds and as $(s_1[l_1 \mapsto 0], s_2[l_2 \mapsto 0])$ easily is a member of $states(\Delta')$, it shall suffice to show that the pair

$$\big(in_\exists([\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{z} \leq !\,\mathtt{x})]\!]_x \rho_1), \\ in_\exists([\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{true})]\!]_x \rho_2)\big)$$

is a member of $[\![\exists \alpha.\, (1 \to \alpha) \times (\alpha \to \mathtt{bool})]\!](\Delta')$. Now take $\Delta'' \sqsupseteq \Delta'$ arbitrary, by Lemma 10.2 it shall suffice to show that

$$\big([\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{z} \leq !\,\mathtt{x})]\!]_x \rho_1, [\![(\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}, \lambda \mathtt{z}.\, \mathtt{true})]\!]_x \rho_2\big)$$

is a member of $comp([\![(1 \to \alpha) \times (\alpha \to \mathtt{bool})]\!]_\alpha[\alpha \mapsto \nu])(\Delta'')$. This again comes down to the following two obligations:

1. Prove that $([\![\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}]\!]_x \rho_1, [\![\lambda \mathtt{z}.\, \mathtt{x} := !\,\mathtt{x} + 1; !\,\mathtt{x}]\!]_x \rho_2)$ is a member of $comp([\![1 \to \alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta'')$.
2. Prove that for $\Delta''' \sqsupseteq \Delta''$ we have that $([\![\lambda \mathtt{z}.\, \mathtt{z} \leq !\,\mathtt{x}]\!]_x \rho_1, [\![\lambda \mathtt{z}.\, \mathtt{true}]\!]_x \rho_2)$ is a member of $comp([\![\alpha \to \mathtt{bool}]\!]_\alpha[\alpha \mapsto \nu])(\Delta''')$.

By inspection of proof obligation 1 we arrive at the following two sub-obligations that we must address:

1.a. Let $\Delta''' \sqsupseteq \Delta''$ be arbitrary. Prove that $(\llbracket \mathtt{x} := !\,\mathtt{x} + \mathtt{1} \rrbracket_x \rho_1, \llbracket \mathtt{x} := !\,\mathtt{x} + \mathtt{1} \rrbracket_x \rho_2)$ is a member of $comp(\llbracket \mathtt{1} \rrbracket_\alpha [\alpha \mapsto \nu])(\Delta''')$.

1.b. Let $\Delta^\dagger \sqsupseteq \Delta'''$ be arbitrary. Prove that $(\llbracket !\,\mathtt{x} \rrbracket_x \rho_1, \llbracket !\,\mathtt{x} \rrbracket_x \rho_2)$ is a member of $comp(\llbracket \alpha \rrbracket_\alpha [\alpha \mapsto \nu])(\Delta^\dagger)$.

We now attack the sub-obligation 1.a head-on. Let $\Delta''' \sqsupseteq \Delta''$ be arbitrary. By definition of the untyped interpretation we derive that $\llbracket \mathtt{x} := !\,\mathtt{x} + \mathtt{1} \rrbracket_x \rho_1$ is

$$lookup\, \lambda_{l_1} \star \lambda v_1. \begin{cases} \eta(in_{\mathbb{Z}}(m+1)) & v_1 = in_{\mathbb{Z}}\, m \\ error & \text{otherwise} \end{cases} \star \lambda w_1.\, assign\, \lambda_{l_1}\, w_1$$

and the same for $\llbracket \mathtt{x} := !\,\mathtt{x} + \mathtt{1} \rrbracket_x \rho_2$, only exchange λ_{l_2} for λ_{l_1}. Take $(k_1, k_2) \in cont(\llbracket \mathtt{1} \rrbracket_\alpha [\alpha \mapsto \nu])(\Delta''')$ and $(t_1, t_2) \in states(\Delta''')$. Since $\Delta''' \sqsupseteq \Delta'$ we know that $\Delta'''.\mathcal{I}(n)$ is well-defined and equals Θ defined above, modulo a change of population. In particular there must be $m \in \omega$ such that $\Delta'''.\mathcal{I}(n).CP = P_m$ and we know that $l_1 \in \mathrm{dom}(t_1)$, $l_2 \in \mathrm{dom}(t_2)$ and that $t_1(l_1) = t_2(l_2) = in_{\mathbb{Z}}\, m$. Summing up we get

$$\begin{aligned} \llbracket \mathtt{x} := !\,\mathtt{x} + \mathtt{1} \rrbracket_x \rho_1\, k_1\, t_1 &= (\eta(in_{\mathbb{Z}}(m+1)) \star \lambda w_1.\, assign\, \lambda_{l_1}\, w_1)\, k_1\, t_1 \\ &= assign\, \lambda_{l_1}\, in_{\mathbb{Z}}(m+1)\, k_1\, t_1 \\ &= k_1\, in_1(*)\, t_1[l_1 \mapsto in_{\mathbb{Z}}(m+1)] \end{aligned}$$

and similarly that

$$\llbracket \mathtt{x} := !\,\mathtt{x} + \mathtt{1} \rrbracket_x \rho_2\, k_2\, t_2 = k_2\, in_2(*)\, t_2[l_2 \mapsto in_{\mathbb{Z}}(m+1)].$$

We now build $\Delta^\dagger$ as a copy of Δ''' with the one exception that $\Delta^\dagger.\mathcal{I}(n).CP = P_{m+1}$ which gives us $\Delta^\dagger \sqsupseteq \Delta'''$ and $(t_1[l_1 \mapsto in_{\mathbb{Z}}(m+1)], t_2[l_2 \mapsto in_{\mathbb{Z}}(m+1)]) \in states(\Delta^\dagger)$ and this sub-obligation is done. Note, amidst the technicalities, that we have just generated a new name $m+1$ and updated the population of island n correspondingly.

Sub-obligation 1.b is a bit shorter. Let $\Delta^\dagger \sqsupseteq \Delta'''$ be arbitrary. Take $(k_1, k_2) \in cont(\llbracket \alpha \rrbracket_\alpha [\alpha \mapsto \nu])(\Delta^\dagger)$ and $(t_1, t_2) \in states(\Delta^\dagger)$. As above, there must be $m \in \omega$ such that $\Delta^\dagger.\mathcal{I}(n).CP = P_m$ and we know that $l_1 \in \mathrm{dom}(t_1)$, $l_2 \in \mathrm{dom}(t_2)$ and that $t_1(l_1) = t_2(l_2) = in_{\mathbb{Z}}\, m$. But then we get

$$(\llbracket !\,\mathtt{x} \rrbracket_x \rho_1\, k_1\, t_1, \llbracket !\,\mathtt{x} \rrbracket_x \rho_2\, k_2\, t2) = (k_1\, (in_{\mathbb{Z}}\, m)\, t_1, k_2\, (in_{\mathbb{Z}}\, m)\, t_2).$$

And all we need to finish this sub-obligation is just to remark that

$$[\![\alpha]\!]_\alpha[\alpha \mapsto \nu](\Delta^\dagger) = \nu(\Delta^\dagger) = \{(in_{\mathbb{Z}}\, v, in_{\mathbb{Z}}\, v) \mid v \in P_m\} \ni (in_{\mathbb{Z}}\, m, in_{\mathbb{Z}}\, m).$$

Finally we tackle obligation 2. Let $\Delta''' \sqsupseteq \Delta''$ be arbitrary. We can derive that $[\![\mathtt{z} \leq ! \mathtt{x}]\!]_{x,z}\rho_1[z \mapsto v_1]\, k_1\, t_1$ is

$$lookup\, \lambda_{l_1} \star \lambda w_1. \begin{cases} \eta(in_{\mathbb{B}}\, 1) & v_1 = in_{\mathbb{Z}}\, k, w_1 = in_{\mathbb{Z}}\, m, k \leq m \\ \eta(in_{\mathbb{B}}\, 0) & v_1 = in_{\mathbb{Z}}\, k, w_1 = in_{\mathbb{Z}}\, m, k > m \\ error & \text{otherwise.} \end{cases}$$

Pick $\Delta^\dagger \sqsupseteq \Delta'''$, $(v_1, v_2) \in [\![\alpha]\!]_\alpha[\alpha \mapsto \nu])(\Delta^\dagger) = \nu(\Delta^\dagger)$, $(k_1, k_2) \in cont([\![\mathtt{bool}]\!]_\alpha[\alpha \mapsto \nu](\Delta^\dagger)$ and $(t_1, t_2) \in states(\Delta^\dagger)$, we must show that

$$([\![\mathtt{z} \leq ! \mathtt{x}]\!]_{x,z}\rho_1[z \mapsto v_1]\, k_1\, t_1, [\![\mathtt{true}]\!]_{x,z}\rho_2[z \mapsto v_2]\, k_2\, t_2) \in R_{Ans}.$$

As above, there must be $m \in \omega$ such that $\Delta^\dagger.\mathcal{I}(n).CP = P_m$ and we know that $l_1 \in \mathrm{dom}(t_1)$, $l_2 \in \mathrm{dom}(t_2)$ and that $t_1(l_1) = t_2(l_2) = in_{\mathbb{Z}}\, m$. This also means that $(v_1, v_2) = (in_{\mathbb{Z}}\, k, in_{\mathbb{Z}}\, k)$ for some $1 \leq k \leq m$. Combining our efforts we arrive at

$$[\![\mathtt{z} \leq ! \mathtt{x}]\!]_{x,z}\rho_1[z \mapsto v_1]\, k_1\, t_1 = k_1(in_{\mathbb{B}}\, 1)\, t_1$$

and as we immediately have

$$[\![\mathtt{true}]\!]_{x,z}\rho_2[z \mapsto v_2]\, k_2\, t_2 = k_2\, (in_{\mathbb{B}} 1)\, t_2$$

we are done. Taking a few steps back, all that happens is that the interpretation of the type $[\![\alpha]\!]_\alpha[\alpha \mapsto \nu] = \nu$ ensures that the input values must be in the population of island n; also the heap law enforces that the related states both contain the maximum name of the population at location l_1 and l_2 respectively.

10.5. Discussion

We have written out the proof of the Name Generator example in much detail so as to make it easy to compare this proof with the one in the ADR model [3]. Looking at the two proofs we can conclude (as claimed in the introduction) that the semantic techniques from the BST model scale to state-of-the-art world descriptions and that the resulting model can be used

to prove programs equivalent at a fairly abstract level, without any form of low-level step-indexed reasoning. Indeed the proof we have given here in the model is at an abstraction level which is similar to the one provided by the LADR logic [23, Pages 56–58].

The same is the case for the other examples involving local state in (L)ADR.[4] The model similarly gives rise to fairly abstract proofs of the Plotkin-Power axioms for global state and local state [30] as formulated by Staton [34].[5]

For proving some equivalences of programs involving recursive types and/or reference types, LADR uses a so-called "later" modality and Löb's rule to abstractly account for induction over step-indices (This idea comes from [7].) For example, the later modality and Löb's rule are used to prove that Landin's knot—the construction of a fixed-point using backpatching—works [22, Section 9.3]. In the present model, this example would instead proceed via fixed-point induction.

Note that the proof of the name generator involves references and thus locations, but that there are no approximate locations in the proof at all. This is typical of examples that involve allocation and uses of references (such as the (L)ADR examples); approximate locations do not appear in the proofs since they are not used as denotations of references allocated in the programs. But, of course, there are examples, where they show up. Indeed, when one considers examples involving free variables of reference type (or, equivalently, functions that take arguments of reference type), then one also has to consider approximate locations. These approximate locations are in some sense additional junk in the denotational model, and because of them there are equivalences that we cannot prove using the model. The simplest concrete example we know of is the following [12]:

$$\emptyset \mid \emptyset \models \lambda x.\mathsf{true} \not\sim \lambda x.\mathsf{false} : \mathsf{ref}\, 0 \to \mathsf{bool}$$

Intuitively, these two functions should be contextually equivalent: since references are initialized when allocated, no closed value encountered in a running program can ever have the type $\mathsf{ref}\, 0$, and therefore neither of the two functions can ever be applied. However, the two functions are not semantically

[4]LADR cannot handle the "callback-with-lock" example of ADR and the same is the case with our model here (see [22] for further discussion).

[5]Three of the axioms, GS6, GS7, and B3, cannot be formulated as simple, typed equations in our language, but equivalent semantic formulations do hold in the model.

related in our model. Loosely speaking, the reason is that approximate locations can be related at the type $\mathsf{ref}\,0$.

We leave it as future work to investigate further if one can find a more abstract model, which does not involve either a form of semantic location or some form of step-indexing. We believe this is a challenging problem — for an earlier version of the BST model we could show that a putative logical relation formulated without approximate locations did not exist! — and it is related to questions of existence of recursively defined relations in [11].

Other future work includes the formulation of a program logic for reasoning about equivalence based on the present model. Such a logic would naturally combine ideas from LADR concerning syntactic formulations of islands, etc., with ideas from earlier domain-theoretically inspired logics for call-by-value (see, e.g., [1]). In particular it would not include the later modality and the Löb rule of LADR but rather have a fixed point induction rule.

Recently, Dreyer, Neis and Birkedal [21] have generalized the world-dynamics to have proper state transition systems instead of the populations and population laws of ADR. Their approach features two kinds of transitions: private and public; all computations are required to perform only public transitions as seen from the outside, but may realize these, internally, by means of private transitions. They go on to prove examples that could not be proved in the ADR model; indeed, they are able to prove all known local-state examples from the literature. Here we just remark, that we could easily extend the present model to worlds with state transition systems and private and public transitions; the shift is big in terms of ideas and applications, but on the technical / metric-space level it is a minor change.

11. Acknowledgements

We would like to thank Derek Dreyer and Georg Neis for helpful discussions and insightful comments including, but not limited to, observations on the consequences of choosing either intensional or extensional interpretations of references. And we thank the anonymous referees for many valuable comments, ranging from typos to directions for future work.

References

[1] M. Abadi and M. P. Fiore. Syntactic considerations on recursive types. In *Proceedings of LICS*, pages 242–252. IEEE Computer Society, 1996.

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

[3] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *Proceedings of POPL*, pages 340–353. ACM Press, 2009.

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

[5] R. M. Amadio. Recursion over realizability structures. *Inf. Comput.*, 91 (1):55–85, 1991.

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

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

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

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

[10] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *Proceedings of TLCA*, volume 3461 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2005.

[11] N. Benton, A. Kennedy, L. Beringer, and M. Hofmann. Relational semantics for effect-based program transformations: higher-order store. In *Proceedings of PPDP*, pages 301–312. ACM Press, 2009.

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

Proceedings of FOSSACS, volume 5504 of *Lecture Notes in Computer Science*, pages 456–470. Springer, 2009.

[13] L. Birkedal, K. Støvring, and J. Thamsborg. Relational parametricity for references and recursive types. In *Proceedings of TLDI*, pages 91–104. ACM Press, 2009.

[14] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations. Technical Report TR-2009-119, IT University of Copenhagen, August 2009.

[15] L. Birkedal, K. Støvring, and J. Thamsborg. Solutions of generalized recursive metric-space equations. In *Proceedings of the 6th Workshop on Fixed Points in Computer Science*, pages 18–24, 2009.

[16] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. Technical Report TR-2010-124, IT University of Copenhagen, January 2010.

[17] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Proceedings of APLAS*, volume 4279 of *Lecture Notes in Computer Science*, pages 79–96. Springer, 2006.

[18] J. de Bakker and E. de Vink. *Control flow semantics*. MIT Press, Cambridge, MA, USA, 1996.

[19] J. de Bakker and J. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[20] D. Dreyer, A. Ahmed, and L. Birkedal. Logical step-indexed logical relations. In *Proceedings of LICS*, pages 71–80. IEEE Computer Society, 2009.

[21] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. In P. Hudak and S. Weirich, editors, *ICFP*, pages 143–156. ACM, 2010. ISBN 978-1-60558-794-3.

[22] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs. In *Proceedings of POPL*, pages 185–198. ACM Press, 2010.

[23] D. Dreyer, G. Neis, A. Rossberg, and L. Birkedal. A relational modal logic for higher-order stateful ADTs (technical appendix). Available at `http://www.mpi-sws.org/~dreyer/papers/ladr/appendix.pdf`, 2010.

[24] P.-A. Melliès and J. Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *Proceedings of LICS*, pages 82–91. IEEE Computer Society, 2005.

[25] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[26] B. C. Pierce. *Types and programming languages*. MIT Press, Cambridge, MA, USA, 2002.

[27] A. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. MIT Press, 2005.

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

[29] A. M. Pitts and I. D. B. Stark. Operational reasoning for functions with local state. In *Higher order operational techniques in semantics*, pages 227–274. Cambridge University Press, New York, NY, USA, 1998.

[30] G. D. Plotkin and J. Power. Notions of computation determine monads. In *Proceedings of FOSSACS*, volume 2303 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2002.

[31] J. C. Reynolds. Types, abstraction, and parametric polymorphism. In *Information Processing 83, Paris, France*, pages 513–523. Elsevier, 1983.

[32] M. B. Smyth. Topology. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

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

[34] S. Staton. Completeness for algebraic theories of local state. In *Proceedings of FOSSACS*, 2010. To appear. Available at `http://www.cl.cam.ac.uk/~ss368/fossacs10.pdf`.

[35] G. Winskel. *The formal semantics of programming languages: an introduction*. MIT Press, Cambridge, MA, USA, 1993.