Transfinite Step-Indexing: Decoupling Concrete and Logical Steps

Kasper Svendsen, Filip Sieczkowski, Lars Birkedal

To cite this version:

Kasper Svendsen, Filip Sieczkowski, Lars Birkedal. Transfinite Step-Indexing: Decoupling Concrete and Logical Steps. 25th European Symposium on Programming Languages and Systems, Dec 2016, Eindhoven, Netherlands. pp.727 - 751, 10.1007/978-3-662-49498-1_28 . hal-01408649

HAL Id: hal-01408649

https://hal.inria.fr/hal-01408649

Submitted on 14 Dec 2016

Transfinite Step-indexing: Decoupling Concrete and Logical Steps

Kasper Svendsen[1], Filip Sieczkowski[2], and Lars Birkedal[3]

[1] University of Cambridge, ks775@cl.cam.ac.uk
[2] INRIA, filip.sieczkowski@inria.fr
[3] Aarhus University, birkedal@cs.au.dk

Abstract. Step-indexing has proven to be a powerful technique for defining logical relations for languages with advanced type systems and models of expressive program logics. In both cases, the model is stratified using natural numbers to solve a recursive equation that has no naive solutions. As a result of this stratification, current models require that each unfolding of the recursive equation – each logical step – must coincide with a concrete reduction step. This tight coupling is problematic for applications where the number of logical steps cannot be statically bounded.
In this paper we demonstrate that this tight coupling between logical and concrete steps is artificial and show how to loosen it using transfinite step-indexing. We present a logical relation that supports an arbitrary but finite number of logical steps for each concrete step.

1 Introduction

Step-indexing has proven to be a powerful technique for defining models of advanced type systems [5, 2, 6, 11, 10] and expressive higher-order program logics [16, 4, 20, 18]. To support abstraction, such type systems and program logics often feature some notion of *impredicative invariants*. For instance, a reference type can be seen as an invariant about the type of values stored at a given location; for languages with general references this is an impredicative invariant.

Modelling impredicative invariants is difficult and a naive approach naturally leads to a circular definition with no solution. To illustrate, consider modelling a language with general references. A natural idea is to interpret types relative to a world (heap typing) that assigns semantic types to all currently allocated locations. The reference type, $\tau\ \mathsf{ref}$, can then be interpreted as the set of locations mapped to $[\![\tau]\!]$ in the current world. A world is a finite function from locations to types. Unfortunately, this idea leads to a circular definition of the semantic domain of types that has no solution in set theory:

$$\mathrm{Type} \cong \mathrm{World} \overset{mon}{\to} \mathcal{P}(\mathrm{Val}) \qquad\qquad \mathrm{World} = \mathrm{Loc} \overset{fin}{\rightharpoonup} \mathrm{Type}$$

To break this circularity, step-indexed models interpret inhabitance of a type as a predicate indexed by the number of steps left "on the clock". This allows for a well-founded definition over the steps, by "going down a step" whenever an impredicative invariant is unfolded. We refer to these as *logical steps*. Since reference types require an unfolding of an impredicative invariant, the interpretation of reference types introduces a logical step: a location l is an inhabitant of type $\tau\ \mathsf{ref}$ with $n+1$ steps left on the clock, if location l contains a value that is an inhabitant of type τ with n steps left on the clock.

To use this artificially stratified model, current step-indexed models take the steps to be concrete reduction steps in the underlying operational semantics. An expression e is thus an inhabitant of type τ with n steps left on the clock, if whenever it reduces to e' in $i < n$ *concrete reduction steps*, then e' is an inhabitant of τ with $n-i$ steps left on the clock. As a consequence, each logical step must be associated with a corresponding reduction step.

This suffices to prove many interesting properties. For instance, to prove type soundness of the typing rule for dereference, we have to prove that $!l$ is an inhabitant of τ for n steps, assuming l is an inhabitant of $\tau\ \mathsf{ref}$ for n steps. Since $!l$ uses one concrete reduction step, it suffices to know that location l contains a value that is an inhabitant of τ for $n-1$ steps. Conceptually, the logical step of unfolding the invariant always happens together with the concrete step of dereferencing the location and the proof goes through. Unfortunately, this is not always the case. For instance, higher-order abstractions often introduce new logical steps without introducing corresponding concrete steps [18].

This is most easily illustrated in the setting of binary logical relations. To prove that e_1 logically approximates e_2, $\Gamma \models e_1 \leq_{log} e_2 : \tau$, current step-indexed logical relations require that each logical step must be associated with a reduction step of e_1. Let τ denote the type

$$\exists \alpha.\ (1 \to \alpha) \times (\alpha \to \mathbb{N})$$

and $f : \tau \to \tau$ denote the following function

$$\lambda x : \tau.\ \mathsf{unpack}\ x\ \mathsf{as}\ (\alpha, y)\ \mathsf{in}\ \mathsf{pack}\ (\alpha\ \mathsf{ref}\,, (\lambda z : 1.\ \mathsf{ref}\ (\mathsf{fst}\ y\ z), \lambda z : \alpha.\ \mathsf{snd}\ y\ (!z)))$$

The function f takes a τ ADT and returns a new one that wraps instances of the ADT in an additional reference. One might hope to be able to prove that $x : \tau \models x \cong_{log} f(x)$ and thus that the ADT returned by f is contextually equivalent to its argument. However, as far as we are aware, no current step-indexed logical relations offer a practical way of proving the left-to-right approximation, $x : \tau \models x \leq_{log} f(x)$.[1] The additional indirection through the heap forces us to introduce an invariant for each instance of the two ADTs we create. To relate the two instances we must unfold this invariant. Unfortunately, there is no reduction step on the left to justify this unfolding.

[1] There are step-indexed logical relations, which are *complete* wrt. contextual approximation [15], which would seem to suggest that they should be flexible enough to prove this example, but completeness is achieved using biorthogonality (a kind of closure under evaluation contexts), which means that while the models are technically complete, they do not offer a practical way of proving this kind of example; cf. the discussion in Section 10 of [15].

Similar problems plague step-indexed program logics, where higher-order abstractions that introduce new logical steps without any corresponding concrete steps are more common. For instance, deriving a new simpler specification for a specific way of using a concurrent data structure from an abstract library specification often introduces a new invariant without introducing any new concrete reduction steps [20, 18]. See [18, Section 8] for a concrete example of this problem in the context of step-indexed program logics. Existing step-indexing program logics have dealt with this issue by adding additional *skip* statements, to allow new logical steps to be related to *skip* reduction steps [14, 18]. Not only is this solution semantically unsatisfying, it is also insufficient for examples where the number of logical unfoldings cannot be statically bounded.

Conceptually, the problem with earlier models is the artificial link made between unfoldings of the recursive domain equation (logical steps) and concrete reduction steps. In this paper we propose a refinement of step-indexing that relaxes this link, by allowing an arbitrary but finite number of logical steps for each concrete reduction step. To achieve this, we stratify the construction of the semantic domain using ω^2, or $\mathbb{N} \times \mathbb{N}$ ordered lexicographically, instead of ω. We then take the first step index to be concrete reduction steps in the underlying operational semantics and the second step index to be the number of logical steps possible before the next concrete reduction step. The lexicographic ordering ensures that we can pick an arbitrary but finite number of possible logical steps until the next concrete reduction step, after each concrete reduction step.

Instead of presenting a transfinitely step-indexed model of our latest and greatest program logic, we focus on a logical relation. Such a setting is simpler, thus allowing us to present our solution in greater detail and concentrate on the problem of decoupling logical and concrete steps. We develop a general theory for stratifying the construction of semantic domains using step-indexing over ω^2. This theory also applies to semantic domains used in recent step-indexed program logics and we believe our approach to decoupling logical and concrete steps also extends to these program logics.

Our main contribution is conceptual: we demonstrate how transfinite step-indexing allows us to loosen the artificial link between concrete and logical steps in step-indexed models. As a technical contribution, we extend previous results for stratifying recursive definitions using step-indexing over ω to the transfinite case of ω^2.

We have included proof sketches of the main results in the article. Full proofs can be found in the accompanying technical report, which is available at the following address: `http://www.kasv.dk/transfinite-tr.pdf`.

Outline. First we introduce the syntax and operational semantics of a simple higher-order language in Section 2. In Section 3 we show how to stratify the construction of semantic domains using step-indexing over ω^2 and recall some mathematical concepts for working with these domains. Next, we apply this theory to define a transfinitely step-indexed logical relation in Section 4. In Section 5 we return to the example mentioned above and prove the troublesome contextual equivalence using our transfinitely step-indexed logical relation. We

$$\begin{aligned}
\tau, \sigma &::= 1 \mid \mathbb{N} \mid \tau \times \sigma \mid \tau \to \sigma \mid \tau\ \mathsf{ref}\ \mid \exists\alpha.\ \tau \mid \alpha \\
v \in \mathrm{Val} &::= * \mid \underline{n} \mid (v_1, v_2) \mid \mathsf{fix}\ f(x).\ e \mid l \mid \mathsf{pack}\ v \\
e \in \mathrm{Exp} &::= * \mid \underline{n} \mid (e_1, e_2) \mid \mathsf{fst}\ e \mid \mathsf{snd}\ e \mid \mathsf{fix}\ f(x).\ e \mid e_1\ e_2 \\
&\quad\mid\ \ l \mid !\,e \mid e_1 := e_2 \mid \mathsf{ref}\ e \mid \mathsf{pack}\ e \mid \mathsf{unpack}\ e_1\ \mathsf{as}\ x\ \mathsf{in}\ e_2
\end{aligned}$$

Fig. 1. Types, values and expressions.

defer most discussions of related work to Section 6. Finally, in Section 7 we conclude and discuss future work.

2 Syntax and operational semantics

In Figure 1 we define the syntax of a higher-order functional language with general references and existential types. This set of language features and type system suffices to study the problems mentioned in the Introduction. We assume countably infinite and disjoint sets of type variables, term variables and locations, with α ranging over type variables and x over term variables and l over locations. We use a Curry-style presentation and thus do not annotate λ-abstractions or pack/unpack with types. The typing rules have the form $\varDelta; \varGamma \vdash e : \tau$, where $\varDelta$ is a context of type variables and $\varGamma$ is a context of term variables. The well-formed type judgment, $\varDelta \vdash \tau$ expresses that all free type variables in τ are bound in $\varDelta$. Figure 2 includes an excerpt of typing rules; the remaining rules are standard and have been omitted.

$$\frac{\varDelta; \varGamma \vdash e : \tau\ \mathsf{ref}}{\varDelta; \varGamma \vdash !\,e : \tau} \qquad \frac{\varDelta; \varGamma \vdash e_1 : \tau\ \mathsf{ref} \qquad \varDelta; \varGamma \vdash e_2 : \tau}{\varDelta; \varGamma \vdash e_1 := e_2 : 1} \qquad \frac{\varDelta; \varGamma \vdash e : \tau}{\varDelta; \varGamma \vdash \mathsf{ref}\ e : \tau\ \mathsf{ref}}$$

$$\frac{\varDelta; \varGamma \vdash e_1 : \exists\alpha.\ \tau \qquad \varDelta, \alpha; \varGamma, x : \alpha \vdash e_2 : \sigma \qquad \varDelta \vdash \sigma}{\varDelta; \varGamma \vdash \mathsf{unpack}\ e_1\ \mathsf{as}\ x\ \mathsf{in}\ e_2 : \sigma}$$

Fig. 2. Excerpt of typing rules.

Note that our type system does not include store typings, assigning types to locations. Store typings are typically used to facilitate syntactic progress and preservation proofs. However, they are unnecessary for our semantic approach and we only consider source programs that do not contain location constants.

The operational semantics is defined as a small-step reduction relation between configurations consisting of an expression e and a heap h: $\langle e, h\rangle \to \langle e', h'\rangle$. A heap is a finite map from locations to values. Figure 3 includes an excerpt of the reduction rules; the rest of the rules are standard. Note that dereferencing or assigning to a location that has not already been allocated results in a stuck configuration. It will follow from our logical relation that well-typed programs never get stuck and thus never try to dereference a location that has not been allocated.

From the small-step reduction semantics we define a step-indexed reduction relation, $\langle e, h\rangle \to^n \langle e', h'\rangle$, which expresses that $\langle e, h\rangle$ reduces in n steps to $\langle e', h'\rangle$. We count every reduction step. Note that while our step-indexed logical relation will be indexed using ω^2, the operational semantics is still only indexed over ω, as is standard in step-indexed models. We use $\langle e, h\rangle \to^* \langle e', h'\rangle$ to denote the reflexive transitive closure of the small-step reduction relation.

$$
\begin{array}{rcl}
h \in \text{Heap} & \overset{def}{=} & \text{Loc} \overset{fin}{\rightharpoonup} \text{Val} \\
K \in \text{ECtx} & ::= & \bullet \mid (K, e) \mid (v, K) \mid \mathsf{fst}\ K \mid \mathsf{snd}\ K \mid K\ e \mid v\ K \\
& \mid & !\,K \mid K := e \mid v := K \mid \mathsf{ref}\ K \mid \mathsf{pack}\ K \mid \mathsf{unpack}\ K\ \mathsf{as}\ x\ \mathsf{in}\ e \\
C \in \text{Ctx} & ::= & \bullet \mid (C, e) \mid (e, C) \mid \mathsf{fst}\ C \mid \mathsf{snd}\ C \mid C\ e \mid e\ C \mid !\,C \mid C := e \mid e := C \mid \mathsf{ref}\ C \\
& \mid & \mathsf{pack}\ C \mid \mathsf{unpack}\ C\ \mathsf{as}\ x\ \mathsf{in}\ e \mid \mathsf{unpack}\ e\ \mathsf{as}\ x\ \mathsf{in}\ C
\end{array}
$$

$$
\overset{\textsc{EvalRead}}{\frac{l \in \mathrm{dom}(h)}{\langle !\,l, h\rangle \to \langle h(l), h\rangle}}
\qquad
\overset{\textsc{EvalWrite}}{\frac{l \in \mathrm{dom}(h)}{\langle l := v, h\rangle \to \langle *, h[l \mapsto v]\rangle}}
\qquad
\overset{\textsc{EvalAlloc}}{\frac{l \notin \mathrm{dom}(h)}{\langle \mathsf{ref}\ v, h\rangle \to \langle l, h[l \mapsto v]\rangle}}
$$

$$
\overset{\textsc{EvalUnpack}}{\frac{}{\langle \mathsf{unpack}\ (\mathsf{pack}\ v)\ \mathsf{as}\ x\ \mathsf{in}\ e, h\rangle \to \langle e[v/x], h\rangle}}
\qquad
\overset{\textsc{EvalCtx}}{\frac{\langle e, h\rangle \to \langle e', h'\rangle}{\langle K[e], h\rangle \to \langle K[e'], h'\rangle}}
$$

$$
\frac{}{\langle e, h\rangle \to^0 \langle e, h\rangle}
\qquad
\frac{\langle e, h\rangle \to \langle e', h'\rangle \qquad \langle e', h'\rangle \to^n \langle e'', h''\rangle}{\langle e, h\rangle \to^{n+1} \langle e'', h''\rangle}
$$

Fig. 3. Excerpt of reduction rules.

Our formal definition of contextual approximation is given in Definition 1 below. We say that e_I contextually approximates e_S if for any closing context C of unit type, if $C[e_I]$ terminates with value $*$, then $C[e_S]$ terminates with value $*$. Since well-typed expressions do not contain any location constants, we can simply reduce $C[e_I]$ and $C[e_S]$ with an empty heap. The $C : (\Delta; \Gamma, \tau) \rightsquigarrow (\Delta'; \Gamma', \tau')$ relation expresses that context C takes a term e such that $\Delta; \Gamma \vdash e : \tau$ to a term $C[e]$ such that $\Delta'; \Gamma' \vdash C[e] : \tau'$. The rules for $C : (\Delta; \Gamma, \tau) \rightsquigarrow (\Delta'; \Gamma', \tau')$ are standard and have been omitted.

Definition 1 (Contextual approximation) *If $\Delta; \Gamma \vdash e_I : \tau$ and $\Delta; \Gamma \vdash e_S : \tau$, then e_I contextually approximates e_S, written $\Delta; \Gamma \vdash e_I \leq_{ctx} e_S : \tau$ iff,*

$$
\begin{array}{l}
\forall C : (\Delta; \Gamma, \tau) \rightsquigarrow (-; -, 1). \\
\quad \forall h_I \in \text{Heap}.\ \langle C[e_I], []\rangle \to^* \langle *, h_I\rangle \Rightarrow \exists h_S \in \text{Heap}.\ \langle C[e_S], []\rangle \to^* \langle *, h_S\rangle.
\end{array}
$$

We refer to e_I in $\Delta; \Gamma \vdash e_I \leq e_S : \tau$ as the left expression or implementation and we refer to e_S as the right expression or specification. Contextual equivalence,

$\Delta; \Gamma \vdash e_1 \cong_{ctx} e_2 : \tau$, is then defined as the conjunction of left-to-right contextual approximation and right-to-left contextual approximation.

3 Step-indexing over ω^2

Step-indexing is often used to solve the recursive definitions of semantic domains that arise when modelling impredicative invariants. Due to a contravariant occurrence of the recursive variable, these definitions have no solution in set-theory. Instead, step-indexing is used to stratify the construction. In this section we present a general theory for stratifying the construction of semantic domains using step-indexing over ω^2.

The idea is to define the semantic domain as a fixed point of a suitably *contractive* functor over a category of step-indexed sets. For instance, recall the type-world circularity from the Introduction. Instead of solving the original equation, we wish to solve the following equation in a category of step-indexed sets:

$$\mathbf{Type} \cong \blacktriangleright((\mathrm{Loc} \overset{\mathit{fin}}{\rightharpoonup} \mathbf{Type}) \overset{\mathit{mon}}{\to} \mathbf{UPred}(\mathrm{Val}))$$

Here the *later* operator ($\blacktriangleright$) ensures that we "go down a step" when unfolding the recursive definition. Intuitively, this ensures that **Type** at step-index i is only defined in terms of **Type** at strictly smaller step-indices, $j < i$, and thus that the equation has a solution. Since the result needs to be a step-indexed set, we have also replaced predicates over values ($\mathcal{P}(\mathrm{Val})$) with its step-indexed counterpart: uniform predicates over values ($\mathbf{UPred}(\mathrm{Val})$).

For step-indexing over ω one can apply general existence theorems for fixed points of locally contractive functors on the category of ω set-indexed sets. Birkedal, Støvring and Thamsborg [12] have generalized the inverse-limit construction to show the existence of fixed points of locally contractive functors over categories where the Hom-sets can be equipped with a suitable metric structure. This existence theorem is directly applicable to the category of ω step-indexed sets. Unfortunately, the results do not apply to the category of ω^2 step-indexed sets. Intuitively, the inverse limit construction is not iterated far enough for step-indexing over ω^2.

In this section we define a category of ω^2 step-indexed sets, $\mathcal{U}$, and give a concrete construction for fixed-points of locally contractive functors on $\mathcal{U}$. While the results for step-indexing over ω^2 are novel, the structure of the development is not and follows existing work. To simplify the exposition we use Di Gianantonio and Miculan's complete ordered families of equivalences [13] to present the category of step-indexed sets and avoid the use of abstract category theory.

Complete ordered families of equivalences. Ordered families of equivalences (o.f.e.) over ω^2 are pairs consisting of a set X and a family of equivalence relations on X, indexed by ω^2. The intuition to keep in mind is that the (n, m)-th equivalence relation, $\overset{n,m}{=}$, expresses equality on the underlying set when there is n, m steps left "on the clock". A priori, these steps are purely artificial to break the circularity, however, in the following section we will relate the first step-index n with concrete reduction steps. We will thus think of $\overset{n,m}{=}$ as equality on the underlying set, with n concrete reduction steps left and m logical steps left before the next concrete reduction step. We use a lexicographic ordering on ω^2:

$$(n_1, m_1) \leq (n_2, m_2) \overset{\mathit{def}}{=} (n_1 = n_2 \wedge m_1 \leq m_2) \vee n_1 < n_2$$

Thus, every time we take one concrete reduction step, we will be able to chose an arbitrary finite number of logical steps that we may do before the next concrete reduction step.

Another intuition to keep in mind is that two elements are (n, m)-equivalent if the elements cannot distinguished with n concrete reduction steps left and m logical steps before the next concrete step. Since fewer observations are possible as the number of steps decreases, we require the equivalence to become coarser as the number of steps is decreased. With no reduction steps left no observations are possible and every element is indistinguishable. We thus require that the equivalence is the total relation at the last step index, $(0, 0)$. Lastly, we require that if two elements are indistinguishable for any number of steps, they are identical.

Definition 2 (Ordered families of equivalences over ω^2) *An ordered family of equivalence relations (o.f.e.) over ω^2 is a pair $(X, (\overset{a}{=})_{a \in \omega^2})$, consisting of a set X and an ω^2-indexed set of equivalence relations $\overset{a}{=}$, satisfying*

- $\forall x, y \in X.\ x \overset{0,0}{=} y$
- $\forall x, y \in X.\ \forall a, b \in \omega^2.\ a \leq b \wedge x \overset{b}{=} y \Rightarrow x \overset{a}{=} y$
- $\forall x, y \in X.\ (\forall a \in \omega^2.\ x \overset{a}{=} y) \Rightarrow x = y$

The function space between two ordered families of equivalences consists of non-expansive functions that map a-equivalent arguments to a-equivalent results. Intuitively, non-expansiveness of f ensures that it takes at least the same number of steps to distinguish $f(x_1)$ from $f(x_2)$ as it takes to distinguish x_1 from x_2. Similarly, a function f is contractive if it is strictly harder to distinguish $f(x_1)$ from $f(x_2)$ than it is to distinguish x_1 from x_2, i.e., if to show that the results are a-equivalent it suffices that the arguments are equivalent at all *strictly smaller* indices. Contractive functions are thus necessarily non-expansive.

Definition 3 (Non-expansive and contractive functions) *Let $\mathcal{X} = (X, (\overset{a}{=}_X)_{a \in \omega^2})$ and $\mathcal{Y} = (Y, (\overset{a}{=}_Y)_{a \in \omega^2})$ be ordered families of equivalences. A function $f : X \to Y$ is non-expansive iff*

$$\forall x_1, x_2 \in X.\ \forall a \in \omega^2.\ x_1 \overset{a}{=}_X x_2 \Rightarrow f(x_1) \overset{a}{=}_Y f(x_2)$$

and contractive iff

$$\forall x_1, x_2 \in X.\ \forall a \in \omega^2.\ (\forall b \in \omega^2.\ b < a \Rightarrow x_1 \overset{b}{=}_X x_2) \Rightarrow f(x_1) \overset{a}{=}_Y f(x_2).$$

By iterating a contractive function, we get a sequence of elements that require more and more steps to distinguish. For *complete* ordered families of equivalences, where all such sequences have limits, we can define a unique fixed point of the contractive function as a suitable limit. When step-indexing over ω it suffices to iterate a contractive function f up to ω:

$$*, f(*), f^2(*), ...$$

as the fixed point x_1 of f is simply the limit of the above sequence. For step-indexing over ω^2 we have to continue the sequence further, taking the limit of all the previous elements at each limit index $(n, 0)$ and iterating f over these intermediate limits:

$$*, f(*), f^2(*), ..., x_1, f(x_1), f^2(x_1), ..., x_2, f(x_2), f^2(x_2), ...$$

Here x_i is the limit of all the previous elements. The fixed point is then the limit of this extended "sequence".

Definition 4 (Limits) *Let $(X, (\overset{a}{=})_{a\in\omega^2})$ be an ordered family of equivalences and Y a subset of ω^2. A Y-indexed family $(x_y)_{y\in Y}$ is coherent iff $\forall a, b \in Y.\ a \leq b \Rightarrow x_a \overset{a}{=} x_b$ and x is a limit of $(x_y)_{y\in Y}$ iff $\forall a \in Y.\ x \overset{a}{=} x_a$.*

Note that limits of coherent families indexed by proper subsets of ω^2 are not necessarily unique. For such coherent families we thus additionally require suitably unique *chosen* limits.

Definition 5 (Chosen limits) *Let $(X, (\overset{a}{=})_{a\in\omega^2})$ be an ordered family of equivalences. Then $(X, (\overset{a}{=})_{a\in\omega^2})$ has chosen limits iff any $b \in \omega^2$ there exists a function $\lim_{a<b} x_a$ that maps a coherent family $(x_a)_{a<b}$ indexed by $\{a \in \omega^2 \mid a < b\}$ to a limit, such that for any two coherent families $(x_a)_{a<b}$ and $(y_a)_{a<b}$ indexed by $\{a \in \omega^2 \mid a < b\}$,*

$$(\forall a < b.\ x_a \overset{a}{=} y_a) \Rightarrow \lim_{a<b} x_a = \lim_{a<b} y_a$$

Definition 6 (Complete ordered families of equivalences) *A complete ordered family of equivalences (c.o.f.e) is an o.f.e. $\mathcal{X}$ such that all ω^2-indexed coherent families in $\mathcal{X}$ have limits and $\mathcal{X}$ has chosen limits.*

Lemma 1 (Banach's fixed point theorem) *Let $\mathcal{X} = (X, (\overset{a}{=})_{a\in\omega^2})$ be a complete ordered family of equivalences and $f : X \to X$ a contractive function. If $\mathcal{X}$ is inhabited (i.e., there exists an $x \in X$), then f has a unique fixed point.*

Complete ordered families of equivalences over ω^2 form a category, $\mathcal{U}$, with non-expansive functions as morphisms. We will use this category to define our semantic domains. To do so, we first introduce a few basic c.o.f.e. constructions followed by a general existence theorem for solutions of recursive domain equations in $\mathcal{U}$.

Definition 7 *Let $\mathcal{U}$ denote the category of complete ordered families of equivalences. The objects of $\mathcal{U}$ are complete ordered families of equivalences and the morphisms are non-expansive functions.*

Basic constructions. The set of non-expansive functions between two c.o.f.e.s forms a c.o.f.e., by lifting the equivalence on the co-domain and computing limits point-wise. Restricting the function space further to finite partial non-expansive functions also yields a c.o.f.e.

Lemma 2 *Let $\mathcal{X} = (X, (\overset{a}{=}_X)_{a \in \omega^2})$ and $\mathcal{Y} = (Y, (\overset{a}{=}_Y)_{a \in \omega^2})$ be complete ordered families of equivalences. Then $\mathcal{X} \to_{ne} \mathcal{Y}$ and $\mathcal{X} \overset{\mathit{fin}}{\rightharpoonup} \mathcal{Y}$ are complete ordered families of equivalences, where*

$$\mathcal{X} \to_{ne} \mathcal{Y} \overset{\mathit{def}}{=} (\{f : X \to Y \mid f \textit{ is non-expansive}\}, (\overset{a}{=}_{\mathcal{X} \to \mathcal{Y}})_{a \in \omega^2})$$
$$\mathcal{X} \overset{\mathit{fin}}{\rightharpoonup} \mathcal{Y} \overset{\mathit{def}}{=} (\{f : X \overset{\mathit{fin}}{\rightharpoonup} Y \mid f \textit{ is non-expansive}\}, (\overset{a}{=}_{\mathcal{X} \overset{\mathit{fin}}{\rightharpoonup} \mathcal{Y}})_{a \in \omega^2})$$

and

$$f \overset{a}{=}_{\mathcal{X} \to \mathcal{Y}} g \quad \textit{iff} \quad \forall x \in X.\ f(x) \overset{a}{=}_Y g(x)$$
$$f \overset{a}{=}_{\mathcal{X} \overset{\mathit{fin}}{\rightharpoonup} \mathcal{Y}} g \quad \textit{iff} \quad \mathrm{dom}(f) = \mathrm{dom}(g) \wedge \forall x \in \mathrm{dom}(f).\ f(x) \overset{a}{=}_Y g(x)$$

Likewise, restricting the non-expansive function space to monotone and non-expansive functions also yields a c.o.f.e. if the partial order on the codomain respects limits in the codomain.

Lemma 3 *Let $\mathcal{X} = (X, (\overset{a}{=}_X)_{a \in \omega^2})$ be an ordered family of equivalences, $\mathcal{Y} = (Y, (\overset{a}{=}_Y)_{a \in \omega^2})$ be complete ordered families of equivalences and $\leq_X$ and $\leq_Y$ be partial orders on X and Y.*

If for any two coherent families $(x_a)_{a \in \omega^2}$ and $(y_a)_{a \in \omega^2}$

$$\forall a \in \omega^2.\ x_a \leq_Y y_a \Rightarrow \lim_a x_a \leq_Y \lim_a y_a$$

and for any $b \in \omega^2$ and any two coherent families $(x_a)_{a<b}$ and $(y_a)_{a<b}$

$$\forall a < b.\ a_a \leq_Y y_a \Rightarrow \lim_{a<b} x_a \leq_Y \lim_{a<b} y_a$$

then $\mathcal{X} \overset{mon}{\to} \mathcal{Y}$ is a complete ordered family of equivalences, where

$$\mathcal{X} \overset{mon}{\to} \mathcal{Y} \overset{\mathit{def}}{=} (\{f : X \overset{mon}{\to} Y \mid f \textit{ is non-expansive}\}, (\overset{a}{=}_{\mathcal{X} \to \mathcal{Y}})_{a \in \omega^2})$$

We can model a predicate over a set X as a predicate over $\omega^2 \times X$, downwards-closed in the step index. The intuition is that $(n, m, x) \in p$ if x satisfies the step-indexed predicate p with n, m steps left on the clock. The downwards-closure captures the intuition that it takes a certain number of steps to show that x does not satisfy a given predicate. These predicates form a c.o.f.e., where we consider two predicates n, m equivalent if they agree for all step-indices strictly below n, m.

Definition 8 (Uniform predicates, UPred(X)) *Let X be a set. Then*

$$\mathbf{UPred}(X) = (\mathcal{P}^{\downarrow}(\omega^2 \times X), (\overset{a}{=})_{a \in \omega^2})$$

is a c.o.f.e., where

$$p \overset{a}{=} q \quad \textit{iff} \quad \lfloor p \rfloor_a = \lfloor q \rfloor_a, \qquad \lfloor p \rfloor_a \overset{\textit{def}}{=} \{(b, x) \in p \mid b < a\}$$

and the ordering on $\omega^2 \times X$ is: $(a_1, x_1) \leq (a_2, x_2)$ iff $a_1 \leq a_2$ and $x_1 = x_2$.

Solving recursive domain equations in $\mathcal{U}$. Banach's fixed point theorem (Lemma 1) allows us to show existence of recursively-defined elements of complete ordered families of equivalences. To show existence of recursively-defined complete ordered families of equivalences, we lift the fixed point theorem from contractive functions on complete ordered families of equivalences to *locally contractive* functors on the category of complete ordered families of equivalences. The fixed point theorem is stated for mixed-variance functors, $F : \mathcal{U}^{op} \times \mathcal{U} \to \mathcal{U}$, where the negative and positive occurrences of the recursive variable have been separated.

Definition 9 (Locally non-expansive and locally contractive functor) *A bi-functor $F : \mathcal{U}^{op} \times \mathcal{U} \to \mathcal{U}$ is locally non-expansive iff*

$$\begin{aligned}&\forall \mathcal{X}, \mathcal{X}', \mathcal{Y}, \mathcal{Y}' \in \mathit{obj}(\mathcal{U}).\ \forall f, f' : \mathit{Hom}_{\mathcal{U}}(\mathcal{X}, \mathcal{X}').\ \forall g, g' : \mathit{Hom}_{\mathcal{U}}(\mathcal{Y}', \mathcal{Y}).\\ &\quad \forall a \in \omega^2.\ f \overset{a}{=} f' \wedge g \overset{a}{=} g' \Rightarrow F(f, g) \overset{a}{=} F(f', g')\end{aligned}$$

and locally contractive iff

$$\begin{aligned}&\forall \mathcal{X}, \mathcal{X}', \mathcal{Y}, \mathcal{Y}' \in \mathit{obj}(\mathcal{U}).\ \forall f, f' : \mathit{Hom}_{\mathcal{U}}(\mathcal{X}, \mathcal{X}').\ \forall g, g' : \mathit{Hom}_{\mathcal{U}}(\mathcal{Y}', \mathcal{Y}).\\ &\quad \forall a \in \omega^2.\ (\forall b \in \omega^2.\ b < a \Rightarrow f \overset{b}{=} f' \wedge g \overset{b}{=} g') \Rightarrow F(f, g) \overset{a}{=} F(f', g')\end{aligned}$$

In both cases, the equality on the Hom-set is the point-wise lifting of the equality on the co-domain (i.e., $f \overset{a}{=} f'$ iff $\forall x \in X.\ f(x) \overset{a}{=} f'(x)$).

Theorem 1 *If $F : \mathcal{U}^{op} \times \mathcal{U} \to \mathcal{U}$ is a locally contractive bi-functor and $F(1, 1)$ is inhabited, then there exists an object $X \in \mathcal{U}$ such that $F(X, X) \cong X \in \mathcal{U}$.*

Proof (Sketch). The construction of the fixed point uses an inverse-limit construction. However, as for Banach's fixed point theorem, when step-indexing over ω^2 we have to iterate the construction further and repeat the inverse-limit construction.

First we show that for any $S \in \mathcal{U}$ and projection/embedding pair

$$p_S : F(S, S) \to S \qquad\qquad e_S : S \to F(S, S)$$

and step-index n such that $p_S \circ e_S = id_S$ and $e_S \circ p_S \overset{n,0}{=} id_{F(S,S)}$, we can define an approximate fixed point, $X \in \mathcal{U}$ with projection/embedding pair

$$p_X : F(X, X) \to X \qquad\qquad e_X : X \to F(X, X)$$

such that $p_X \circ e_X = id_X$ and $\forall m.\ e_X \circ p_X \overset{n,m}{=} id_{F(X,X)}$. This approximate fixed point is constructed as an inverse-limit.

Next, we iterate this construction to obtain increasingly better approximations of the fixed point. The real fixed point is then constructed as an inverse limit of these approximate fixed points. □

Returning to the recursive equation from the beginning of this Section, we can reformulate the equation as a fixed point of a bi-functor, by separating the positive and negative occurrences as follows:

$$F(X^-, X^+) \cong (\blacktriangleright \circ G)(X^-, X^+), \quad G(X^-, X^+) \overset{\mathit{def}}{=} (\mathbb{N} \overset{\mathit{fin}}{\rightharpoonup} X^-) \overset{\mathit{mon}}{\to} \mathbf{UPred}(\mathrm{Val})$$

Exponentiation using both the non-expansive monotone function space and the non-expansive finite partial function space extends to locally non-expansive bi-functors. The functor G defined above is thus locally non-expansive.

Given a locally non-expansive functor it is possible to define a locally contractive functor by shifting all the equivalence relations one step. The functor $\blacktriangleright$, defined below, takes a complete ordered family of equivalences and shifts all the equivalences one step up, such that (n, m)-equivalence becomes $(n, m+1)$-equivalence. At limits $(n+1, 0)$ the equivalence is taken to be the limit of all smaller equivalences.

Definition 10 ($\blacktriangleright$) *Let $\blacktriangleright : \mathcal{U} \to \mathcal{U}$ denote the following functor,*

$$\blacktriangleright \left(X, (\overset{a}{=})_{a \in \omega^2}\right) \overset{\mathit{def}}{=} \left(X, (\overset{a}{\equiv})_{a \in \omega^2}\right) \qquad\qquad \blacktriangleright(f) \overset{\mathit{def}}{=} f$$

where $\overset{0,0}{\equiv}$ is the total relation on X, $\overset{n,m+1}{\equiv}$ is $\overset{n,m}{=}$ and $\overset{n+1,0}{\equiv}$ is defined as follows

$$x_1 \overset{n+1,0}{\equiv} x_2 \quad \mathit{iff} \quad \forall m \in \mathbb{N}.\ x_1 \overset{n,m}{=} x_2.$$

Lemma 4 *If $F : \mathcal{U}^{op} \times \mathcal{U} \to \mathcal{U}$ is locally non-expansive then $\blacktriangleright \circ F$ is locally contractive.*

Using Lemma 4 we can obtain a locally contractive functor from a locally non-expansive functor, and thus use Theorem 1 to obtain a fixed point. However, as we will see in the following section, due to the shifting by $\blacktriangleright$, we are forced to "go down a step" if we want our definitions to remain non-expansive, whenever we unfold the isomorphism.

4 Logical relation

In this section we define a logical relation step-indexed over ω^2 for the language introduced in Section 2. By only relating the first step-index with concrete reduction steps, we obtain a logical relation that allows for an arbitrary finite number of logical steps for each concrete reduction step. We prove that this logical relation is sound with respect to contextual approximation.

Semantic domains. To model dynamic allocation of references, we use a Kripke logical relation and index relations with worlds that contain general relational invariants over the heap. These invariants can themselves assert the existence of invariants and are thus also world-indexed. This leads to a recursive definition of the semantic domain of invariants with a contravariant occurrence of the recursive variable. Consequently, we use Theorem 1 to show the existence of a c.o.f.e. **Inv** satisfying the following isomorphism:

$$\xi : \mathbf{Inv} \cong \blacktriangleright((\mathbb{N} \overset{fin}{\rightharpoonup} \mathbf{Inv}) \overset{mon}{\to} \mathbf{UPred}(\mathrm{Heap} \times \mathrm{Heap}))) \tag{1}$$

An invariant is modelled as a uniform relation on heaps, indexed by a world that is itself a finite map from invariant identifiers ($\mathbb{N}$) to invariants. The monotone function space is with respect to extension ordering on $\mathbb{N} \overset{fin}{\rightharpoonup} \mathbf{Inv}$ and subset inclusion on $\mathbf{UPred}(-)$. The monotonicity requirement allows us to dynamically allocate new invariants, without invaliding existing invariants. A type is modelled as a uniform relation on values indexed by a world:

$$\mathbf{World} \overset{def}{=} \mathbb{N} \overset{fin}{\rightharpoonup} \mathbf{Inv} \qquad\qquad \mathbf{Type} \overset{def}{=} \mathbf{World} \overset{mon}{\to} \mathbf{UPred}(\mathrm{Val} \times \mathrm{Val})$$

We use $\widehat{\mathbf{Inv}}$ as a shorthand for $\mathbf{World} \to \mathbf{UPred}(\mathrm{Heap} \times \mathrm{Heap})$. Note that here we index worlds by invariant identifiers ($\mathbb{N}$) rather than the physical locations (Loc) we used in the Introduction. Due to our use of general relational invariants on heaps, it is no longer necessary to index the world with physical locations. Instead, we index the world with invariant identifiers which allows us to simplify the meta-theory by allowing us to refer to individual invariants.

A reference invariant is a particular instance of these general invariants. A reference invariant, $\mathrm{inv}(\nu, l_I, l_S) \in \widehat{\mathbf{Inv}}$, for type $\nu \in \mathbf{Type}$ and two locations l_I and l_S relates two heaps h_I and h_S if $h_I(l_I)$ and $h_S(l_S)$ are related at type ν:

$$\begin{aligned}\mathrm{inv}(\nu, l_I, l_S) \overset{def}{=} \lambda W.\ \{(n, m, h_I, h_S) \mid\ & l_I \in \mathrm{dom}(h_I) \wedge l_S \in \mathrm{dom}(h_S) \wedge \\ & (n, m, h_I(l_I), h_S(l_S)) \in \nu(W)\}\end{aligned}$$

The more general invariants supported by our model are not necessary to define the logical relation or prove that it implies contextual approximation. However, they are very useful when relating two concrete programs directly in the model, as we will see in Section 5.

Heap satisfaction. To define the relational interpretation of types, we first need to define heap satisfaction, which expresses when two heaps are related at a given world. Intuitively, this is the case when the heaps satisfy all the invariants in world. However, to support local reasoning about invariant satisfaction, we borrow the idea of ownership from separation logic [19] and require each invariant to hold for a *disjoint* part of the heap. Two heaps h_I and h_S are thus related, if they can be split into disjoint parts $h_{1I}, h_{2I}, ..., h_{kI}$ and $h_{1S}, h_{2S}, ..., h_{kS}$, respectively, such that h_{jI} and h_{jS} are related by the j'th invariant. To simplify the meta-theory we indexed heap satisfaction, $\lfloor W \rfloor_{\mathcal{I}}$, with a set of invariant identifiers $\mathcal{I} \subseteq \mathbb{N}$ indicating which invariants are active (i.e., currently required to hold).

$$\begin{aligned}\lfloor W \rfloor_{\mathcal{I}} \overset{def}{=} \{(n, h_I, h_S) \mid (\exists r_I, r_S : \ & \mathrm{dom}(W) \cap \mathcal{I} \to \mathrm{Heap}. \\ & h_I = \uplus_{\iota \in \mathrm{dom}(r_I)} r_I(\iota) \wedge h_S = \uplus_{\iota \in \mathrm{dom}(r_S)} r_S(\iota) \wedge \\ & \forall \iota \in \mathrm{dom}(W) \cap \mathcal{I}.\ \forall m \in \mathbb{N}. \\ & \quad (n-1, m, r_I(\iota), r_S(\iota)) \in \xi(W(\iota))(W)) \vee n = 0\}\end{aligned}$$

Note that heap satisfaction is only step-indexed using one step-index, n. This is because heap satisfaction always requires that each h_{jI} and h_{jS} must be related for an arbitrary number of logical steps m, no matter how many logical steps are currently left on the clock. This pattern of universally quantifying over the number of logical steps will reappear in many definitions and is crucial in allowing us to unfold an arbitrary finite number of invariants for each concrete reduction step. Intuitively, this bakes in the assumption and proof obligation that there is always an unbounded number of possible logical steps left before the next concrete reduction step in all our definitions. Except when we are relating concrete examples we never commit to an actual number of logical steps.

Since each invariant asserts exclusive ownership of the parts of each heap that are related by the invariant, we can use $\mathcal{I}$ to reason locally about satisfaction of individual invariants. This is captured by Lemma 5. We use $\lfloor W \rfloor$ as shorthand for heap satisfaction in the case where all invariants are active, i.e., $\lfloor W \rfloor_{\mathrm{dom}(W)}$.

Lemma 5 (Invariant locality)

$$\begin{array}{l}\forall W \in \mathbf{World}.\ \forall h_I, h_S \in \mathrm{Heap}.\ \forall n \in \mathbb{N}.\ \forall \mathcal{I}_1, \mathcal{I}_2 \in \mathcal{P}(\mathbb{N}).\\ \quad (n, h_I, h_S) \in \lfloor W \rfloor_{\mathcal{I}_1 \uplus \mathcal{I}_2} \ \Leftrightarrow \\ \qquad \left(\begin{array}{l}\exists h_{1I}, h_{2I}, h_{1S}, h_{2S} \in \mathrm{Heap}.\ h_I = h_{1I} \uplus h_{2I} \wedge h_S = h_{1S} \uplus h_{2S} \\ \qquad \wedge\, (n, h_{1I}, h_{1S}) \in \lfloor W \rfloor_{\mathcal{I}_1} \wedge (n, h_{2I}, h_{2S}) \in \lfloor W \rfloor_{\mathcal{I}_2}\end{array}\right)\end{array}$$

As expected, two heaps satisfy the heap invariant $\mathrm{inv}(\nu, l_I, l_S)$ if and only if $h_I(l_I)$ and $h_S(l_S)$ contain ν-related values (Lemma 6). Note that Lemma 6 requires that there is at least one concrete reduction step left, even just to prove that l_I and l_S are in the domain of h_I and h_S.

Lemma 6 (Reference invariant satisfaction)

$$\begin{array}{l}\forall n \in \mathbb{N}.\ \forall \iota \in \mathbb{N}.\ \forall W \in \mathbf{World}.\ \forall \nu \in \mathbf{Type}.\ \forall h_I, h_S \in \mathrm{Heap}.\ \forall l_I, l_S \in \mathrm{Loc}.\\ \quad n > 0 \wedge \xi(W(\iota)) \overset{n,0}{=} \mathrm{inv}(\nu, l_I, l_S) \Rightarrow \\ \quad \left(\begin{array}{l}(n, h_I, h_S) \in \lfloor W \rfloor_{\{\iota\}} \Leftrightarrow l_I \in \mathrm{dom}(h_I) \wedge l_S \in \mathrm{dom}(h_S)\ \wedge \\ \hfill \forall m.\ (n-1, m, h_I(l_I), h_S(l_S)) \in \nu(W)\end{array}\right)\end{array}$$

To ensure that definitions using heap satisfaction are suitably non-expansive, we require heap satisfaction to satisfy the following non-expansiveness property (Lemma 7). Intuitively, this property holds because we "go down" one step index from n to $n-1$ in the definition of heap satisfaction.

Lemma 7

$$\begin{array}{l}\forall W_1, W_2 \in \mathbf{World}.\ \forall h_I, h_S \in \mathrm{Heap}.\ \forall n, m \in \mathbb{N}.\\ \qquad W_1 \overset{n,m}{=}_{\mathbf{World}} W_2 \wedge (n, h_I, h_S) \in \lfloor W_1 \rfloor \Rightarrow (n, h_I, h_S) \in \lfloor W_2 \rfloor\end{array}$$

Proof (Sketch). Assume that two heap parts h_{1I} and h_{1S} are $n-1, m'$ related at $\xi(W_1(\iota))(W_1)$. By non-expansiveness of ξ it follows that $\xi(W_1(\iota))$ is n, m

equivalent to $\xi(W_2(\iota))$ in c.o.f.e. $\blacktriangleright\widehat{\mathbf{Inv}}$. Since $(n-1, m'+1) < (n, m)$ it follows that they are also $n-1, m'+1$ equivalent in c.o.f.e. $\widehat{\mathbf{Inv}}$. Hence, by the equivalence on uniform predicates, $\xi(W_1(\iota))(W_1)$ and $\xi(W_2(\iota))(W_2)$ agree on all step-indices strictly below $n-1, m'+1$. It follows that h_{1I} and h_{1S} are $n-1, m$ related at $\xi(W_2(\iota))(W_2)$. □

Expression closure. The expression closure, $\mathcal{E}(\nu)$, takes a semantic type $\nu \in \mathbf{Type}$ and extends it to a relation on expressions. Intuitively, two expressions e_I and e_S are related by the expression closure $\mathcal{E}(\nu)$ if they reduce to ν-related values and related terminal heaps, whenever they are executed from related initial heaps. Since the expression closure is where we actually reduce the underlying expressions, this is also the definition where the steps from the step-indexed model are tied to concrete reduction steps. The idea is to tie the first step-index to concrete reduction steps and go down one step in the first step-index for every concrete reduction step of e_I. We define the expression closure, $\mathcal{E}(\nu)$, as follows, for a semantic type $\nu \in \mathbf{Type}$:

$$\begin{aligned}
\mathcal{E}(\nu) \overset{\mathit{def}}{=} \lambda W.\ \{(n, e_I, e_S) \mid{} & \forall n' \leq n.\ \forall i < n'.\ \forall h_I, h'_I, h_S.\ \forall e'_I.\ \forall W' \geq W. \\
& (n', h_I, h_S) \in \lfloor W' \rfloor \wedge \langle e_I, h_I \rangle \to^i \langle e'_I, h'_I \rangle \not\to \quad \Rightarrow \\
& e'_I \in \mathrm{Val} \wedge \exists v'_S, h'_S.\ \exists W'' \geq W'. \\
& \quad \langle e_S, h_S \rangle \to^* \langle v'_S, h'_S \rangle \wedge \\
& \quad (n' - i, h'_I, h'_S) \in \lfloor W'' \rfloor \wedge \\
& \quad \forall m.\ (n' - i, m, e'_I, v'_S) \in \nu(W'')\}
\end{aligned}$$

The expression closure is only step-indexed using the first step-index, corresponding to the number of concrete reduction steps. If the two initial heaps are related for n' steps and $\langle e_I, h_I \rangle$ reduces in $i < n'$ steps then the terminal heaps must be related for $n' - i$ steps. Like the definition of heap satisfaction, we require the return values to be related for $n' - i$ concrete steps and an arbitrary number of logical steps, m. The reason for requiring i to be strictly smaller than n', is to ensure that we have enough concrete steps left to prove that we do not get stuck trying to dereference a location that has not been allocated. This is due to our use of general invariants, which requires that we unfold a reference invariant to prove that two τ ref related locations are currently allocated.

To ensure that the expressions remain related after more invariants have been allocated, we take initial heaps related in an arbitrary future world W'. Likewise, the terminal heaps and return value is related in some future world W'', to allow the allocation of new invariants.

The expression closure further requires that whenever $\langle e_I, h_I \rangle$ reduces to some irreducible configuration $\langle e'_I, h'_I \rangle$, then e'_I is in fact a value. This allows us to prove that well-typed expressions do not get stuck.

Just like for heap satisfaction, we need the expression closure to satisfy the following non-expansiveness property (Lemma 8), to ensure that definitions using the expression closure are suitably non-expansive.

Lemma 8

$$\forall W_1, W_2 \in \mathbf{World}.\ \forall \nu \in \mathbf{Type}.\ \forall n, m \in \mathbb{N}.\ \forall e_I, e_S \in \mathrm{Exp}.$$
$$W_1 \overset{n,m}{=}_{\mathbf{World}} W_2 \wedge (n, e_I, e_S) \in \mathcal{E}(\nu)(W_1) \Rightarrow (n, e_I, e_S) \in \mathcal{E}(\nu)(W_2)$$

Proof (Sketch). The result follows easily from non-expansiveness of ν, Lemma 7 and the following property that allows an n, m equivalence to be extended to a future world.

$$\forall W_1, W_2, W_1' \in \mathbf{World}.\ \forall n, m.$$
$$W_1 \overset{n,m}{=}_{\mathbf{World}} W_2 \wedge W_1 \leq W_1' \Rightarrow \exists W_2' \geq W_2.\ W_1' \overset{n,m}{=}_{\mathbf{World}} W_2'$$

□

Interpretation. We now have all the ingredients to define the relational interpretation of types. The relational interpretation of types, $\mathcal{V}[\![\Delta \vdash \tau]\!] : \mathbf{Type}^{\Delta} \to \mathbf{Type}$, is defined by induction on the type well-formedness derivation, $\Delta \vdash \tau$. It is parametrized by a function $\rho \in \mathbf{Type}^{\Delta}$ that assigns a semantic type to each type variable in context Δ. Intuitively, it expresses when two values are related at a given type at a particular world and step-index. The full definition is given in Figure 4. The most interesting clauses are the interpretation of function types and reference types.

For function types, we assume that the arguments are related for an arbitrary number of logical steps m', just as we did for heap satisfaction. We also require the application to be related for one more concrete reduction step than the arguments. This is standard in step-indexed models and reflects the fact that we do at least one concrete reduction step (the beta-reduction of the application), before using the arguments.

The interpretation of reference types asserts the existence of a general invariant in the current world that is n, m equivalent to the reference invariant $\mathrm{inv}(\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho, l_I, l_S)$. Note that the equivalence is stated at the c.o.f.e., $\blacktriangleright \widehat{\mathbf{Inv}}$.

In turn, the reference invariant asserts exclusive ownership of locations l_I and l_S and that these locations contain $\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho$ related values.

For product types, we require that the components are pair-wise related and for existential types, we require the existence of a semantic type to interpret the abstract type.

To show that this logical relation is well-defined, we need to prove that the reference invariant, $\mathrm{inv}(\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho, l_I, l_S)$ is an element of the right c.o.f.e., $\mathbf{World} \overset{mon}{\to} \mathbf{UPred}(\mathrm{Heap} \times \mathrm{Heap})$. This reduces to proving that the reference invariant is non-expansive and monotone in the worlds, which in turn induces non-expansiveness and monotonicity requirements on $\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho$.

Lemma 9 *The logical relation is well-defined. In particular,*

- $\mathrm{inv}(\nu, l_I, l_S)$ *is non-expansive and monotone and* $\mathrm{inv}(\nu, l_I, l_S)(W)$ *is downwards-closed for all* $\nu \in \mathbf{Type}$, $l_I, l_S \in \mathrm{Loc}$ *and* $W \in \mathbf{World}$.

$$\begin{aligned}
\mathcal{V}[\![\Delta \vdash 1]\!]_\rho(W) &\overset{def}{=} \{(n, m, *, *)\}\\
\mathcal{V}[\![\Delta \vdash \mathbb{N}]\!]_\rho(W) &\overset{def}{=} \{(n, m, \underline{k}, \underline{k}) \mid k \in \mathbb{N}\}\\
\mathcal{V}[\![\Delta \vdash \tau \times \sigma]\!]_\rho(W) &\overset{def}{=} \{(n, m, v_I, v_S) \mid \exists v_{1I}, v_{2I}, v_{1S}, v_{2S}.\\
&\qquad v_I = (v_{1I}, v_{2I}) \wedge v_S = (v_{1S}, v_{2S}) \wedge\\
&\qquad (n, m, v_{1I}, v_{1S}) \in \mathcal{V}[\![\Delta \vdash \tau]\!]_\rho(W) \wedge\\
&\qquad (n, m, v_{2I}, v_{2S}) \in \mathcal{V}[\![\Delta \vdash \sigma]\!]_\rho(W)\}\\
\mathcal{V}[\![\Delta \vdash \tau \to \sigma]\!]_\rho(W) &\overset{def}{=} \{(n, m, v_I, v_S) \mid \forall n' < n.\ \forall W' \geq W.\ \forall u_I, u_S.\\
&\qquad (\forall m'.\ (n', m', u_I, u_S) \in \mathcal{V}[\![\Delta \vdash \tau]\!]_\rho(W'))\\
&\qquad \Rightarrow\ (n' + 1, v_I\ u_I, v_S\ u_S) \in \mathcal{E}(\mathcal{V}[\![\Delta \vdash \sigma]\!]_\rho)(W')\}\\
\mathcal{V}[\![\Delta \vdash \tau\ \mathsf{ref}\,]\!]_\rho(W) &\overset{def}{=} \{(n, m, l_I, l_S) \mid \exists \iota \in \mathrm{dom}(W).\\
&\qquad \xi(W(\iota)) \overset{n,m}{=}_{\blacktriangleright \widehat{\mathbf{Inv}}} \mathrm{inv}(\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho, l_I, l_S)\}\\
\mathcal{V}[\![\Delta, \alpha \vdash \alpha]\!]_\rho(W) &\overset{def}{=} \rho(\alpha)(W)\\
\mathcal{V}[\![\Delta \vdash \exists \alpha.\ \tau]\!]_\rho(W) &\overset{def}{=} \{(n, m, \mathsf{pack}\ v_I, \mathsf{pack}\ v_S) \mid \exists \nu \in \mathbf{Type}.\\
&\qquad (n, m, v_I, v_S) \in \mathcal{V}[\![\Delta, \alpha \vdash \tau]\!]_{\rho[\alpha \mapsto \nu]}(W)\}
\end{aligned}$$

Fig. 4. Relational interpretation of types.

– *$\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho$ is non-expansive and monotone and $\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho(W)$ is downwards-closed for all $\rho \in \mathbf{Type}^{\Delta}$ and $W \in \mathbf{World}$.*

Proof (Sketch). We prove the second property by induction on the well-formedness derivation, $\Delta \vdash \tau$. Most of the cases are straightforward. For reference types, non-expansiveness follows from Lemma 8. □

Finally, the logical relation relates two expressions e_I and e_S at type τ if they are related for all step-indices in the expression closure of the τ-relation, after substituting related values for all free term variables.

Definition 11 (Logical relation)

$$\begin{aligned}
\Delta; \Gamma \models e_I \leq_{log} e_S : \tau \overset{def}{=} &\forall n \in \mathbb{N}.\ \forall W \in \mathbf{World}.\ \forall \sigma_I, \sigma_S \in \mathrm{Val}^{\Gamma}.\ \forall \rho \in \mathbf{Type}^{\Delta}.\\
&(\forall m \in \mathbb{N}.\ \forall (x : \tau) \in \Gamma.\ (n, m, \sigma_I(x), \sigma_S(x)) \in \mathcal{V}[\![\Delta \vdash \tau]\!]_\rho(W))\\
&\Rightarrow\ (n, \sigma_I(e_I), \sigma_S(e_S)) \in \mathcal{E}(\mathcal{V}[\![\Delta \vdash \tau]\!]_\rho)(W)
\end{aligned}$$

The logical relation is compatible with the typing rules of the language. Here we include two of the interesting cases of the compatibility proof, namely dereference and unpack. For illustrative purposes we take a more constrained versions of the lemmas where the arguments of the dereferencing and unpacking operations are values. The accompanying technical report features complete proofs of all the compatibility lemmas in their general form.

Lemma 10 *If $\Delta;\Gamma \models v_I \leq_{log} v_S : \tau\ \mathsf{ref}$ then $\Delta;\Gamma \models\ !\,v_I \leq_{log}\ !\,v_S : \tau$.*

Proof. We unfold the definition of the logical relation: take $n \in \mathbb{N}$, $W \in \mathbf{World}$, $\sigma_I, \sigma_S \in \mathrm{Val}^{\Gamma}$ and $\rho \in \mathbf{Type}^{\Delta}$ such that

$$\forall m \in \mathbb{N}.\ \forall (x : \tau') \in \Gamma.\ (n, m, \sigma_I(x), \sigma_S(x)) \in \mathcal{V}[\![\Delta \vdash \tau']\!]_{\rho}(W)$$

We unfold the definition of expression closure: take $n' \leq n$, $i < n'$, $W' \geq W$ and $h_I, h_I', h_S \in \mathrm{Heap}$ such that

$$(n', h_I, h_S) \in \lfloor W' \rfloor \qquad\qquad \langle !\,\sigma_I(v_I), h_I\rangle \to^i \langle e_I', h_I'\rangle \not\to$$

Since $\sigma_I(v_I)$ and $\sigma_S(v_S)$ are both values, we can use the assumption to obtain a world $W'' \geq W'$ such that $(n', h_I, h_S) \in \lfloor W'' \rfloor$ and

$$\forall m.\ (n', m, \sigma_I(v_I), \sigma_S(v_S)) \in \mathcal{V}[\![\Delta \vdash \tau\ \mathsf{ref}\,]\!]_{\rho}(W'').$$

By definition of the interpretation of reference types, our values must be locations, and they must be related by the reference invariant in the world. In other words, setting $m = 0$ we get two locations l_I and l_S and an invariant identifier $\iota \in \mathrm{dom}(W'')$ such that

$$\sigma_I(v_I) = l_I \qquad \sigma_S(v_S) = l_S \qquad \xi(W''(\iota)) \overset{n',0}{=}_{\blacktriangleright \widehat{\mathbf{Inv}}} \mathrm{inv}(\mathcal{V}[\![\Delta \vdash \tau]\!]_{\rho}, l_I, l_S)$$

Using Lemma 5 we can obtain parts of h_I and h_S that satisfy the invariant ι: we have $h_I^{\iota} \subseteq h_I$ and $h_S^{\iota} \subseteq h_S$ such that $(n', h_I^{\iota}, h_S^{\iota}) \in \lfloor W'' \rfloor_{\{\iota\}}$. Since $0 \leq i < n'$, we can now use Lemma 6, which gives us that,

$$l_I \in \mathrm{dom}(h_I^{\iota}) \quad l_S \in \mathrm{dom}(h_S^{\iota}) \quad \forall m.\ (n'-1, m, h_I^{\iota}(l_I), h_S^{\iota}(l_S)) \in \mathcal{V}[\![\Delta \vdash \tau]\!]_{\rho}(W'')$$

Now we can establish, by definition of the operational semantics, that $i = 1$, $e_I' = h_I(l_I) = h_I^{\iota}(l_I)$ and $h_I' = h_I$. We are now ready to pick witnesses required by the expression closure. We pick $v_S = h_S(l_S)$, $h_S' = h_S$ and use the same world, W''. Clearly, $\langle !l_S, h_S\rangle \to^* \langle h_S(l_S), h_S\rangle$ holds trivially and since the heaps did not change, we get the heap-satisfaction obligation by downwards-closure. Finally, we need to show that $(n'-1, m, h_I(l_I), h_S(l_S)) \in \mathcal{V}[\![\Delta \vdash \tau]\!]_{\rho}(W'')$ for any m, which is precisely what we obtained from Lemma 6, since $h_I(l_I) = h_I^{\iota}(l_I)$ and $h_S(l_S) = h_S^{\iota}(l_S)$. □

Lemma 11 *If $\Delta;\Gamma \models v_I \leq_{log} v_S : \exists\alpha.\ \tau$ and $\Delta, \alpha;\Gamma, x : \tau \models e_I \leq_{log} e_S : \sigma$ and $\Delta \vdash \sigma$ then*

$$\Delta;\Gamma \models \mathsf{unpack}\ v_I\ \mathsf{as}\ x\ \mathsf{in}\ e_I \leq_{log} \mathsf{unpack}\ v_S\ \mathsf{as}\ x\ \mathsf{in}\ e_S : \sigma$$

Proof. We unfold the definition of the logical relation: take $n \in \mathbb{N}$, $W \in \mathbf{World}$, $\sigma_I, \sigma_S \in \mathrm{Val}^{\Gamma}$ and $\rho \in \mathbf{Type}^{\Delta}$ such that

$$\forall m \in \mathbb{N}.\ \forall (x : \tau') \in \Gamma.\ (n, m, \sigma_I(x), \sigma_S(x)) \in \mathcal{V}[\![\Delta \vdash \tau']\!]_{\rho}(W).$$

We unfold the definition of expression closure: take $n' \leq n$, $i < n'$, $W' \geq W$ and $h_I, h'_I, h_S \in \text{Heap}$ such that

$$(n', h_I, h_S) \in \lfloor W' \rfloor \qquad \langle \mathsf{unpack}\ \sigma_I(v_I)\ \mathsf{as}\ x\ \mathsf{in}\ \sigma_I(e_I), h_I \rangle \to^i \langle e'_I, h'_I \rangle \not\to$$

Since $\sigma_I(v_I)$ and $\sigma_S(v_S)$ are both values, we can use the first assumption to obtain a world $W'' \geq W'$ such that $(n', h_I, h_S) \in \lfloor W'' \rfloor$ and

$$\forall m.\ (n', m, \sigma_I(v_I), \sigma_S(v_S)) \in \mathcal{V}[\![\Delta \vdash \exists \alpha.\ \tau]\!]_\rho(W'').$$

We pick $m = 0$, and obtain, by definition of interpretation of existential types, two values, v'_I and v'_S, and $\nu \in \mathbf{Type}$ such that

$$\sigma_I(v_I) = \mathsf{pack}\ v'_I \quad \sigma_S(v_S) = \mathsf{pack}\ v'_S \quad (n', 0, v'_I, v'_S) \in \mathcal{V}[\![\Delta, \alpha \vdash \tau]\!]_{\rho[\alpha \mapsto \nu]}(W'')$$

Looking at our reduction, we now observe that $i > 0$ and

$$\langle \mathsf{unpack}\ \sigma_I(v_I)\ \mathsf{as}\ x\ \mathsf{in}\ \sigma_I(e_I), h_I \rangle \to \langle (\sigma_I(e_I))[v'_I/x], h_I \rangle \to^{i-1} \langle e'_I, h'_I \rangle \not\to .$$

Note that $(\sigma_I(e_I))[v'_I/x] = (\sigma_I[x \mapsto v'_I])(e_I)$.

It is now time to turn to the second of our assumptions. We instantiate the definition of the logical relation with $n' - 1$, W'', $\sigma_I[x \mapsto v'_I]$, $\sigma_S[x \mapsto v'_S]$, and $\rho[\alpha \mapsto \nu]$. We need to show that the substitutions are related, i.e., that

$$(n' - 1, m, \sigma_I[x \mapsto v'_I](y), \sigma_S[x \mapsto v'_S](y)) \in \mathcal{V}[\![\Delta, \alpha \vdash \tau']\!]_{\rho[\alpha \mapsto \nu]}(W'')$$

for any $m \in \mathbb{N}$ and any $(y : \tau') \in \Gamma, x : \tau$. For all the variables other than x this holds by weakening, world monotonicity and downwards-closure; for x we obtained the necessary property relating v'_I to v'_S from the first assumption. Note that picking $n' - 1$ as the index was crucial, since we only got this final relation for 0 logical steps.[2] However, since we count unpack-pack reductions, we can safely use $n' - 1$ here.

By showing the substitutions to be related, we have learned that the expressions e_I and e_S are also related, after applying the substitutions:

$$(n' - 1, \sigma_I(e_I)[v'_I/x], \sigma_S(e_S)[v'_S/x]) \in \mathcal{E}(\mathcal{V}[\![\Delta, \alpha \vdash \tau]\!]_{\rho[\alpha \mapsto \nu]}(W'')).$$

If we unfold the definition of the expression closure, we notice that if we instantiate it with $n' - 1$, $i - 1$, h_I, h'_I, h_S and W'' we can use it to complete the proof, since $\langle \mathsf{unpack}\ \sigma_S(v_S)\ \mathsf{as}\ x\ \mathsf{in}\ \sigma_S(e_S), h_S \rangle \to \langle \sigma_S(e_S)[v'_S/x], h_S \rangle$. Thus, it suffices to show that

$$\langle (\sigma_I(e_I))[v'_I/x], h_I \rangle \to^{i-1} \langle e'_I, h'_I \rangle \not\to \qquad (n' - 1, h_I, h_S) \in \lfloor W'' \rfloor$$

However, the first of these properties we already obtained, and the second holds by downwards-closure of erasure. □

[2] This is caused by the order of quantifiers in the definition: we needed to pick a number of logical steps *before* we obtained the witnesses from the definition, including v'_I, v'_S and ν.

The fundamental theorem of logical relations (Lemma 12) and a similar property for contexts (Lemma 13) follow as corollaries of the compatibility lemmas. The proofs follow by induction on the respective typing derivations.

Lemma 12 *If $\Delta; \Gamma \vdash e : \tau$ then $\Delta; \Gamma \models e \leq_{log} e : \tau$.*

Lemma 13 *For any context C such that $C : (\Delta_i; \Gamma_i, \tau_i) \rightsquigarrow (\Delta_o; \Gamma_o, \tau_o)$, if $\Delta_i; \Gamma_i \vdash e_I \leq_{log} e_S : \tau_i$ then $\Delta_o; \Gamma_o \vdash C[e_I] \leq_{log} C[e_S] : \tau_o$.*

Theorem 2 (Soundness) *If $\Delta; \Gamma \models e_I \leq_{log} e_S : \tau$ then $\Delta; \Gamma \vdash e_I \leq_{ctx} e_S : \tau$.*

Proof. Let C be an arbitrary context such that $C : (\Delta; \Gamma, \tau) \rightsquigarrow (-; -, 1)$ and $\langle C[e_I], []\rangle \to^* \langle *, h_I\rangle$. Then there exists an i such that $\langle C[e_I], []\rangle \to^i \langle *, h_I\rangle$. By Lemma 13 it follows that $-; - \models C[e_I] \leq_{log} C[e_S] : 1$ and thus $(i + 1, C[e_I], C[e_S]) \in \mathcal{E}(\mathcal{V}[\![-; - \vdash 1]\!])([])$. By definition of $\mathcal{E}$ this gives us v_S, h_S and W such that $\langle C[e_S], []\rangle \to^* \langle v_S, h_S\rangle$ and $(1, m, *, v_S) \in \mathcal{V}[\![-; - \vdash 1]\!](W)$ for any $m \in \mathbb{N}$. From the latter we obtain $v_S = *$, which ends the proof. □

5 Example

Recall the example from the Introduction where f takes as an argument an ADT implementation of type $\exists\alpha.\ (1 \to \alpha) \times (\alpha \to \mathbb{N})$ and returns a new ADT implementation of the same type:

$$\lambda x.\ \mathsf{unpack}\ x\ \mathsf{as}\ y\ \mathsf{in}\ \mathsf{pack}\ (\lambda z.\ \mathsf{ref}\ (\pi_1(y)(z)), \lambda z.\ \pi_2(y)(!z))$$

Under the hood, f wraps instances of the new ADT implementation in an additional reference. However, since this is transparent to clients, one would expect that $f(x)$ is contextually equivalent to x:

$$-; x : \tau \vdash x \cong_{ctx} f(x) : \tau$$

where τ is the type $\exists\alpha.\ (1 \to \alpha) \times (\alpha \to \mathbb{N})$.

The left-to-right approximation, $-; x : \tau \vdash x \leq_{ctx} f(x) : \tau$, causes problems for previous step-indexed logical relations, as it requires an additional logical step without introducing a corresponding reduction step on the left. In this Section we prove the left-to-right approximation using our logical relation.

Conceptually, to prove the left-to-right approximation, we need to relate values of type α to values of type $\alpha\ \mathsf{ref}$, the type constructed by the right-hand-side package, given a relation ν that relates pairs of values of type α. The obvious way to proceed is to introduce an invariant that relates the value on the left, with the value stored at the given location on the right:

$$S(v_I, l_S)(W) \overset{def}{=} \{(n, m, h_I, h_S) \mid l_S \in \mathrm{dom}(h_S) \wedge (n, m, v_I, h_S(l_S)) \in \nu(W)\} \quad (2)$$

This definition forms a valid *invariant* for any v_I and l_S, i.e., $S(v_I, l_S) \in \widehat{\mathbf{Inv}}$. Since the objects we want to relate are existential packages, we need to pick an interpretation $\nu' \in \mathbf{Type}$ for the existential type *before* any locations for the right-hand-side are actually allocated. This is why we use a Kripke logical relation: we can use the world, in much the same way as we did for the reference type in Figure 4, and take

$$\nu'(W) \overset{\mathit{def}}{=} \{(n, m, v_I, v_S) \mid \exists \iota \in \mathrm{dom}(W).\ \xi(W(\iota)) \overset{n,m}{=}_{\blacktriangleright \widehat{\mathbf{Inv}}} S(v_I, v_S)\} \qquad (3)$$

Thus far, we made no particular use of our transfinite step-indexing: these definitions could well be stated in a standard setup. The power of our approach comes from the following property.

Lemma 14 *For any* $k > 0$*, any pair of values* $v_I, v_S \in \mathrm{Val}$ *such that* $(k, m + 2, v_I, v_S) \in \nu'(W)$ *and any heaps* $h_I, h_S \in \mathrm{Heap}$ *such that* $(k + 1, h_I, h_S) \in \lfloor W \rfloor$*,*

$$(k, m, v_I, h_S(v_S)) \in \nu(W).$$

The higher logical step-index in the assumption is required since, as we shall see, we need to unfold the recursive domain equation to prove it. Once we have the lemma, if we need to show that $(k, m, v_I, h_S(v_S)) \in \nu(W)$ for an *arbitrary* m, as the expression closure obliges us to do, and know that $\forall m.\ (k, m, v_I, v_S) \in \nu'(W)$, we can simply instantiate our assumption with $m + 2$ and use it. In this way, the fact that our model allows us to take arbitrary number of logical steps is crucial to allow us to prove this type of abstractions correct. We first prove the preceding lemma, and then turn to formally treating the example.

Proof (Lemma 14). By (3), we know there is an invariant identifier $\iota \in \mathrm{dom}(W)$ such that $\xi(W(\iota)) \overset{k,m+2}{=}_{\blacktriangleright \widehat{\mathbf{Inv}}} S(v_I, v_S)$. This last property is, by definition of $\blacktriangleright$, equivalent to

$$\xi(W(\iota)) \overset{k,m+1}{=}_{\widehat{\mathbf{Inv}}} S(v_I, v_S).$$

Furthermore, since we know that ι defines an invariant in the world W, we can use the heap satisfaction assumption, together with Lemma 5 to obtain subheaps $h_I^\iota \subseteq h_I$ and $h_S^\iota \subseteq h_S$ such that $(k, n, h_I^\iota, h_S^\iota) \in \xi(W(\iota))(W)$ for any n. Taking $n = m$, we thus get $(k, m, h_I^\iota, h_S^\iota) \in \xi(W(\iota))(W)$, and ultimately, from the $(k, m + 1)$-equality of the invariants,

$$(k, m, h_I^\iota, h_S^\iota) \in S(v_I, v_S)(W).$$

Unfolding the definition of S, (2), we find that $(k, m, v_I, h_S^\iota(v_S)) \in \nu(W)$, which ends the proof. □

Lemma 15

$$-; x : \tau \vdash x \leq_{ctx} f(x) : \tau$$

Proof. By Theorem 2 it suffices to prove that x is logically related to $f(x)$: $-; x : \tau \models x \leq_{log} f(x) : \tau$.

Let $n \in \mathbb{N}$, $W \in \mathbf{World}$ and $\sigma_I, \sigma_S \in \mathrm{Val}^{x:\tau}$, such that

$$\forall m.\ (n, m, \sigma_I(x), \sigma_S(x)) \in \mathcal{V}[\![- \vdash \tau]\!]_{[]}(W)$$

From the interpretation of existential and product types it follows that there exists a $\nu \in \mathbf{Type}$ and $v_{1I}, v_{2I}, v_{1S}, v_{2S} \in \mathrm{Val}$ such that $\sigma_I(x) = (v_{1I}, v_{2I})$, $\sigma_S(x) = (v_{1S}, v_{2S})$ and

$$(n, 0, v_{1I}, v_{1S}) \in \mathcal{V}[\![\alpha \vdash 1 \to \alpha]\!]_{[\alpha \mapsto \nu]}(W) \tag{4}$$

$$(n, 0, v_{2I}, v_{2S}) \in \mathcal{V}[\![\alpha \vdash \alpha \to \mathbb{N}]\!]_{[\alpha \mapsto \nu]}(W) \tag{5}$$

Since $\sigma_I(x)$ is already a value and $f(\sigma_S(x))$ pure-reduces to the following value

$$v_s \overset{def}{=} \mathsf{pack}\ (\lambda z.\ \mathsf{ref}\ (\pi_1(v_{1S}, v_{2S})(z)), \lambda z.\ \pi_2(v_{1S}, v_{2S})(!z))$$

it suffices to prove that $\sigma_I(x)$ and v_S are related by the τ value relation:

$$\forall n' \leq n.\ \forall m \in \mathbb{N}.\ \forall W' \geq W.\ (n', m, \sigma_I(x), v_S) \in \mathcal{V}[\![- \vdash \tau]\!]_{[]}(W')$$

We thus have to pick a relational interpretation of the new abstract type. As discussed before, we relate the implementation value v_I with the specification location l_S through the type ν', defined in (3). Recall, the definition states that there exists an invariant S (2) that owns the specification location l_S and asserts that this location contains a value v_S on the specification side, such that v_I and v_S are ν-related. Next, we have to prove that the components of the pairs are related:

$$\begin{aligned}
&\forall n' \leq n.\ \forall m \in \mathbb{N}.\ \forall W' \geq W.\\
&\quad (n', m, v_{1I}, \lambda z.\ \mathsf{ref}\ (\pi_1(v_{1S}, v_{2S})(z))) \in \mathcal{V}[\![\alpha \vdash 1 \to \alpha]\!]_{[\alpha \mapsto \nu']}(W') \ \wedge\\
&\quad (n', m, v_{2I}, \lambda z.\ \pi_2(v_{1S}, v_{2S})(!z))) \in \mathcal{V}[\![\alpha \vdash \alpha \to \mathbb{N}]\!]_{[\alpha \mapsto \nu']}(W')
\end{aligned}$$

We will focus on proving the second conjunct, as this is the one that requires an unfolding of the impredicative invariant without a corresponding reduction step. From the interpretation of function types, we have to prove the functions take ν'-related arguments to $\mathbb{N}$-related expressions:

$$\begin{aligned}
&\forall n' \leq n.\ \forall W' \geq W.\ \forall n'' < n'.\ \forall u_I, u_S.\\
&\quad (\forall m \in \mathbb{N}.\ (n'', m, u_I, u_S) \in \nu'(W'))\\
&\qquad \Rightarrow\ (n'' + 1, v_{2I}\ u_I, (\lambda z.\ \pi_2(v_{1S}, v_{2S})(!z))\ u_S) \in \mathcal{E}(\mathcal{V}[\![\alpha \vdash \mathbb{N}]\!]_{[\alpha \mapsto \nu']})(W')
\end{aligned}$$

Let $n', n'' \in \mathbb{N}$, $W' \in \mathbf{World}$ and $u_I, u_S \in \mathrm{Val}$ such that

$$n'' < n' \leq n \qquad W' \geq W \qquad \forall m \in \mathbb{N}.\ (n'', m, u_I, u_S) \in \nu'(W')$$

To show that

$$(n'' + 1, m, v_{2I}\ u_I, (\lambda z.\ \pi_2(v_{1S}, v_{2S})(!z))\ u_S) \in \mathcal{E}(\mathcal{V}[\![\alpha \vdash \mathbb{N}]\!]_{[\alpha \mapsto \nu']})(W')$$

let $k, i \in \mathbb{N}$, $h_I, h_S, h'_I \in$ Heap, $W'' \in$ **World** and $e'_I \in$ Exp such that

$$i < k \leq n'' + 1 \quad W'' \geq W' \quad (k, h_I, h_S) \in \lfloor W'' \rfloor \quad \langle v_{2I}\ u_I, h_I \rangle \to^i \langle e'_I, h'_I \rangle \not\to$$

To apply our earlier hypothesis (5), $(n, 0, v_{2I}, v_{2S}) \in \mathcal{V}[\![\alpha \vdash \alpha \to \mathbb{N}]\!]_{[\alpha \mapsto \nu]}(W)$, we need to prove that u_I and $h_S(u_S)$ are $(k-1, m)$-related for an arbitrary number of logical steps m. We now revert to the earlier discussion, and use Lemma 14, which requires us to instantiate our earlier assumption about ν' relatedness of u_I and u_S with $m+2$ logical steps (since we need two logical steps to unfold the invariant).

To show that $\forall m.\ (k-1, m, u_I, h_S(u_S)) \in \nu(W'')$, take any $m \in \mathbb{N}$. Since $k - 1 \leq n''$ and $W'' \geq W'$ we have $(k-1, m+2, u_I, u_S) \in \nu'(W')$, and by Lemma 14 the property holds.

Hence, $\forall m.\ (k-1, m, u_I, h_S(u_S)) \in \nu(W'')$. From our original assumption (5) it thus follows that $v_{2I}\ u_I$ and $v_{2S}\ (h_S(u_S))$ are k-related in the expression closure:

$$(k, v_{2I}\ u_I, v_{2S}\ h_S(u_S)) \in \mathcal{E}(\mathcal{V}[\![\alpha \vdash \mathbb{N}]\!]_{[\alpha \mapsto \nu]})(W'')$$

Since $i < k \leq k$ it thus follows that there exists h'_S, v'_S and W''' such that

$$W''' \geq W'' \qquad \langle v_{2S}\ h_S(u_S), h_S \rangle \to^* \langle v'_S, h'_S \rangle \qquad (k-i, h'_I, h'_S) \in \lfloor W''' \rfloor$$

and $\forall m.\ (k-i, m, e'_I, v'_S) \in \mathcal{V}[\![\alpha \vdash \mathbb{N}]\!]_{[\alpha \mapsto \nu]}(W''')$, as required. □

6 Related work

Step-indexing was invented by Appel and McAllester as a way of proving type safety for a language with recursive types using only simple mathematics suitable for foundational proof-carrying code [5]. Ahmed and co-workers developed the technique to support relational reasoning and more advanced languages featuring general references and impredicative polymorphism [3, 1, 2]. These relational models have subsequently been refined with ever more powerful forms of invariants and applied to reason about local state and control effects [15], compiler correctness [17, 7], type-based program transformations [11] and fine-grained concurrent data structures [22], among other things. In another concurrent line of work, step-indexing has been applied to define models of increasingly expressive capability type systems [10] and concurrent program logics [4, 21, 20, 18]. A common thread in both of these lines of work, is increasingly powerful recursively-defined worlds to capture various forms of invariants. We believe the complexity of invariants and associated recursively-defined worlds is orthogonal to the problem of the tight coupling between concrete and logical steps. In particular, our fixed point construction for locally contractive functors (Theorem 1) allows us to define recursively-defined worlds of the same form as prior approaches.

Usually, step-indexed logical relations are indexed over ω, because the closure ordinal of the inductively-defined convergence (termination) predicate is ω. An exception is the logical relation in [8] for reasoning about must-equivalence for a pure functional language without references but with countable non-determinism.

Because of the presence of countable non-determinism, the closure ordinal of the inductively defined must-convergence predicate in *loc. cit.* is not ω. However, it is bounded by ω_1, and therefore the logical relation is indexed over ω_1. Note that this use of transfinite indexing is quite different from our use of transfinite indexing in this paper. Here we use transfinite indexing even though the closure ordinal of the convergence predicate is ω, because we seek to avoid linking unfoldings of the recursive domain equation to concrete reduction steps in the operational semantics.

Various ways have been proposed for stratifying the construction of semantic domains using step-indexing. These include explicit step-indexing [3, 6], Hobor et al.'s indirection theory [16], Birkedal et al.'s ultrametric approach [12] and Birkedal et al.'s guarded recursion approach [9]. Indirection theory and the ultrametric approach are both currently specific to step-indexing over ω. Birkedal et al. have shown that sheaves over any complete Heyting algebra with a well-founded base models guarded dependent type theory [9]. It would be interesting to explore the possibility of using the internal language of sheaves over $\omega^2 + 1$ to define and reason about models step-indexed over ω^2.

Di Gianantonio and Miculan [13] introduced complete ordered families of equivalences as a unifying theory for mixed-variance recursive definitions that supports transfinite fixed point constructions. They define complete ordered families of equivalences over an arbitrary well-founded order and prove a generalized fixed point theorem for contractive endofunctions over these complete ordered families of equivalences. This allows for mixed-variance recursive definitions of elements of c.o.f.e.s. We extend their theory with an explicit construction for mixed-variance recursive definitions of c.o.f.e.s, for the specific instance of c.o.f.e.s over ω^2.

7 Conclusion and Future Work

Step-indexing has proven to be a very powerful technique for modelling advanced type systems and expressive program logics. Impredicative invariants are crucial for achieving modular reasoning and they show up in many different forms in recent models of type systems and program logics. Step-indexing allows us to model impredicative invariants by stratifying the model construction using steps. Unfortunately, current step-indexed models relate these steps directly to concrete reduction steps in the underlying operational semantics and require a concrete reduction step for each logical step. This is especially problematic for higher-order abstractions that introduce new logical steps, without any corresponding reduction steps. This is a common occurrence in higher-order program logics when deriving new specifications for specific use-cases of libraries from an abstract library specification.

In this paper we have isolated the problem in the setting of logical relations and demonstrated a solution based on transfinite step-indexing. This setting is sufficiently simple that we can present all the necessary details to see how transfinite step-indexing solves the problem. To do so we have developed a general theory for solving recursive domain equations using step-indexing over ω^2. Since our theory allows us to solve the equations used in the step-indexed models of cutting-edge program logics, we believe our solution will also scale to these systems, which can hopefully lead to development of more robust reasoning principles.

A natural question to ask is whether step-indexing over ω^2 suffices or whether we might wish to index beyond ω^2. It is not clear whether indexing beyond ω^2 will allow us to prove more equivalences than with our current model. However, it seems plausible that indexing beyond ω^2 could simplify reasoning by allowing less precise counting of logical steps and in the particular case of a step-indexed program logic might help with modularity by allowing the logic to abstract over the precise number of logical steps used. We leave these questions open for future work.

Acknowledgements

This research was supported in part by the ModuRes Sapere Aude Advanced Grant from The Danish Council for Independent Research for the Natural Sciences (FNU) and Danish Council for Independent Research project DFF – 4181-00273.

References

1. A. Ahmed. Step-Indexed Syntactic Logical Relations for Recursive and Quantified Types. In *Proceedings of ESOP*, 2006.
2. A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent Representation Independence. In *Proceedings of POPL*, 2009.
3. A. J. Ahmed. *Semantics of Types for Mutable State.* PhD thesis, Princeton University, 2004.
4. A. W. Appel, R. Dockins, A. Hobor, J. Dodds, X. Leroy, S. Blazy, G. Stewart, and L. Beringer. *Program Logics for Certified Compilers.* Cambridge University Press, 2014.
5. A. W. Appel and D. McAllester. An Indexed Model of Recursive Types for Foundational Proof-carrying Code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.
6. A. W. Appel, P.-A. Melliès, C. D. Richards, and J. Vouillon. A Very Modal Model of a Modern, Major, General Type System. In *Proceedings of POPL*, 2007.
7. N. Benton and C.-K. Hur. Biorthogonality, Step-Indexing and Compiler Correctness. In *Proceedings of ICFP*, 2009.
8. L. Birkedal, A. Bizjak, and J. Schwinghammer. Step-Indexed Relational Reasoning for Countable Nondeterminism. *Logical Methods in Computer Science*, 9(4):1–23, 2013.
9. L. Birkedal, R. E. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8(4), 2012.
10. L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-Indexed Kripke Models over Recursive Worlds. In *Proceedings of POPL*, 2011.

11. L. Birkedal, F. Sieczkowski, and J. Thamsborg. A Concurrent Logical Relation. In *Proceedings of CSL*, 2012.
12. L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations. *Theoretical Computer Science*, 411:4102–4122, 2010.
13. P. Di Gianantonio and M. Miculan. A Unifying Approach to Recursive and Corecursive Definitions. In *Proceedings of TYPES*, 2002.
14. M. Dodds, S. Jagannathan, M. Parkinson, K. Svendsen, and L. Birkedal. Verifying Custom Synchronisation Constructs Using Higher-Order Separation Logic. Accepted for publication in TOPLAS.
15. D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. *Journal of Functional Programming*, 22:477–528, 2012.
16. A. Hobor, R. Dockins, and A. W. Appel. A Theory of Indirection via Approximation. In *Proceedings of POPL*, 2010.
17. C.-K. Hur and D. Dreyer. A Kripke Logical Relation Between ML and Assembly. In *Proceedings of POPL*, 2011.
18. R. Jung, D. Swasey, F. Sieczkowski, K. Svendsen, A. Turon, L. Birkedal, and D. Dreyer. Iris: Monoids and Invariants as an Orthogonal Basis for Concurrent Reasoning. In *Proceedings of POPL*, 2015.
19. J. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proceedings of LICS*, 2002.
20. K. Svendsen and L. Birkedal. Impredicative Concurrent Abstract Predicates. In *Proceedings of ESOP*, 2014.
21. A. Turon, D. Dreyer, and L. Birkedal. Unifying Refinement and Hoare-Style Reasoning in a Logic for Higher-Order Concurrency. In *Proceedings of ICFP*, 2013.
22. A. J. Turon, J. Thamsborg, A. Ahmed, L. Birkedal, and D. Dreyer. Logical Relations for Fine-grained Concurrency. In *Proceedings of POPL*, 2013.