A Step-indexed Kripke Model of Hidden State via Recursive Properties on Recursively Defined Metric Spaces

Lars Birkedal
IT University of Copenhagen
birkedal@itu.dk

Jan Schwinghammer
Saarland University
jan@ps.uni-saarland.de

Kristian Støvring
IT University of Copenhagen
kss@itu.dk

Abstract

Frame and anti-frame rules have been proposed as proof rules for modular reasoning about programs. Frame rules allow one to hide irrelevant parts of the state during verification, whereas the anti-frame rule allows one to hide local state from the context. We give a possible worlds semantics for Charguéraud and Pottier's type and capability system including frame and anti-frame rules, based on the operational semantics and step-indexed heap relations. The worlds are constructed as a recursively defined predicate on a recursively defined metric space, which provides a considerably simpler alternative compared to a previous construction.

1 Introduction

Reasoning about higher-order stateful programs is notoriously difficult, and often involves the need to track aliasing information. A particular line of work that addresses this point are substructural type systems with regions, capabilities and singleton types [1, 7, 8]. In this context, Pottier [9] presented the anti-frame rule as a proof rule for hiding local state. It states that (the description of) a piece of mutable state that is *local* to a procedure can be removed from the procedure's external interface (expressed in the type system). The benefits of hiding local state include simpler interface specifications, simpler reasoning about client code, and fewer restrictions on the procedure's use because of potential aliasing. Thus, in combination with frame rules that allow the irrelevant parts of the state to be hidden during verification, the anti-frame rule provides an important ingredient for modular reasoning about programs.

Soundness of the frame and anti-frame rules is subtle, and relies on properties of the programming language. Pottier [9] sketched a proof for the anti-frame rule by a progress and preservation argument, resting on assumptions about the existence of certain recursively defined types. Subsequently, Schwinghammer et al. [12] investigated the semantic foundations of the anti-frame rule by identifying sufficient conditions for its soundness, and by instantiating their general setup to prove soundness for a separation logic variant of the rule. The latter was done by giving a Kripke model where assertions are indexed over recursively defined worlds. The recursive domain equation involved functions that should be monotone with respect to an order relation that is specified using the isomorphism of the solution itself and an operator on the recursively defined worlds. This means that the standard existence theorems do not appear to apply, and thence we had to give the solution by a laborious explicit inverse-limit construction [12].

Here we explore an alternative approach, which consists of two steps. First, we consider a recursive metric space domain equation without any monotonicity requirement, for which we obtain a solution by appealing to a standard existence theorem. Second, we carve out a suitable subset of what might be called *hereditarily monotonic* functions. We show how to define this recursively specified subset. The resulting subset of monotone functions is, however, not a solution to the original recursive domain equation; hence we verify that the semantic constructions used to justify the anti-frame rule in [12] suitably restrict to the recursively defined subset of hereditarily monotone functions. In summa, this results in a considerably simpler model construction than the earlier one in *loc. cit.*

In the next section we give a brief overview of Charguéraud and Pottier's type and capability system [7, 9] with higher-order frame and anti-frame rules. Section 3 summarizes some background on ultrametric spaces and presents the construction of a set of hereditarily monotonic recursive worlds. Following

recent work by Birkedal et al. [4], we work with step-indexed heap relations based on the operational semantics of the calculus. The worlds thus constructed are then used (Section 4) to give a model of the type and capability system. Due to space limitations, many details have been relegated to an appendix.

2 A Calculus of Capabilities

Syntax and operational semantics. We consider a standard call-by-value, higher-order language with general references, sum and product types, and polymorphic and recursive types. For concreteness, the following grammar gives the syntax of values and expressions, keeping close to the notation of [7, 9]:

$$v ::= x \mid () \mid \mathsf{inj}^i\, v \mid (v_1, v_2) \mid \mathsf{fun}\, f(x){=}t \mid l \qquad t ::= v \mid (v\,t) \mid \mathsf{case}(v_1, v_2, v) \mid \mathsf{proj}^i\, v \mid \mathsf{ref}\, v \mid \mathsf{get}\, v \mid \mathsf{set}\, v$$

Here, the term $\mathsf{fun}\, f(x){=}t$ stands for the recursive procedure f with body t. The operational semantics is given by a relation $(t \mid h) \longmapsto (t' \mid h')$ between configurations that consist of a (closed) expression t and a heap h. We take a heap h to be a finite map from locations l to closed values, we use the notation $h\#h'$ to indicate that two heaps h, h' have disjoint domains, and we write $h \cdot h'$ for the union of two such heaps. By *Val* we denote the set of closed values.

Types. Charguéraud and Pottier's type system uses *capabilities*, *value types*, and *memory types*. A capability C describes a heap property, much like the assertions of a Hoare-style program logic. For instance, $\{\sigma : \mathsf{ref}\,\mathsf{int}\}$ asserts that σ is a valid location that contains an integer value. More complex assertions can be built by separating conjunctions $C_1 * C_2$ and universal and existential quantification over names σ. Value types τ classify values; they include base types, singleton types $[\sigma]$, and are closed under products, sums, and universal quantification. *Memory types* χ, θ describe the result of computations. They extend the value types by a type of references, and also include all types of the form $\exists\vec{\sigma}.\tau * C$ which describe the final value and heap that result from the evaluation of an expression. Arrow types (which are value types) have the form $\chi_1 \to \chi_2$ and thus, like the pre- and post-conditions of a triple in Hoare logic, make explicit which part of the heap is accessed and modified when a procedure is called. We also allow recursive capabilities, value types, and memory types, resp., provided the recursive definition is (formally) contractive, i.e., the recursion must go through a type constructor such as $\to$.

Frame and anti-frame rules. Each of the syntactic categories is equipped with an *invariant extension* operation, $\cdot \otimes C$. Intuitively, this operation conjoins C to the domain and codomain of every arrow type that occurs within its left hand argument, which means that the capability C is preserved by all procedures of this type. This intuition is made precise by regarding capabilities and types modulo a structural equivalence which subsumes the "distribution axioms" for $\otimes$ that are used to express generic higher-order frame rules [6]. Two key cases of the structural equivalence are the distribution axioms for arrow types, $(\chi_1 \to \chi_2) \otimes C = (\chi_1 \otimes C * C) \to (\chi_2 \otimes C * C)$, and for successive extensions, $(\chi \otimes C_1) \otimes C_2 = \chi \otimes (C_1 \circ C_2)$ where the derived operation $C_1 \circ C_2$ abbreviates the conjunction $(C_1 \otimes C_2) * C_2$.

There are two typing judgements, $x_1{:}\tau_1, \ldots, x_n{:}\tau_n \vdash v : \tau$ for values, and $x_1{:}\chi_1, \ldots, x_n{:}\chi_n \Vdash t : \chi$ for expressions. The latter is similar to a Hoare triple where (the separating conjunction of) $\chi_1, \ldots, \chi_n$ serves as a precondition and χ as a postcondition. This view provides some intuition for the following "shallow" and "deep" frame rules, and for the (roughly dual) anti-frame rule:

$$[SF]\frac{\Gamma \Vdash t : \chi}{\Gamma * C \Vdash t : \chi * C} \qquad [DF]\frac{\Gamma \Vdash t : \chi}{(\Gamma \otimes C) * C \Vdash t : (\chi \otimes C) * C} \qquad [AF]\frac{\Gamma \otimes C \Vdash t : (\chi \otimes C) * C}{\Gamma \Vdash t : \chi} \quad (1)$$

As in separation logic, the frame rules can be used to add a capability C (which might assert the existence of an integer reference, say) as an invariant to a specification $\Gamma \Vdash t : \chi$, which is useful for local reasoning.

The difference between the shallow variant $[SF]$ and the deep variant $[DF]$ is that the former adds C only on the top-level, whereas the latter also extends all arrow types nested inside Γ and χ, via $\cdot \otimes C$. While the frame rules can be used to reason about certain forms of information hiding [6], the anti-frame rule expresses a hiding principle more directly: the capability C can be removed from the specification if C is an invariant that is established by t, expressed by $\cdot * C$, and guaranteed to hold whenever control passes from t to the context and back, expressed by $\cdot \otimes C$.

3 Hereditarily Monotonic Recursive Worlds

Intuitively, capabilities describe heaps. A key idea of the model that we present next is that capabilities (as well as types and type contexts) are parameterized by invariants – this will make it easy to interpret the invariant extension operation $\otimes$, as in [11, 12]. But, as the frame and anti-frame rules in (1) indicate, invariants can be arbitrary capabilities again. Thus, we are led to consider a Kripke model where the worlds are *recursively defined*: to a first approximation, we need a solution to the equation

$$W \;=\; W \to \mathit{Pred}(\mathit{Heap})\;. \tag{2}$$

In fact, we will also need to consider a preorder on W and ensure that the interpretation of capabilities and types is *monotonic*. We will find a solution to a suitable variant of (2) using ultrametric spaces.

Ultrametric spaces. We recall some basic definitions and results about ultrametric spaces; for a less condensed introduction to ultrametric spaces we refer to [13]. A *1-bounded ultrametric space* (X,d) is a metric space where the distance function $d : X \times X \to \mathbb{R}$ takes values in the closed interval $[0,1]$ and satisfies the "strong" triangle inequality $d(x,y) \leq \max\{d(x,z),d(z,y)\}$. A metric space is *complete* if every Cauchy sequence has a limit. A function $f : X_1 \to X_2$ between metric spaces (X_1,d_1), (X_2,d_2) is *non-expansive* if $d_2(f(x),f(y)) \leq d_1(x,y)$ for all $x,y \in X_1$. It is *contractive* if there exists some $\delta < 1$ such that $d_2(f(x),f(y)) \leq \delta \cdot d_1(x,y)$ for all $x,y \in X_1$. By multiplication of the distances of (X,d) with a non-negative factor $\delta < 1$, one obtains a new ultrametric space, $\delta \cdot (X,d) = (X,d')$ where $d'(x,y) = \delta \cdot d(x,y)$.

The complete, 1-bounded, non-empty, ultrametric spaces and non-expansive functions between them form a Cartesian closed category CBUlt_{ne}. Products are given by the set-theoretic product where the distance is the maximum of the componentwise distances. 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 underlying set, and the distance function is given by $d_{X_1 \to X_2}(f,g) = \sup\{d_2(f(x),g(x)) \mid x \in X_1\}$.

The notation $x \stackrel{n}{=} y$ means that $d(x,y) \leq 2^{-n}$. Each relation $\stackrel{n}{=}$ is an equivalence relation because of the ultrametric inequality; we refer to the relation $\stackrel{n}{=}$ as "n-equality." Since the distances are bounded by 1, $x \stackrel{0}{=} y$ always holds, and the n-equalities become finer as n increases. If $x \stackrel{n}{=} y$ holds for all n then $x = y$.

Uniform predicates, worlds and world extension. Let $(A,\sqsubseteq)$ be a partially ordered set. An *upwards closed, uniform predicate* on A is a subset $p \subseteq \mathbb{N} \times A$ that is downwards closed in the first and upwards closed in the second component: if $(k,a) \in p$, $j \leq k$ and $a \sqsubseteq b$, then $(j,b) \in p$. We write $\mathit{UPred}(A)$ for the set of all upwards closed, uniform predicates on A, and we define $p_{[k]} = \{(j,a) \mid j < k\}$. Note that $p_{[k]} \in \mathit{UPred}(A)$. We equip $\mathit{UPred}(A)$ with the distance function $d(p,q) = \inf\{2^{-n} \mid p_{[n]} = q_{[n]}\}$, which makes $(\mathit{UPred}(A),d)$ an object of CBUlt_{ne}.

In our model, we use $\mathit{UPred}(A)$ with the following concrete instances for the partial order $(A,\sqsubseteq)$: (1) *heaps* $(\mathit{Heap},\sqsubseteq)$, where $h \sqsubseteq h'$ iff $h' = h \cdot h_0$ for some $h_0 \# h$, (2) *values* $(\mathit{Val},\sqsubseteq)$, where $u \sqsubseteq v$ iff $u = v$, and (3) *stateful values* $(\mathit{Val} \times \mathit{Heap},\sqsubseteq)$, where $(u,h) \sqsubseteq (v,h')$ iff $u = v$ and $h \sqsubseteq h'$. We also use variants of the latter two instances where the set *Val* is replaced by the set of value substitutions, *Env*, and by the set

of closed expressions, *Exp*. On $UPred(Heap)$, ordered by subset inclusion, we have a complete Heyting BI algebra structure [3]. Below we only need the separating conjunction and its unit I, given by

$$p_1 * p_2 \;=\; \{(k,h) \mid \exists h_1,h_2.\; h = h_1 \cdot h_2 \;\wedge\; (k,h_1) \in p_1 \wedge\; (k,h_2) \in p_2\} \quad \text{and} \quad I \;=\; \mathbb{N} \times Heap\;.$$

It is well-known that one can solve recursive domain equations in $CBUlt_{ne}$ by an adaptation of the inverse-limit method from classical domain theory [2]. In particular, with regard to (2) above:

Theorem 1. *There exists a unique (up to isomorphism)* $(X,d) \in CBUlt_{ne}$ *s.t.* $\iota : \frac{1}{2}\cdot X \to UPred(Heap) \cong X$.

Using the pointwise lifting of separating conjunction to $1/2\cdot X \to UPred(Heap)$ we define a *composition operation* on X, which reflects the syntactic abbreviation $C_1 \circ C_2 = C_1 \otimes C_2 * C_2$ of conjoining C_1 and C_2 and additionally applying an invariant extension to C_1. Formally, $\circ : X \times X \to X$ is a non-expansive operation that for all $p,q,x \in X$ satisfies

$$\iota^{-1}(p \circ q)(x) \;=\; \iota^{-1}(p)(q \circ x) * \iota^{-1}(q)(x)\;,$$

and which can be defined by an easy application of Banach's fixed point theorem as in [11]. One can show that this operation is associative and has a left and right unit given by $emp = \iota(\lambda w.I)$; thus $(X, \circ, emp)$ is a monoid in $CBUlt_{ne}$. Using $\circ$ we define an *extension operation* $\otimes : Y^{(1/2\cdot X)} \times X \to Y^{(1/2\cdot X)}$ for any $Y \in CBUlt_{ne}$ by $(f \otimes x)(x') = f(x \circ x')$. Without going into details, let us remark that this operation is the semantic counterpart to the syntactic invariant extension, and thus plays a key role in explaining the frame and anti-frame rules. However, for Pottier's anti-frame rule we also need to ensure that specifications are not invalidated by invariant extension. This requirement is stated via monotonicity, as we discuss next.

Relations on ultrametric spaces and hereditarily monotonic worlds. As a conseqence of the fact that $\circ$ defines a monoid structure on X there is an induced preorder on X:

$$x \sqsubseteq y \;\Leftrightarrow\; \exists x_0.\; y = x \circ x_0\;.$$

For modelling the anti-frame rule, we aim for a set of worlds similar to $X \cong 1/2 \cdot X \to UPred(Heap)$ but where the function space consists of the non-expansive functions that are additionally monotonic, with respect to the order induced by $\circ$ on X and with respect to set inclusion on $UPred(Heap)$:

$$(W, \sqsubseteq) \;\cong\; \tfrac{1}{2} \cdot (W, \sqsubseteq) \to_{mon} (UAdm, \subseteq)\;. \tag{3}$$

Because the definition of the order $\sqsubseteq$ (induced by $\circ$) already uses the isomorphism between left-hand and right-hand side, and because the right-hand side depends on the order for the monotonic function space, the standard existence theorems for solutions of recursive domain equations do not appear to apply to (3). Previously we have constructed a solution to this equation explicitly as inverse limit of a suitable chain of approximations [12]. We show in the following that we can alternatively carve out from X a suitable subset of what we call *hereditarily monotonic* functions. This subset needs to be defined recursively.

Let $\mathscr{R}$ be the collection of all non-empty and closed relations $R \subseteq X$. Given $R \in \mathscr{R}$, we define

$$R_{[n]} \stackrel{def}{=} \{y \mid \exists x \in X.\; x \stackrel{n}{=} y \;\wedge\; x \in R\}\;.$$

Thus, $R_{[n]}$ is the set of all points within distance 2^{-n} of R. Note that $R_{[n]} \in \mathscr{R}$. In fact, $R \subseteq R_{[n]}$ by the reflexivity of n-equivalence, and if $(y_k)_{k\in\mathbb{N}}$ is a sequence in $R_{[n]}$ with limit y then $d(y_k,y) \leq 2^{-n}$ for some k, i.e., $y_k \stackrel{n}{=} y$. So there exists $x \in X$ with $x \in R$ and $x \stackrel{n}{=} y_k$, and hence $x \stackrel{n}{=} y$ which gives $\lim_n y_n \in R_{[n]}$.

We make some further observations that follow from the properties of n-equality on X. First, $R \subseteq R_{[n]}$ by the reflexivity of n-equivalence, and $R \subseteq S$ implies $R_{[n]} \subseteq S_{[n]}$ for any $R,S \in \mathscr{R}$. Moreover, using the fact that the n-equalities become increasingly finer it follows that $(R_{[m]})_{[n]} = R_{[\min(m,n)]}$ for all $m,n \in \mathbb{N}$, so in particular each $(\cdot)_{[n]}$ is a closure operation on $\mathscr{R}$. As a consequence, we have $R \subseteq \ldots \subseteq R_{[n]} \subseteq \ldots \subseteq R_{[1]} \subseteq R_{[0]}$. Finally, $R = S$ if and only if $R_{[n]} = S_{[n]}$ for all $n \in \mathbb{N}$.

Proposition 2. *Let $d : \mathscr{R} \times \mathscr{R} \to \mathbb{R}$ be defined by $d(R,S) = \inf\{2^{-n} \mid R_{[n]} = S_{[n]}\}$, where we take* $\inf \emptyset$ *to mean 1. Then $(\mathscr{R}, d)$ is a complete, 1-bounded, non-empty ultrametric space. The limit of a Cauchy chain $(R_n)_{n\in\mathbb{N}}$ with $d(R_n, R_{n+1}) \leq 2^{-n}$ is given by $\bigcap_n (R_n)_{[n]}$, and in particular $R = \bigcap_n R_{[n]}$ for any $R \in \mathscr{R}$.*

We will now define the set of hereditarily monotonic functions W as a recursive predicate on X. Let the function Φ on subsets of X be defined by $\Phi(A) = \{\iota(p) \mid \forall x, x_0 \in A.\ p(x) \subseteq p(x \circ x_0)\}$. If $A \in \mathscr{R}$ then $\Phi(A)$ is non-empty and closed (i.e., Φ restricts to a function on $\mathscr{R}$), and it is contractive. By Proposition 2 and the Banach theorem we can now define W as the (uniquely determined) fixed point of Φ and obtain

$$w \in W \;\Leftrightarrow\; \exists p.\ w = \iota(p) \;\wedge\; \forall w, w_0 \in W.\ p(w) \subseteq p(w \circ w_0)\ .$$

Note that W thus constructed does not quite satisfy (3). We do not have an isomorphism between W and the non-expansive and monotonic functions from W (viewed as an ultrametric space itself), but rather between W and all functions from X that *restrict* to monotonic functions whenever applied to hereditarily monotonic arguments. Keeping this in mind, we abuse notation and write

$$\tfrac{1}{2} \cdot W \to_{mon} \mathit{UPred}(A) \stackrel{def}{=} \{p : \tfrac{1}{2} \cdot X \to \mathit{UPred}(A) \mid \forall w_1, w_2 \in W.\ p(w_1) \subseteq p(w_1 \circ w_2)\}\ .$$

Then, for our particular application of interest, we also have to ensure that all the operations restrict appropriately. First, by induction we show that for all $n \in \mathbb{N}$, if $w_1, w_2 \in W$ then $w_1 \circ w_2 \in W_{[n]}$, and this entails that the composition operation $\circ$ restricts to W. In turn, this means that the $\otimes$ operator restricts accordingly: if $w \in W$ and p is in $\frac{1}{2} \cdot W \to_{mon} \mathit{UPred}(A)$ then so is $p \otimes w$.

4 Possible World Semantics of Capabilities

We define semantic domains for the capabilities and types of the calculus described in Section 2,

$$\begin{aligned} \mathit{Cap} &= \tfrac{1}{2} \cdot W \to_{mon} \mathit{UPred}(\mathit{Heap}) \\ \mathit{VT} &= \tfrac{1}{2} \cdot W \to_{mon} \mathit{UPred}(\mathit{Val}) \\ \mathit{MT} &= \tfrac{1}{2} \cdot W \to_{mon} \mathit{UPred}(\mathit{Val} \times \mathit{Heap})\ , \end{aligned}$$

so that $p \in \mathit{Cap}$ if and only if $\iota(p) \in W$. Next, we define operations on the semantic domains that correspond to the syntactic type and capability constructors. The most interesting of these is the one for arrow types. Given $p, q \in 1/2 \cdot X \to \mathit{UPred}(\mathit{Val} \times \mathit{Heap})$, $p \to q$ in $\frac{1}{2} \cdot X \to \mathit{UPred}(\mathit{Val})$ is defined by

$$\begin{aligned} (p \to q)(x) \stackrel{def}{=} \{(k, \mathsf{fun}\, f(y){=}t) \mid \forall j < k.\ \forall w \in W.\ \forall r \in \mathit{UPred}(\mathit{Heap}).\ \\ \forall (j,(v,h)) \in p(x \circ w) * \iota^{-1}(x \circ w)(\mathit{emp}) * r. \\ (j, (t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathscr{E}(q)(x \circ w) * r\}\ , \end{aligned} \tag{4}$$

where $\mathscr{E}(q)$ is the extension of a world-indexed, uniform predicate on $\mathit{Val} \times \mathit{Heap}$ to one on $\mathit{Exp} \times \mathit{Heap}$. It is here where the index is linked to the operational semantics: $(k,(t,h)) \in \mathscr{E}(q)(x)$ iff for all $j \leq k, t', h'$,

$$(t\,|\,h) \longmapsto^j (t'\,|\,h') \;\wedge\; (t'\,|\,h') \text{ irreducible} \;\Rightarrow\; (k{-}j, (t', h')) \in \textstyle\bigcup_{w' \in W} q(x \circ w') * \iota^{-1}(x \circ w')(\mathit{emp})\ .$$

Definition (4) realizes the key ideas of our model as follows. First, the universal quantification over $w \in W$ and subsequent use of the world $x \circ w$ builds in monotonicity, and intuitively means that $p \to q$ is parametric in (and hence preserves) invariants that have been added by the procedure's context. In particular, (4) states that procedure application preserves this invariant, when viewed as the predicate $\iota^{-1}(x \circ w)(\mathit{emp})$. By also conjoining r as an invariant we "bake in" the first-order frame property, which

results in a subtyping axiom $\chi_1 \to \chi_2 \leq \chi_1 * C \to \chi_2 * C$ in the type system. The existential quantification over w', in the definition of $\mathscr{E}$, allows us to absorb a part of the local heap description into the world. Finally, the quantification over indices $j < k$ in (4) achieves that $(p \to q)(x)$ is uniform. There are two explanations why we require that j be *strictly* less than k. Technically, the use of $\iota^{-1}(x \circ w)$ in the definition "undoes" the scaling by $1/2$, and $j < k$ is needed to ensure the non-expansiveness of $p \to q$ as a function $1/2 \cdot W \to \mathit{UPred}(\mathit{Val})$. Moreover, it lets us prove the typing rule for *recursive* functions by induction on k. Intuitively, the use of $j < k$ for the arguments suffices since application consumes a step.

All these constructors restrict to $\mathit{Cap}, \mathit{VT}$ and MT, respectively. With their help it becomes straightforward to define the interpretation of capabilities and types, and to verify that the type equivalences hold with respect to this interpretation. The semantics of typing judgements is defined in analogy to (4), but also universally quantifying over worlds and indices, and it validates the typing rules of the calculus.

5 Conclusion and Future Work

To justify proof rules that take advantage of hidden state, like the frame and anti-frame rules, one needs semantic models that adequately capture this aspect of programming languages. In this paper, we have described the construction of a suitable possible worlds model where the worlds are given by a recursively defined predicate on a recursively defined metric space. In contrast to a similar model [12] which involved a tedious explicit inverse-limit construction, the present approach uses standard existence and fixed point theorems. We believe that this method provides a realistic approach to study frame and anti-frame rules in the presence of other programming language features, and to investigate the soundness of generalizations of these rules that have recently proposed by Pottier [10].

References

[1] A. Ahmed, M. Fluet, and G. Morrisett. L3: A linear language with locations. *Fundam. Inf.*, 77(4):397–449, 2007.

[2] 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.

[3] B. Biering, L. Birkedal, and N. Torp-Smith. Bi-hyperdoctrines, higher-order separation logic, and abstraction. *ACM Trans. Program. Lang. Syst.*, 29(5), 2007.

[4] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. Draft, April 2010.

[5] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In *FOSSACS*, pages 456–470, 2009.

[6] L. Birkedal, N. Torp-Smith, and H. Yang. Semantics of separation-logic typing and higher-order frame rules for Algol-like languages. *LMCS*, 2(5:1), 2006.

[7] A. Charguéraud and F. Pottier. Functional translation of a calculus of capabilities. In *ICFP*, pages 213–224, 2008.

[8] K. Crary, D. Walker, and G. Morrisett. Typed memory management in a calculus of capabilities. In *POPL*, pages 262–275, 1999.

[9] F. Pottier. Hiding local state in direct style: a higher-order anti-frame rule. In *LICS*, pages 331–340, 2008.

[10] F. Pottier. Generalizing the higher-order frame and anti-frame rules. Unpublished, July 2009.

[11] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In *CSL*, pages 440–454, 2009.

[12] J. Schwinghammer, H. Yang, L. Birkedal, F. Pottier, and B. Reus. A semantic foundation for hidden state. In *FOSSACS*, pages 2–16, 2010.

[13] M. B. Smyth. Topology. In *Handbook of Logic in Computer Science*, volume 1. Oxford Univ. Press, 1992.

$$
\begin{array}{rcl}
v & ::= & x \mid () \mid \mathsf{inj}^i\, v \mid (v_1, v_2) \mid \mathsf{fun}\, f(x){=}t \mid l \\
t & ::= & v \mid (v\,t) \mid \mathsf{case}(v_1, v_2, v) \mid \mathsf{proj}^i\, v \mid \mathsf{ref}\, v \mid \mathsf{get}\, v \mid \mathsf{set}\, v
\end{array}
$$

Figure 1: Syntax of values and expressions

$$
\begin{array}{rll}
(\mathsf{fun}\, f(x){=}t)\, v \,|\, h \longmapsto t[f := \mathsf{fun}\, f(x){=}t, x := v] \,|\, h & & \\
\mathsf{proj}^i(v_1, v_2) \,|\, h \longmapsto v_i \,|\, h & & \text{for } i = 1,2 \\
\mathsf{case}(v_1, v_2, \mathsf{inj}^i v) \,|\, h \longmapsto v_i\, v \,|\, h & & \text{for } i = 1,2 \\
\mathsf{ref}\, v \,|\, h \longmapsto l \,|\, h{\cdot}[l \mapsto v] & & \text{if } l \notin \mathsf{dom}(h) \\
\mathsf{get}\, l \,|\, h \longmapsto h(l) \,|\, h & & \text{if } l \in \mathsf{dom}(h) \\
\mathsf{set}\,(l, v) \,|\, h \longmapsto () \,|\, h[l := v] & & \text{if } l \in \mathsf{dom}(h) \\
v\,t \,|\, h \longmapsto v\,t' \,|\, h' & & \text{if } t \,|\, h \longmapsto t' \,|\, h'
\end{array}
$$

Figure 2: Operational semantics

A Definitions

In this section we give the details of the programming language and the type and capability system. For more details and motivation we refer to [7, 9, 4, 12].

Figures 1 and 2 give the syntax and operational semantics of a standard call-by-value higher-order language with recursive procedures. Figures 3 and 4 give the syntax and a structural equivalence relation on types, and Figure 5 presents some subtyping axioms. Figure 6 gives the typing rules that define the typing judgements for values and expressions.

Variables	$\xi ::= \alpha \mid \beta \mid \gamma \mid \sigma$
Capabilities	$C ::= C \otimes C \mid \emptyset \mid C * C \mid \{\sigma : \theta\} \mid \exists \sigma.C \mid \gamma \mid \mu\gamma.C \mid \forall \xi.C$
Value types	$\tau ::= \tau \otimes C \mid 0 \mid 1 \mid \mathsf{int} \mid \tau + \tau \mid \tau \times \tau \mid \chi \to \chi \mid [\sigma] \mid \alpha \mid \mu\alpha.\tau \mid \forall \xi.\tau$
Memory types	$\theta ::= \theta \otimes C \mid \tau \mid \theta + \theta \mid \theta \times \theta \mid \mathsf{ref}\, \theta \mid \theta * C \mid \exists \sigma.\theta \mid \beta \mid \mu\beta.\theta \mid \forall \xi.\theta$
Computation types	$\chi ::= \chi \otimes C \mid \tau \mid \chi * C \mid \exists \sigma.\chi$
Value environments	$\Delta ::= \Delta \otimes C \mid \varnothing \mid \Delta, x{:}\tau$
Linear environments	$\Gamma ::= \Gamma \otimes C \mid \varnothing \mid \Gamma, x{:}\chi \mid \Gamma * C$

Figure 3: Capabilities and types

monoids

$$C_1 \circ C_2 \stackrel{def}{=} (C_1 \otimes C_2) * C_2 \qquad C_1 * C_2 = C_2 * C_1 \tag{5}$$
$$(C_1 \circ C_2) \circ C_3 = C_1 \circ (C_2 \circ C_3) \qquad (C_1 * C_2) * C_3 = C_1 * (C_2 * C_3) \tag{6}$$
$$C \circ \emptyset = C \qquad C * \emptyset = C \tag{7}$$

monoid actions

$$(\cdot \otimes C_1) \otimes C_2 = \cdot \otimes (C_1 \circ C_2) \qquad \cdot \otimes \emptyset = \cdot \tag{8}$$
$$(\cdot * C_1) * C_2 = \cdot * (C_1 * C_2) \qquad \cdot * \emptyset = \cdot \tag{9}$$

action by $*$ on singleton

$$\{\sigma : \theta\} * C = \{\sigma : \theta * C\} \tag{10}$$

action by $*$ on linear environments

$$(\Gamma, x{:}\chi) * C = \Gamma, x{:}(\chi * C) = (\Gamma * C), x{:}\chi \tag{11}$$

action by $\otimes$ on capabilities, types, and environments

$$(\cdot * \cdot) \otimes C = (\cdot \otimes C) * (\cdot \otimes C) \tag{12}$$
$$(\exists \sigma . \cdot) \otimes C = \exists \sigma . (\cdot \otimes C) \qquad \text{if } \sigma \notin \mathit{RegNames}(C) \tag{13}$$

$$\emptyset \otimes C = \emptyset \tag{14}$$
$$\{\sigma : \theta\} \otimes C = \{\sigma : \theta \otimes C\} \tag{15}$$

$$0 \otimes C = 0 \tag{16}$$
$$1 \otimes C = 1 \tag{17}$$
$$\mathsf{int} \otimes C = \mathsf{int} \tag{18}$$
$$(\theta_1 + \theta_2) \otimes C = (\theta_1 \otimes C) + (\theta_2 \otimes C) \tag{19}$$
$$(\theta_1 \times \theta_2) \otimes C = (\theta_1 \otimes C) \times (\theta_2 \otimes C) \tag{20}$$
$$(\forall \xi . \theta) \otimes C = \forall \xi . (\theta \otimes C) \qquad \text{if } \xi \notin \mathrm{fv} C \tag{21}$$
$$(\chi_1 \to \chi_2) \otimes C = (\chi_1 \circ C) \to (\chi_2 \circ C) \tag{22}$$
$$[\sigma] \otimes C = [\sigma] \tag{23}$$
$$(\mathsf{ref}\, \theta) \otimes C = \mathsf{ref}\, (\theta \otimes C) \tag{24}$$

$$\varnothing \otimes C = \varnothing \tag{25}$$
$$(\Gamma, x{:}\chi) \otimes C = (\Gamma \otimes C), x{:}(\chi \otimes C) \tag{26}$$
$$(\Gamma * C_1) \otimes C_2 = (\Gamma \otimes C_2) * (C_1 \otimes C_2) \tag{27}$$

region abstraction

$$\exists \sigma_1 . \exists \sigma_2 . \cdot = \exists \sigma_2 . \exists \sigma_1 . \cdot \tag{28}$$
$$\cdot * (\exists \sigma . C) = \exists \sigma . (\cdot * C) \tag{29}$$
$$\{\sigma_1 : \exists \sigma_2 . \theta\} = \exists \sigma_2 . \{\sigma_1 : \theta\} \qquad \text{where } \sigma_1 \neq \sigma_2 \tag{30}$$

focusing

$$\{\sigma_1 : \mathsf{ref}\, \theta\} = \exists \sigma_2 . \{\sigma_1 : \mathsf{ref}\, [\sigma_2]\} * \{\sigma_2 : \theta\} \tag{31}$$
$$\{\sigma : \theta_1 \times \theta_2\} = \exists \sigma_1 . \{\sigma : [\sigma_1] \times \theta_2\} * \{\sigma_1 : \theta_1\} \tag{32}$$
$$\{\sigma : \theta_1 \times \theta_2\} = \exists \sigma_2 . \{\sigma : \theta_1 \times [\sigma_2]\} * \{\sigma_2 : \theta_2\} \tag{33}$$
$$\{\sigma : \theta_1 + 0\} = \exists \sigma_1 . \{\sigma : [\sigma_1] + 0\} * \{\sigma_1 : \theta_1\} \tag{34}$$
$$\{\sigma : 0 + \theta_2\} = \exists \sigma_2 . \{\sigma : 0 + [\sigma_2]\} * \{\sigma_2 : \theta_2\} \tag{35}$$

recursion

$$\mu \gamma . C = C[\gamma{:=}\mu \gamma . C] \tag{36}$$
$$\mu \alpha . \tau = \tau[\alpha{:=}\mu \alpha . \tau] \tag{37}$$
$$\mu \beta . \theta = \theta[\beta{:=}\mu \beta . \theta] \tag{38}$$

Figure 4: Structural equivalence

(first-order) frame axiom

$$\chi_1 \to \chi_2 \leq (\chi_1 * C) \to (\chi_2 * C) \tag{39}$$

free

$$C_1 * C_2 \leq C_1 \tag{40}$$

singletons

$$\tau \leq \exists\sigma.[\sigma] * \{\sigma : \tau\} \tag{41}$$

$$[\sigma] * \{\sigma : \tau\} \leq \tau * \{\sigma : \tau\} \tag{42}$$

Figure 5: Some subtyping axioms

$$\textsc{Var}\ \frac{(x:\tau)\in\Delta}{\Delta \vdash x:\tau} \qquad \textsc{Unit}\ \frac{}{\Delta\vdash():1} \qquad \textsc{Inj}\ \frac{\Delta\vdash v:\tau_i}{\Delta\vdash(\mathsf{inj}^i\ v):(\tau_1+\tau_2)} \qquad \textsc{Pair}\ \frac{\Delta\vdash v_1:\tau_1 \quad \Delta\vdash v_2:\tau_2}{\Delta\vdash(v_1,v_2):(\tau_1\times\tau_2)} \qquad \textsc{RecFun}\ \frac{\Delta, f:\chi_1\to\chi_2, x:\chi_1 \Vdash t:\chi_2}{\Delta\vdash \mathsf{fun}\, f(x){=}t:\chi_1\to\chi_2}$$

$$\textsc{Val}\ \frac{\Delta\vdash v:\tau}{\Delta\Vdash v:\tau} \qquad \textsc{App}\ \frac{\Delta\vdash v:\chi_1\to\chi_2 \quad \Delta,\Gamma\Vdash t:\chi_1}{\Delta,\Gamma\Vdash(vt):\chi_2} \qquad \textsc{Proj-1}\ \frac{\Gamma\Vdash v:[\sigma]*\{\sigma:\tau_1\times\theta_2\}}{\Gamma\Vdash \mathsf{proj}^1 v:\tau_1*\{\sigma:\tau_1\times\theta_2\}} \qquad \textsc{Proj-2}\ \frac{\Gamma\Vdash v:[\sigma]*\{\sigma:\theta_1\times\tau_2\}}{\Gamma\Vdash \mathsf{proj}^2 v:\tau_2*\{\sigma:\theta_1\times\tau_2\}}$$

$$\textsc{Case}\ \frac{\begin{array}{c}\Delta\vdash v_1:(\exists\sigma_1.[\sigma_1]*\{\sigma:[\sigma_1]+0\}*\{\sigma_1:\theta_1\}*C)\to\chi \\ \Delta\vdash v_2:(\exists\sigma_2.[\sigma_2]*\{\sigma:0+[\sigma_2]\}*\{\sigma_2:\theta_2\}*C)\to\chi \\ \Delta,\Gamma\Vdash v:[\sigma]*\{\sigma:\theta_1+\theta_2\}*C\end{array}}{\Delta,\Gamma\Vdash \mathsf{case}(v_1,v_2,v):\chi} \qquad \forall\textsc{-Intro}\ \frac{\Delta\vdash v:\tau}{\Delta\vdash v:\forall\xi.\tau}\,\xi\notin\Delta \qquad \forall\textsc{-Elim-1}\ \frac{\Delta\vdash v:\forall\alpha.\tau}{\Delta\vdash v:\tau[\alpha:=\tau']}$$

$$\textsc{Ref}\ \frac{\Gamma\Vdash v:\tau}{\Gamma\Vdash \mathsf{ref}\, v:\exists\sigma.[\sigma]*\{\sigma:\mathsf{ref}\,\tau\}} \qquad \textsc{Get}\ \frac{\Gamma\Vdash v:[\sigma]*\{\sigma:\mathsf{ref}\,\tau\}}{\Gamma\Vdash \mathsf{get}\, v:\tau*\{\sigma:\mathsf{ref}\,\tau\}} \qquad \textsc{Set}\ \frac{\Gamma\Vdash v:([\sigma]\times\tau_2)*\{\sigma:\mathsf{ref}\,\tau_1\}}{\Gamma\Vdash \mathsf{set}\, v:1*\{\sigma:\mathsf{ref}\,\tau_2\}}$$

$$\textsc{Shallow-Frame}\ \frac{\Gamma\Vdash t:\chi}{\Gamma*C\Vdash t:\chi*C} \qquad \textsc{Deep-Frame}\ \frac{\Gamma\Vdash t:\chi}{(\Gamma\otimes C)*C\Vdash t:(\chi\otimes C)*C} \qquad \textsc{Anti-Frame}\ \frac{\Gamma\otimes C\Vdash t:(\chi\otimes C)*C}{\Gamma\Vdash t:\chi} \qquad \textsc{Sub}\ \frac{\Gamma\Vdash t:\chi_1 \quad \chi_1\leq\chi_2}{\Gamma\Vdash t:\chi_2}$$

Figure 6: Typing of values and expressions

B Proofs

B.1 Relations on complete ultrametric spaces

Let $(X,d)\in \mathit{CBUlt}_{ne}$, and let $\mathscr{R}(X)$ be the collection of all non-empty and closed relations $R\subseteq X$.

Proposition 3. *Let $d:\mathscr{R}(X)\times\mathscr{R}(X)\to\mathbb{R}$ be defined by $d(R,S)=\inf\{2^{-n}\mid R_{[n]}=S_{[n]}\}$, where we take $\inf\emptyset$ to mean 1. Then $(\mathscr{R}(X),d)$ is a complete, 1-bounded, non-empty ultrametric space.*

Proof. First, $\mathscr{R}(X)$ is non-empty since it contains X itself. Next, since $R=S$ is equivalent to $R_{[n]}=S_{[n]}$ for all $n\in\mathbb{N}$, it follows that $d(R,S)=0$ if and only if $R=S$. That the ultrametric inequality $d(R,S)\leq \max\{d(R,T),d(T,S)\}$ holds is immediate by the definition of d, as is the fact that d is symmetric and 1-bounded.

To show completeness, assume that $(R_n)_{n\in\mathbb{N}}$ is a Cauchy sequence in $\mathscr{R}(X)$. Without loss of generality we may assume that $d(R_n,R_{n+1})\leq 2^{-n}$ holds for all $n\in\mathbb{N}$, and therefore that $(R_n)_{[n]}=(R_{n+1})_{[n]}$ for

all $n \geq 0$. Writing S_n for $(R_n)_{[n]}$, we define $R \subseteq X$ by

$$R \stackrel{def}{=} \bigcap_{n \geq 0} S_n \ .$$

R is closed since each S_n is closed. We now prove that $R(\cdot, a)$ is non-empty, and therefore $R \in \mathscr{R}(X)$, by inductively constructing a sequence $(x_n)_{n\in\mathbb{N}}$ with $x_n \in S_n$: Let x_0 be an arbitrary element in $S_0 = X$. Having chosen $x_0, \ldots, x_n$, we pick some $x_{n+1} \in S_{n+1}$ such that $x_{n+1} \stackrel{n}{=} x_n$; this is always possible because $S_n = (S_{n+1})_{[n]}$ by our assumption on the sequence $(R_n)_{n\in\mathbb{N}}$. Clearly this is a Cauchy sequence in X, and from $S_n \supseteq S_{n+1}$ it follows that $(x_n)_{n\geq k}$ is in fact a sequence in S_k for each $k \in \mathbb{N}$. But then also $\lim_{n\in\mathbb{N}} x_n$ is in S_k for each k, and thus also in R.

We now prove that R is the limit of the sequence $(R_n)_{n\in\mathbb{N}}$. By definition of d it suffices to show that $R_{[k]} = (R_k)_{[k]}$ for all $k \geq 1$, or equivalently, that $R_{[k]} = S_k$. From the definition of R, $R \subseteq S_k$, which immediately entails $R_{[k]} \subseteq (S_k)_{[k]} = S_k$.

To prove the other direction, i.e., $S_k \subseteq R_{[k]}$, assume that $x \in S_k$. To show that $x \in R_{[k]}$ we inductively construct a Cauchy sequence $(x_n)_{n\geq k}$ with $x_n \in S_n$, $x_k = x$ and $x_{n+1} \stackrel{n}{=} x_n$ analogously to the one above. Then $\lim_m x_m$ is in S_n for each $n \geq 0$, and thus also in R. Since $d_X(x_k, \lim_{n\geq k} x_n) \leq 2^{-k}$ by the ultrametric inequality, $x_k \in R_{[k]}$, or equivalently, $x \in R_{[k]}$. □

B.2 Hereditarily monotonic recursive worlds

Let the function $\Phi : \mathscr{P}(X) \to \mathscr{P}(X)$ be defined by $\Phi(A) = \{\iota(p) \mid \forall x, x_0 \in A.\ p(x) \subseteq p(x \circ x_0)\}$.

Lemma 4. *For each $A \in \mathscr{R}$, $\Phi(A)$ is non-empty and closed.*

Proof. $\Phi(A)$ is non-empty since it contains the constant functions into $UPred(Heap)$. As in [5], one can use the completeness of $UPred(Heap)$ and the way its metric interacts with subset inclusion to show that $\Phi(A)$ is closed. □

Lemma 5. *Φ is contractive: $A \stackrel{n}{=} B$ implies $\Phi(A) \stackrel{n+1}{=} \Phi(B)$.*

Proof. Let $n \geq 0$ and assume $A \stackrel{n}{=} B$. Let $\iota(p) \in \Phi(A)_{[n+1]}$. We must show that $\iota(p) \in \Phi(B)_{[n+1]}$. By definition there exists $\iota(q) \in \Phi(A)$ such that $p \stackrel{n+1}{=} q$. Set $r(w) = q(w)_{[n+1]}$. Then $r \stackrel{n+1}{=} p$ and it suffices to show that $\iota(r) \in \Phi(B)$. To this end, let $w_0, w_1 \in B$. By assumption there exist $w_0', w_1' \in A$ such that $w_0' \stackrel{n}{=} w_0$ and $w_1' \stackrel{n}{=} w_1$ in X, or equivalently, $w_0' \stackrel{n+1}{=} w_0$ and $w_1' \stackrel{n+1}{=} w_1$ in $\frac{1}{2}\cdot X$. Using the non-expansiveness of $\circ$, this also implies $w_0' \circ w_1' \stackrel{n+1}{=} w_0 \circ w_1$ in $\frac{1}{2}\cdot X$. Since $q(w_0) \stackrel{n+1}{=} q(w_0') \subseteq q(w_0' \circ w_1') \stackrel{n+1}{=} q(w_0 \circ w_1)$ by the non-expansiveness of q and the assumption that $\iota(q) \in \Phi(A)$ we obtain the required inclusion $r(w_0) \subseteq r(w_0 \circ w_1)$. □

Lemma 6. *$\frac{1}{2}\cdot W \to_{mon} UPred(A)$ is a non-empty and closed subset of $\frac{1}{2}\cdot X \to UPred(A)$.*

Proof. Similar to the proof of Lemma 4. □

B.3 Closure of W under composition

Lemma 7. *For all $n \in \mathbb{N}$, if $w_1, w_2 \in W$ then $w_1 \circ w_2 \in W_{[n]}$.*

Since $W = \bigcap_n W_{[n]}$ it follows that $w_1, w_2 \in W \ \Rightarrow\ w_1 \circ w_2 \in W$.

Proof. Since $W_{[0]} = X$ the claim trivially holds in case $n = 0$. Now suppose $n > 0$ and let $w_1, w_2 \in W$; we must prove that $w_1 \circ w_2 \in W_{[n]}$. Let w_1' be such that $\iota^{-1}(w_1')(w) = \iota^{-1}(w_1)(w)_{[n]}$. Observe that $w_1' \in W$, and $w \stackrel{n}{=} w'$ in $\frac{1}{2} \cdot X$ implies $w_1'(w) = w_1'(w')$. Since $w_1 \stackrel{n}{=} w_1'$, the non-expansiveness of $\circ$ implies $w_1 \circ w_2 \stackrel{n}{=} w_1' \circ w_2$, and thus it suffices to show that $w_1' \circ w_2 \in W = \Phi(W)$. To see this, let $w, w_0 \in W$. Note that by induction hypothesis $w_2 \circ w \in W_{[n-1]}$, i.e., there exists $w' \in W$ such that $w' \stackrel{n}{=} w_2 \circ w$ holds in $\frac{1}{2} \cdot W$. We thus obtain

$$\begin{aligned}
\iota^{-1}(w_1' \circ w_2)(w) &= \iota^{-1}(w_1')(w_2 \circ w) * \iota^{-1}(w_2)(w) \\
&= \iota^{-1}(w_1')(w') * \iota^{-1}(w_2)(w) \\
&\subseteq \iota^{-1}(w_1')(w' \circ w_0) * \iota^{-1}(w_2 \circ w_0)(w) \\
&= \iota^{-1}(w_1')((w_2 \circ w) \circ w_0) * \iota^{-1}(w_2 \circ w_0)(w) \\
&= \iota^{-1}(w_1' \circ w_2)(w \circ w_0) \ ,
\end{aligned}$$

i.e., $w_1' \circ w_2 \in W$. □

B.4 Closure under extension

Lemma 8. *If $w \in W$ and $f \in \frac{1}{2} \cdot W \to_{mon} UPred(A)$ then $f \otimes w \in \frac{1}{2} \cdot W \to_{mon} UPred(A)$. Moreover, the assignment of $f \otimes w$ to f, w is non-expansive as a function of f and contractive as a function of w.*

Proof. Let $w_1, w_2 \in W$. Then $w \circ w_1 \in W$ by Lemma 7, and hence

$$(f \otimes w)(w_1) = f(w \circ w_1) \subseteq f((w \circ w_1) \circ w_2) = (f \otimes w)(w_1 \circ w_2)$$

by the assumption that f is in $\frac{1}{2} \cdot W \to_{mon} UPred(A)$. □

B.5 Closure under universal quantification

Suppose $F : S \to (\frac{1}{2} \cdot X \to UPred(A))$. Then we define $\forall F : \frac{1}{2} \cdot X \to UPred(A)$ by

$$(\forall F)(x) = \bigcap_{s \in S} F(s)(x)$$

Lemma 9. *With F as above, $\forall F$ is non-expansive, and $(\forall F)(x)$ is upwards closed and uniform. If $F(s) \in \frac{1}{2} \cdot W \to_{mon} UPred(A)$ for all $s \in S$ then $\forall F \in \frac{1}{2} \cdot W \to_{mon} UPred(A)$. The assignment of $\forall F$ to F is non-expansive.*

This observation can be used to justify quantification over types, capabilities and region names in *Cap*, *VT* and *MT*.

B.6 Recursion

Cap, *VT* and *MT* are non-empty and closed subsets of complete ultrametric spaces, by Lemma 6. Thus, any contractive function that restricts to these sets has a unique fixed point in the respective set. This observation can be used to justify recursive definitions of capabilities and types, noting that formal contractiveness of a syntactic type expression ensures contractiveness of its interpretation (see below). Moreover, the assignment to a contractive function of its unique fixed point is a non-expansive operation.

B.7 Closure of *Cap* under separating conjunction

Lemma 10. *If $f,g \in Cap$ then $f * g \in Cap$. Moreover, the assignment of $f * g$ to f,g is non-expansive.*

Proof. Let $w_1, w_2 \in W$, then $(f*g)(w_1) = f(w_1) * g(w_1) \subseteq f(w_1 \circ w_2) * g(w_1 \circ w_2) = (f*g)(w_1 \circ w_2)$ follows from the monotonicity of separating conjunction on $UPred(Heap)$. □

B.8 Closure of *Cap* under singletons

For $v \in Val$ and f in $\frac{1}{2}\cdot X \to UPred(Heap)$ define $\{v : f\}$ in $\frac{1}{2}\cdot X \to UPred(Heap)$ by

$$\{v : f\}(x) \stackrel{def}{=} \{(k,h) \mid (k,(v,h)) \in f(x)\}$$

Lemma 11. *With v, f as above, $\{v : f\}$ is non-expansive, and $\{v : f\}(x)$ is upwards closed and uniform for all $x \in X$. If $f \in Cap$ then $\{v : f\} \in Cap$, and the assignment of $\{v : f\}$ to f is non-expansive.*

Proof. The non-expansiveness of $\{v : f\}$ follows from the non-expansiveness of f. Similarly, the claim $\{v : f\}(x) \in UPred(Heap)$ follows from $f(x) \in UPred(Val \times Heap)$. Finally, if $w_1, w_2 \in W$ then $\{v : f\}(w_1) \subseteq \{v : f\}(w_1 \circ w_2)$ follows if $f \in Cap$, for then $f(w_1) \subseteq f(w_1 \circ w_2)$. □

B.9 Closure of *VT* under sums

For f_1, f_2 in $\frac{1}{2}\cdot X \to UPred(Val)$ define $f_1 + f_2$ in $\frac{1}{2}\cdot X \to UPred(Val)$ by

$$(f_1 + f_2)(x) \stackrel{def}{=} \{(k, \mathsf{inj}^i v) \mid (k-1, v) \in f_i(x)\}$$

Lemma 12. *With f_1, f_2 as above, $f_1 + f_2$ is non-expansive, and $(f_1 + f_2)(x)$ is uniform for all $x \in X$. If $f_1, f_2 \in VT$ then $f_1 + f_2 \in VT$, and the assignment of $f_1 + f_2$ to f_1, f_2 is contractive.*

Proof. Let $x \stackrel{n}{=} x'$. Then, for any $k \leq n$, $(k-1, v) \in f_i(x)$ iff $(k-1, v) \in f_i(x')$ by the non-expansiveness of f_i and the definition of the metric on $UPred(Val)$. Thus $(f_1 + f_2)(x) \stackrel{n+1}{=} (f_1 + f_2)(x')$. That $(f_1 + f_2)(x) \in UPred(Val)$ follows from $f_i(x) \in UPred(Val)$. Finally, if $w_1, w_2 \in W$ then $(f_1 + f_2)(w_1) \subseteq (f_1 + f_2)(w_1 \circ w_2)$ follows from by definition of $f_1 + f_2$ if $f_1, f_2 \in VT$. □

B.10 Closure of *VT* under products

For f_1, f_2 in $\frac{1}{2}\cdot X \to UPred(Val)$ define $f_1 \times f_2$ in $\frac{1}{2}\cdot X \to UPred(Val)$ by

$$(f_1 \times f_2)(x) \stackrel{def}{=} \{(k, (v_1, v_2)) \mid (k-1, v_i) \in f_i(x)\}$$

Lemma 13. *With f_1, f_2 as above, $f_1 \times f_2$ is non-expansive, and $(f_1 \times f_2)(x)$ is uniform for all $x \in X$. If $f_1, f_2 \in VT$ then $f_1 \times f_2 \in VT$, and the assignment of $f_1 \times f_2$ to f_1, f_2 is contractive.*

Proof. Let $x \stackrel{n}{=} x'$. Then, for any $k \leq n$, $(k-1, v) \in f_i(x)$ iff $(k-1, v) \in f_i(x')$ by the non-expansiveness of f_i and the definition of the metric on $UPred(Val)$. Thus $(f_1 \times f_2)(x) \stackrel{n+1}{=} (f_1 \times f_2)(x')$. That $(f_1 \times f_2)(x) \in UPred(Val)$ follows from $f_i(x) \in UPred(Val)$. Finally, if $w_1, w_2 \in W$ then $(f_1 \times f_2)(w_1) \subseteq (f_1 \times f_2)(w_1 \circ w_2)$ follows from by definition of $f_1 \times f_2$ if $f_1, f_2 \in VT$. □

B.11 Extension to expressions

For $p \in UPred(A \times Heap)$ and $r \in UPred(Heap)$ we define $p * r$ by

$$p * r = \{(k,(a,h \cdot h')) \mid (k,(a,h)) \in p \ \wedge \ (k,h') \in r\} \ .$$

Then $p * r \in UPred(A \times Heap)$. This operation can be lifted pointwise, and if $f \in MT$ and $p \in Cap$ then $f * p \in MT$.

Using the former operation on uniform predicates, we define the following extension of memory types from values to expressions.

Definition 14 (Expression typing)**.** *Let p in $\frac{1}{2} \cdot X \to UPred(Val \times Heap)$. Then the function $\mathscr{E}(p) : X \to UPred(Exp \times Heap)$ is defined by $(k,(t,h)) \in \mathscr{E}(p)(x)$ iff*

$$\forall j \leq k, t', h' .\ (t \,|\, h) \longmapsto^{j} (t' \,|\, h') \ \wedge \ (t' \,|\, h') \text{ irreducible} \\ \Rightarrow \ (k-j,(t',h')) \in \bigcup_{w \in W} p(x \circ w) * \iota^{-1}(x \circ w)(emp) \ .$$

Lemma 15. *With p as above, $\mathscr{E}(p)$ is non-expansive, and $\mathscr{E}(p)(x)$ is upwards closed and uniform for all $x \in X$. Moreover, the assignment of $\mathscr{E}(p)$ to p is non-expansive.*

B.12 Closure of *VT* under arrows

For p, q in $\frac{1}{2} \cdot X \to UPred(Val \times Heap)$ define $p \to q$ in $\frac{1}{2} \cdot X \to UPred(Val)$ by

$$(p \to q)(x) \stackrel{def}{=} \{(k, \mathsf{fun}\, f(y){=}t) \mid \forall j < k .\ \forall w \in W .\ \forall r \in UPred(Heap). \\ \forall (j,(v,h)) \in p(x \circ w) * \iota^{-1}(x \circ w)(emp) * r. \\ (j,(t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathscr{E}(q)(x \circ w) * r\}$$

Lemma 16. *With p, q as above, $p \to q$ is non-expansive, and $(p \to q)(x)$ is uniform for all $x \in X$. Moreover, $p \to q \in VT$, and the assignment of $p \to q$ to p, q is contractive.*

Proof. The non-expansiveness is straightforward to check, using Lemma 15. The uniformity is ensured by the explicit quantification over $j < k$ in the definition of $(p \to q)(x)$. Similarly, that $p \to q \in VT$ is guaranteed by the explicit quantification over $w \in W$ in the definition of $(p \to q)(x)$, using the closure of W under $\circ$ (Lemma 7). Finally, the contractiveness of $\cdot \to \cdot$ follows since $p(x \circ w)$ and $q(\circ w \circ w')$ are only considered up to index j which is strictly below k. □

B.13 Inclusion of *VT* into *MT*

The inclusion of value types into memory types,

$$f \in (\tfrac{1}{2} \cdot X \to UPred(Val)) \ \mapsto \ \lambda x. \{(k,(v,h)) \mid h \in Heap \ \wedge \ (k,v) \in f(x)\}$$

is non-expansive and maps into non-expansive functions from $\frac{1}{2} \cdot X$ to $UPred(Val \times Heap)$. If $f \in VT$ then the right hand side is in MT.

B.14 Closure of *MT* under sums

For f_1, f_2 in $\frac{1}{2} \cdot X \to UPred(Val \times Heap)$ define $f_1 + f_2$ in $\frac{1}{2} \cdot X \to UPred(Val \times Heap)$ by

$$(f_1 + f_2)(x) \stackrel{def}{=} \{(k,(\mathsf{inj}^i v, h)) \mid (k{-}1,(v,h)) \in f_i(x)\}$$

As with the sum types on values, this is well-defined, in MT if both f_1 and f_2 are in MT, and the assignment of $f_1 + f_2$ to f_1 and f_2 is contractive.

B.15 Closure of *MT* under products

For f_1, f_2 in $\frac{1}{2}\cdot X \to \mathit{UPred}(\mathit{Val}\times\mathit{Heap})$ define $f_1 \times f_2$ in $\frac{1}{2}\cdot X \to \mathit{UPred}(\mathit{Val}\times\mathit{Heap})$ by

$$(f_1 \times f_2)(x) \stackrel{def}{=} \{(k,((v_1,v_2),h_1\cdot h_2)) \mid (k{-}1,(v_i,h_i)) \in f_i(x)\}$$

As with the product types on values, this is well-defined, $f_1 \times f_2$ is in *MT* if both f_1 and f_2 are in *MT*, and the assignment of $f_1 \times f_2$ to f_1 and f_2 is contractive.

B.16 Closure of *MT* under reference types

For f in $\frac{1}{2}\cdot X \to \mathit{UPred}(\mathit{Val}\times\mathit{Heap})$ define $\mathit{ref}(f)$ in $\frac{1}{2}\cdot X \to \mathit{UPred}(\mathit{Val}\times\mathit{Heap})$ by

$$\mathit{ref}(f)(x) \stackrel{def}{=} \{(k,(l,h\cdot[l\mapsto v])) \mid (k{-}1,(v,h)) \in f(x)\}$$

Lemma 17. *With f as above, $\mathit{ref}(f)$ is non-expansive, and $\mathit{ref}(f)(x)$ is upwards closed and uniform for all $x \in X$. If $f \in MT$ then $\mathit{ref}(f) \in MT$, and the assignment of $\mathit{ref}(f)$ to f is contractive.*

B.17 Interpretation of types and capabilities

The interpretation depends on an environment η, which maps region names $\sigma \in \mathit{RegName}$ to closed values $\eta(\sigma) \in \mathit{Val}$, capability variables γ to semantic capabilities $\eta(\gamma) \in \mathit{Cap}$, and type variables α and β to semantic types $\eta(\alpha) \in \mathit{VT}$ and $\eta(\beta) \in \mathit{MT}$. Then, we use the semantic type constructors in the evident way, for instance defining $[\![\chi_1 \to \chi_2]\!]_\eta = [\![\chi_1]\!]_\eta \to [\![\chi_2]\!]_\eta$. Importantly, we have the extension operation available to interpret invariant extension:

$$[\![C_1 \otimes C_2]\!]_\eta = [\![C_1]\!]_\eta \otimes \iota([\![C_2]\!]_\eta)\ ,$$

and similarly for $\tau \otimes C$ and $\theta \otimes C$.

We end up with interpretations $[\![C]\!]_\eta \in \mathit{Cap}$, $[\![\tau]\!]_\eta \in \mathit{VT}$ and $[\![\theta]\!]_\eta \in \mathit{MT}$.

B.18 Distribution of $\otimes$ over $\to$

As an example of validating the type equivalence in the model we prove, on the semantic level, that the distribution axiom for arrow types.

Lemma 18. *Let $f_1, f_2 \in MT$ and $p \in \mathit{Cap}$. Then $(f_1 \to f_2) \otimes p = (f_1 \otimes p * p) \to (f_2 \otimes p * p)$.*

Proof. Let $x \in X$ and $(k,(\mathsf{fun}\, f(y){=}t)) \in ((f_1 \to f_2)\otimes p)(x) = (f_1 \to f_2)(\iota(p)\circ x)$. We must prove that $(k,(\mathsf{fun}\, f(y){=}t)) \in (f_1 \otimes p * p) \to (f_2 \otimes p * p)$. To this end, let $j < k$, $w \in W$, $r \in \mathit{UPred}(\mathit{Heap})$, and suppose

$$\begin{aligned}(j,(v,h)) &\in (f_1 \otimes p * p)(x\circ w) * \iota^{-1}(x\circ w)(\mathit{emp}) * r\\ &= f_1(\iota(p)\circ x\circ w) * p(x\circ w) * \iota^{-1}(x\circ w)(\mathit{emp}) * r\\ &= f_1(\iota(p)\circ x\circ w) * \iota^{-1}(\iota(p)\circ x\circ w)(\mathit{emp}) * r\ .\end{aligned}$$

Then, by assumption, $(j,(t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v],h)) \in \mathscr{E}(f_2)(\iota(p)\circ x\circ w) * r$. By unfolding the definition of $\mathscr{E}$, the latter is seen to be equivalent to

$$(j,(t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v],h)) \in \mathscr{E}(f_2 \otimes p * p)(x\circ w) * r\ ,$$

and thus $(k,(\mathsf{fun}\, f(y){=}t)) \in (f_1 \otimes p * p) \to (f_2 \otimes p * p)$.

The other direction is similar. □

Remark 19. Note that we did not use the fact that $f_1, f_2 \in MT$ and $p \in Cap$. This is in line with the semantics given in the earlier work by Birkedal et al. [4]. There, the anti-frame rule was not considered and a model based on the simpler set of worlds X sufficed; the language also contained all of the distribution axioms that we consider here.

B.19 First-order frame axiom

As an example of validating the subtyping axioms, we consider $\chi_1 \to \chi_2 \leq \chi_1 * C \to \chi_2 * C$.

Lemma 20. *Let $f_1, f_2 \in MT$ and $p \in Cap$. Let $w \in W$. Suppose that $(k, \mathsf{fun}\, f(y)=t) \in (f_1 \to f_2)(w)$. Then $(k, \mathsf{fun}\, f(y)=t) \in (f_1 * p \to f_2 * p)(w)$.*

Proof. By unfolding the definitions, and instantiating the universally quantified $r \in UPred(Heap)$ accordingly. The proof then relies on the fact that $p(w \circ w') \subseteq p(w \circ w' \circ w'')$ holds for all w', w'' in W, since $w \circ w' \in W$ and $p \in Cap$. □

B.20 Semantics of typing judgements

Recall that we have two kinds of judgments, one for typing of values and the other for the typing of expressions:

$$\Delta \vdash v : \tau \qquad\qquad \Gamma \Vdash t : \chi$$

The semantics of a value judgement simply establishes truth with respect to all worlds w, all environments η and all $k \in \mathbb{N}$:

$$\models (\Delta \vdash v : \tau) \stackrel{def}{\iff} \forall \eta.\, \forall w \in W.\, \forall k \in \mathbb{N}.\, \forall (k, \rho) \in [\![\Delta]\!]_\eta\, w.\, (k, \rho(v)) \in [\![\tau]\!]_\eta\, w\ .$$

Here $\rho(v)$ means the application of the substitution ρ to v. The judgement for expressions mirrors the interpretation of the arrow case for value types, in that there is also a quantification over heap predicates $r \in UPred(Heap)$ and an existential quantification over $w' \in W$ through the use of $\mathscr{E}$:

$$\begin{aligned}\models\ (\Gamma \Vdash t : \chi) \stackrel{def}{\iff}\ & \forall \eta.\, \forall w \in W.\, \forall k \in \mathbb{N}.\, \forall r \in UPred(Heap).\\ & \forall (k, (\rho, h)) \in [\![\Gamma]\!]_\eta\, w * \iota^{-1}(w)(emp) * r.\ (k, (\rho(t), h)) \in \mathscr{E}([\![\chi]\!]_\eta)(w) * r.\end{aligned}$$

The universal quantifications allow us to have frame rules: the universal quantification over worlds w ensures the soundness of the deep frame rule, and the universal quantification over heap predicates r validates the shallow frame rule, as we show next. The existential quantifier plays an important part in the verification of the anti-frame rule below.

B.21 Shallow frame rule

Soundness of the shallow frame rule is proved analogously to the soundness of the first-order frame axiom. In particular, it is essential that $[\![C]\!] \in Cap$ below:

Lemma 21. *Suppose $\models (\Gamma \Vdash t : \chi)$. Then $\models (\Gamma * C \Vdash t : \chi * C)$.*

B.22 Deep frame rule

Lemma 22. *Suppose* $\models (\Gamma \Vdash t : \chi)$. *Then* $\models (\Gamma \otimes C * C \Vdash t : \chi \otimes C * C)$.

Proof. We prove $\models (\Gamma \otimes C * C \Vdash t : \chi \otimes C * C)$. Let $w \in W$, $k \in \mathbb{N}$, $r \in \mathit{UPred}(\mathit{Heap})$ and

$$(k,(\rho,h)) \in [\![\Gamma \otimes C * C]\!](w) * \iota^{-1}(w)(\mathit{emp}) * r = [\![\Gamma]\!](\iota([\![C]\!]) \circ w) * \iota^{-1}(\iota([\![C]\!]) \circ w)(\mathit{emp}) * r \,.$$

Since $[\![C]\!] \in \mathit{Cap}$ we can instantiate $\models (\Gamma \Vdash t : \chi)$ with the world $w' = \iota([\![C]\!]) \circ w$ to obtain $(k,(\rho(t),h)) \in \mathscr{E}([\![\chi]\!])(w') * r$. The latter is equivalent to $(k,(\rho(t),h)) \in \mathscr{E}([\![\chi \otimes C * C]\!])(w) * r$. □

B.23 Anti-frame rule

Our soundness proof of the anti-frame rule employs the technique of so-called commutative pairs. This idea had already been present in Pottier's syntactic proof sketch [9], and has been worked out in more detail in [12].

Lemma 23. *For all worlds* $w_0, w_1 \in W$, *there exist* $w_0', w_1' \in W$ *such that*

$$w_0' = \iota(\iota^{-1}(w_0) \otimes w_1'), \quad w_1' = \iota(\iota^{-1}(w_1) \otimes w_0'), \quad \textit{and} \quad w_0 \circ w_1' = w_1 \circ w_0' \,.$$

Proof. Fix $w_0, w_1 \in W$, and define a function F on $X \times X$ defined by

$$F(x_0', x_1') \;=\; \big(\iota(\iota^{-1}(w_0) \otimes x_1'),\, \iota(\iota^{-1}(w_1) \otimes x_0')\big) \,.$$

Then, F is contractive, since $\otimes$ is contractive in its right argument. Also, F restricts to a function on the non-empty and closed subset $W \times W$. Thus, by Banach's fixpoint theorem, there exists a unique fixpoint w_0' and w_1' of F. This means that

$$w_0' \;=\; \iota(\iota^{-1}(w_0) \otimes w_1') \quad \text{and} \quad w_1' \;=\; \iota(\iota^{-1}(w_1) \otimes w_0'). \tag{43}$$

Note that these are the first two equalities claimed by this lemma. The remaining claim is $w_0 \circ w_1' = w_1 \circ w_0'$, and it can be proved as follows. Let $w \in X$.

$$\begin{aligned}
\iota^{-1}(w_0 \circ w_1')(w) &= \iota^{-1}(w_0)(w_1' \circ w) * \iota^{-1}(w_1')(w) && \text{(by definition of } \circ) \\
&= (\iota^{-1}(w_0) \otimes w_1')(w) * \iota^{-1}(w_1')(w) && \text{(by def. of } \otimes) \\
&= \iota^{-1}(w_0')(w) * (\iota^{-1}(w_1) \otimes w_0')(w) && \text{(by (43))} \\
&= \iota^{-1}(w_0')(w) * \iota^{-1}(w_1)(w_0' \circ w) && \text{(by def. of } \otimes) \\
&= \iota^{-1}(w_1)(w_0' \circ w) * \iota^{-1}(w_0')(w) && \text{(by commutativity of } *) \\
&= \iota^{-1}(w_1 \circ w_0')(w) && \text{(by definition of } \circ).
\end{aligned}$$

Since w was chosen arbitrarily, we have $\iota^{-1}(w_0 \circ w_1') = \iota^{-1}(w_1 \circ w_0')$, and the claim follows from the injectivity of ι^{-1}. □

Lemma 24. *Suppose* $\models (\Gamma \otimes C \Vdash t : \chi \otimes C * C)$. *Then* $\models (\Gamma \Vdash t : \chi)$.

Proof. We prove $\models (\Gamma \Vdash t : \chi)$. Let $w \in W$, $k \in \mathbb{N}$, $r \in \mathit{UPred}(\mathit{Heap})$ and

$$(k,(\rho,h)) \in [\![\Gamma]\!](w) * \iota^{-1}(w)(\mathit{emp}) * r \,.$$

We must prove $(k,(\rho(t),h)) \in \mathscr{E}([\![\chi]\!])(w) * r$.

By Lemma 23, there exist worlds w_1, w_2 in W such that

$$w_1 = \iota(\iota^{-1}(w) \otimes w_2), \quad w_2 = \iota(\llbracket C \rrbracket \otimes w_1) \quad \text{and} \quad \iota(\llbracket C \rrbracket) \circ w_1 = w \circ w_2 \,. \tag{44}$$

First, we find a superset of the precondition $\llbracket \Gamma \rrbracket(w) * \iota^{-1}(w)(emp) * r$ in the assumption above. Specifically, we replace the first two $*$-conjuncts in the precondition by bigger sets as follows:

$$\begin{aligned}
\llbracket \Gamma \rrbracket(w) &\subseteq \llbracket \Gamma \rrbracket(w \circ w_2) && \text{(by monotonicity of } \llbracket \Gamma \rrbracket \text{ and } w_2 \in W) \\
&= \llbracket \Gamma \rrbracket(\iota(\llbracket C \rrbracket) \circ w_1) && \text{(since } \iota(\llbracket C \rrbracket) \circ w_1 = w \circ w_2) \\
&= \llbracket \Gamma \otimes C \rrbracket(w_1) && \text{(by definition of } \otimes).
\end{aligned}$$

$$\begin{aligned}
\iota^{-1}(w)(emp) &\subseteq \iota^{-1}(w)(emp \circ w_2) && \text{(by monotonicity of } \iota^{-1}(w) \text{ and } w_2 \in W) \\
&= \iota^{-1}(w)(w_2 \circ emp) && \text{(since } emp \text{ is the unit)} \\
&= (\iota^{-1}(w) \otimes w_2)(emp) && \text{(by definition of } \otimes) \\
&= \iota^{-1}(w_1)(emp) && \text{(since } w_1 = \iota(\iota^{-1}(w) \otimes w_2))
\end{aligned}$$

Thus, we have that

$$(k, (\rho, h)) \in \llbracket \Gamma \rrbracket(w) * \iota^{-1}(w)(emp) * r \;\subseteq\; \llbracket \Gamma \otimes C \rrbracket(w_1) * \iota^{-1}(w_1)(emp) * r \,. \tag{45}$$

By the assumed validity of the judgement $\Gamma \otimes C \Vdash t : \chi \otimes C * C$, (45) entails

$$(k, (\rho(t), h)) \in \mathscr{E}(\llbracket \chi \otimes C * C \rrbracket)(w_1) * r \,. \tag{46}$$

We need to show that $(k, (\rho(t), h)) \in \mathscr{E}(\llbracket \chi \rrbracket)(w) * r$, so assume $(\rho(t) \,|\, h) \longmapsto^j (t' \,|\, h')$ for some $j \leq k$ such that $(t' \,|\, h')$ is irreducible. By (46) this means

$$(k-j, (t', h')) \in \bigcup_{w'} \llbracket \chi \otimes C * C \rrbracket(w_1 \circ w') * \iota^{-1}(w_1 \circ w')(emp) * r \,, \tag{47}$$

Note that we have

$$\begin{aligned}
\llbracket \chi \otimes C * C \rrbracket(w_1 \circ w') * \iota^{-1}(w_1 \circ w')(emp) &= \llbracket \chi \rrbracket(\iota(\llbracket C \rrbracket) \circ w_1 \circ w') * \llbracket C \rrbracket(w_1 \circ w') * \iota^{-1}(w_1 \circ w')(emp) \\
&= \llbracket \chi \rrbracket(\iota(\llbracket C \rrbracket) \circ w_1 \circ w') * \iota^{-1}(\iota(\llbracket C \rrbracket) \circ w_1 \circ w')(emp) \\
&= \llbracket \chi \rrbracket(w \circ w'') * \iota^{-1}(w \circ w'')(emp)
\end{aligned}$$

for $w'' \stackrel{def}{=} w_2 \circ w'$, since $w \circ w_2 = \iota(\llbracket C \rrbracket) \circ w_1$. Thus, (47) entails

$$(k-j, (t', h')) \in \bigcup_{w''} \llbracket \chi \rrbracket(w \circ w'') * \iota^{-1}(w \circ w'')(emp) * r \,,$$

and we are done. □