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

Jan Schwinghammer, Lars Birkedal, and Kristian Støvring

[1] Saarland University
[2] IT University of Copenhagen
[3] DIKU, University of Copenhagen

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 the first sound model for Charguéraud and Pottier's type and capability system including both frame and anti-frame rules. The model is a possible worlds model based on the operational semantics and step-indexed heap relations, and the worlds are constructed as a recursively defined predicate on a recursively defined metric space.
We also extend the model to account for Pottier's *generalized* frame and anti-frame rules, where invariants are generalized to *families* of invariants indexed over pre-orders. This generalization enables reasoning about some well-bracketed as well as (locally) monotonic uses of local state.

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 [2, 8, 9]. In this context, Pottier [14] presented the anti-frame rule as a proof rule for hiding invariants on encapsulated state: 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 invariants on local state include simpler interface specifications, simpler reasoning about client code, and fewer restrictions on the procedure's use because potential aliasing is reduced. 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.

Essentially, the frame and anti-frame rules exploit the fact that programs cannot access non-local state directly. However, in an ML-like language with higher-order procedures and the possibility of call-backs, the dependencies on non-local state can be complex; consequently, the soundness of frame and anti-frame rules is anything but obvious.

Pottier [14] sketched a soundness proof for the anti-frame rule by a progress and preservation argument, which rests on assumptions about the existence of certain recursively defined types and capabilities. (He has since formalized the details in Coq.) More recently, Birkedal et al. [6] developed a step-indexed model of Charguéraud and Pottier's type and capability system with higher-order frame rules, but without the anti-frame rule. This was a Kripke model in which capabilities are viewed as assertions (on heaps) that are indexed over recursively defined worlds: intuitively, these worlds are used to represent the invariants that have been added by the frame rules.

Proving soundness of the anti-frame rule requires a refinement of this idea, as one needs to know that additional invariants do not invalidate the invariants on local state which have been hidden by the anti-frame rule. This requirement can be formulated in terms of a monotonicity condition for the world-indexed assertions, using an order on the worlds that is induced by invariant extension, i.e., the addition of new invariants [17]. (The fact that ML-style untracked references can be encoded from strong references with the anti-frame rule [14] also indicates that a monotonicity condition is required: Kripke models of ML-style references involve monotonicity in the worlds [7, 1].) More precisely, in the presence of the anti-frame rule, it turns out that the recursive domain equation for the worlds involves monotonic functions with respect to an order relation on worlds, and that this order is specified using the isomorphism of the recursive world solution itself. This circularity means that standard existence theorems, in particular the one used in [6], cannot be applied to define the worlds. Thus Schwinghammer et al. [17], who considered a separation logic variant of the anti-frame rule for a simple language (without higher-order functions, and untyped), had to give the solution to a similar recursive domain equation by a laborious inverse-limit construction.

In the present paper we develop a new model of Charguéraud and Pottier's system, which can also be used to show soundness of the anti-frame rule. Moreover, we show how to extend our model to show soundness of Pottier's *generalized* frame and anti-frame rules, which allow hiding of *families* of invariants [15]. The new model is a non-trivial extension of the earlier work because, as pointed out above, the anti-frame rule is the "source" of a circular monotonicity requirement.

Our approach can loosely be described as a metric space analogue of Pitts' approach to relational properties of domains [13] and thus 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 as a fixed point of a suitable operator. The resulting subset of monotonic 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 [17] suitably restrict to the recursively defined subset of hereditarily monotonic functions. This results in a considerably simpler model construction than the earlier one in *loc. cit.* We show that our approach scales by extending the model to also allow for hiding of families of invariants, and using it to prove the soundness of Pottier's generalized frame and anti-frame rules [15].

Contributions. In summary, the contributions of this paper are (1) the development of a considerably simpler model of recursive worlds for showing the soundness of the anti-frame rule; (2) the use of this model to give the first soundness proof of the anti-frame rule in the expressive type and capability system of Charguéraud and Pottier; and (3) the extension of the model to include hiding of families of invariants, and showing the soundness of generalized frame and anti-frame rules. Moreover, at a conceptual level, we augment our earlier approach to constructing (step-indexed) recursive possible worlds based on a programming language's operational semantics via metric spaces [6] by a further tool, *viz.*, defining worlds as recursive subsets of recursive metric spaces.

Outline. In the next section we give a brief overview of Charguéraud and Pottier's type and capability system [8, 14] 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. The worlds thus constructed are then used (Section 4) to give a model of the type and capability system. Finally, in Section 5 we show how to extend the model to also prove soundness of the generalized frame and anti-frame rules.

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 [8, 14]:

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

Here, the term $\mathsf{fun}\, f(x){=}t$ stands for the recursive procedure f with body t, and locations l range over a countably infinite set Loc. The operational semantics is given by a relation $(t \,|\, h) \longmapsto (t' \,|\, h')$ 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 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*, as summarized in Figure 1. A capability C describes a heap property, much like the assertions of a Hoare-style program logic. For instance, $\{\sigma : \mathsf{ref\ int}\}$ 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

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 contexts	$\Delta ::= \Delta \otimes C \mid \varnothing \mid \Delta, x{:}\tau$
Linear contexts	$\Gamma ::= \Gamma \otimes C \mid \varnothing \mid \Gamma, x{:}\chi \mid \Gamma * C$

Fig. 1. Capabilities and types

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. (We do not consider existential types in this paper.) *Memory types* (and the subset of computation types χ) 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 both the 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 by a procedure call. We allow recursive capabilities, value types, and memory types, resp., provided the recursive definition is formally contractive [11], i.e., the recursion must go through a type constructor such as $\times$ or $\to$.

Since Charguéraud and Pottier's system tracks aliasing, so-called strong (i.e., non-type preserving) updates are permitted: a possible type for such an update operation is $\forall\sigma,\sigma'.([\sigma] \times [\sigma']) * \{\sigma : \mathsf{ref}\,\tau\} \to \mathbf{1} * \{\sigma : \mathsf{ref}\,[\sigma']\}$. Here, the argument to the procedure is a pair consisting of a location (named σ) and the value to be stored (named σ'), and the location is assumed to be allocated in the initial heap (and store a value of some type τ). The result of the procedure is unit, but as a side-effect σ' will be stored at the location σ.

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 [5]. The 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 (essentially dual) anti-frame rule:

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

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 [5], 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 that is guaranteed to hold whenever control passes from t to the context and back, expressed by $\cdot \otimes C$.

Pottier [14] illustrates the anti-frame rule by a number of applications. One of these is a fixed-point combinator implemented by means of "Landin's knot", i.e., recursion through heap. Every time the combinator is called with a functional $f : (\chi_1 \multimap \chi_2) \to (\chi_1 \multimap \chi_2)$, a new reference cell σ is allocated in order to set up the recursion required for the resulting fixed point *fix f*. Subsequent calls to *fix f* still rely on this cell, and in Charguéraud and Pottier's system this is reflected in the type $(\chi_1 \multimap \chi_2) \otimes I$ of *fix f*, where the capability $I = \{\sigma : \mathsf{ref}\,(\chi_1 \multimap \chi_2) \otimes I\}$ describes the cell σ after it has been initialized. However, the anti-frame rule allows one to hide the existence of σ, and leads to a purely functional interface of the fixed point combinator. In particular, after hiding I, *fix f* has the much simpler type $(\chi_1 \multimap \chi_2)$, which means that we can reason about aliasing and type safety of programs that *use* the fixed-point combinator without considering the reference cells used internally by that combinator.

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 [16, 17]. That is, rather than interpreting a capability C directly as a set of heaps, we interpret it as a function $[\![C]\!] : W \to \mathit{Pred}(\mathit{Heap})$ that maps "invariants" from W to sets of heaps. Intuitively, invariant extension of C is then interpreted by applying $[\![C]\!]$ to the given invariant. In contrast, a simple interpretation of C as a set of heaps would not contain enough information to determine the meaning of every invariant extension of C.

The question is now what the set W of invariants should be. As the frame and anti-frame rules in (1) indicate, invariants are in fact arbitrary capabilities, so W

should be the set used to interpret capabilities. But, as we just saw, capabilities should be interpreted as functions from W to $\mathit{Pred}(\mathit{Heap})$. 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 [18]. 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 $d(x, y) \le \max\{d(x, z), d(z, y)\}$. 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 $d_2(f(x), f(y)) \le d_1(x, y)$ for all $x, y \in X_1$. It is *contractive* if there exists some $\delta < 1$ such that $d_2(f(x), f(y)) \le \delta \cdot d_1(x, y)$ for all $x, y \in X_1$. By the Banach fixed point theorem, every contractive function $f : X \to X$ on a complete and non-empty metric space (X, d) has a (unique) fixed point. 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 $x \overset{n}{=} y$ means that $d(x, y) \le 2^{-n}$. Each relation $\overset{n}{=}$ is an equivalence relation because of the ultrametric inequality; we refer to this relation as "n-equality." Since the distances are bounded by 1, $x \overset{0}{=} y$ always holds, and the n-equalities become finer as n increases. If $x \overset{n}{=} y$ 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 $(k, a) \in p$, $j \le k$ and $a \sqsubseteq b$, then $(j, b) \in p$. We write $\mathit{UPred}(A)$ for the set of all such 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 $\mathit{UPred}(\mathit{Heap})$, ordered by subset inclusion, we have a complete Heyting BI algebra structure [4]. Below we only need the separating conjunction and its unit I, given by

$$p_1 * p_2 \;=\; \{(k,h) \mid \exists h_1, h_2.\; h = h_1 {\cdot} h_2 \;\wedge\; (k,h_1) \in p_1 \wedge\; (k,h_2) \in p_2\}$$

and $I = \mathbb{N} \times \mathit{Heap}$. Still, this observation on $\mathit{UPred}(\mathit{Heap})$ suggests that Pottier and Charguéraud's system could be extended to a full-blown program logic.

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 [3]. In particular, with regard to the domain equation (2) above:

Theorem 1. *There exists a unique (up to isomorphism) metric space* $(X,d) \in \mathit{CBUlt}_{ne}$ *and an isomorphism* ι *from* $\frac{1}{2} \cdot X \to \mathit{UPred}(\mathit{Heap})$ *to* X.

Using the pointwise lifting of separating conjunction to $\frac{1}{2}{\cdot}X \to \mathit{UPred}(\mathit{Heap})$ we define a *composition operation* on X. More precisely, $\circ : X \times X \to X$ 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 it can be defined by an easy application of Banach's fixed point theorem as in [16]. This operation 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; the isomorphism ι^{-1} lets us view p, q and $p \circ q$ as $\mathit{UPred}(\mathit{Heap})$-valued functions on $\frac{1}{2} \cdot X$. One can show that this operation $\circ$ is associative and has a left and right unit given by $\mathit{emp} = \iota(\lambda w.I)$; thus $(X, \circ, \mathit{emp})$ is a monoid in CBUlt_{ne}.

Then, using $\circ$ we define an *extension operation* $\otimes : Y^{(1/2 \cdot X)} \times X \to Y^{(1/2 \cdot X)}$ for any $Y \in \mathit{CBUlt}_{ne}$ by $(f \otimes x)(x') = f(x \circ x')$. Not going into details here, let us remark that $\otimes$ is the semantic counterpart to the syntactic invariant extension, and thus plays a key role in the model. 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 $X \;\cong\; 1/2 \cdot X \;\to\; \mathit{UPred}(\mathit{Heap})$ 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 $\mathit{UPred}(\mathit{Heap})$:

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

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 [17]. 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 $\mathcal{R}$ be the collection of all non-empty and closed relations $R \subseteq X$. We set

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

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

We make some further observations that follow from the properties of n-equality on X. First, $R \subseteq S$ implies $R_{[n]} \subseteq S_{[n]}$ for any $R, S \in \mathcal{R}$. 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 $\mathcal{R}$. As a consequence, we have $R \subseteq \ldots \subseteq R_{[n]} \subseteq \ldots \subseteq R_{[1]} \subseteq R_{[0]}$. By the 1-boundedness of X, $R_{[0]} = X$ for all $R \in \mathcal{R}$. Finally, $R = S$ if and only if $R_{[n]} = S_{[n]}$ for all $n \in \mathbb{N}$.

Proposition 2. *Let $d : \mathcal{R} \times \mathcal{R} \to \mathbb{R}$ be defined by $d(R, S) = \inf\{2^{-n} \mid R_{[n]} = S_{[n]}\}$. Then $(\mathcal{R}, d)$ 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 $R \in \mathcal{R}$.*

We will now define the set of hereditarily monotonic functions W as a recursive predicate on the space X. Let the function $\varPhi : \mathcal{P}(X) \to \mathcal{P}(X)$ on subsets of X be given by $\varPhi(R) = \{\iota(p) \mid \forall x, x_0 \in R.\ p(x) \subseteq p(x \circ x_0)\}$.

Lemma 3. *$\varPhi$ restricts to a contractive function on $\mathcal{R}$: if $R \in \mathcal{R}$ then $\varPhi(R)$ is non-empty and closed, and $R \overset{n}{=} S$ implies $\varPhi(R) \overset{n+1}{=} \varPhi(S)$.*

While the proof of this lemma is not particularly difficult, we include it here to illustrate the kind of reasoning that is involved.

Proof. It is clear that $\varPhi(R) \neq \emptyset$ since $\iota(p) \in \varPhi(R)$ for every constant function p from $\frac{1}{2} \cdot X$ to $\mathit{UPred}(\mathit{Heap})$. Limits of Cauchy chains in $\frac{1}{2} \cdot X \to \mathit{UPred}(\mathit{Heap})$ are given pointwise, hence $(\lim_n p_n)(x) \subseteq (\lim_n p_n)(x \circ x_0)$ holds for all Cauchy chains $(p_n)_{n\in\mathbb{N}}$ in $\varPhi(R)$ and all $x, x_0 \in R$. This proves $\varPhi(R) \in \mathcal{R}$.

We now show that $\varPhi$ is contractive. To this end, let $n \geq 0$ and assume $R \overset{n}{=} S$. Let $\iota(p) \in \varPhi(R)_{[n+1]}$. We must show that $\iota(p) \in \varPhi(S)_{[n+1]}$. By definition of the closure operation there exists $\iota(q) \in \varPhi(R)$ such that p and q are $(n+1)$-equal. Set $r(w) = q(w)_{[n+1]}$. Then r and p are also $(n+1)$-equal, hence it suffices to show that $\iota(r) \in \varPhi(S)$. To establish the latter, let $w_0, w_1 \in S$ be arbitrary. By the assumption that R and S are n-equal there exist elements $w_0', w_1' \in R$ such that $w'_0 \overset{n}{=} w_0$ and $w'_1 \overset{n}{=} w_1$ in holds X, or equivalently, such that w'_0 and w_0 as well as w'_1 and w_1 are $(n{+}1)$-equal in $\frac{1}{2} \cdot X$. By the non-expansiveness of $\circ$, this implies that also $w'_0 \circ w'_1$ and $w_0 \circ w_1$ are $(n{+}1)$-equal in $\frac{1}{2} \cdot X$. Since

$$q(w_0) \overset{n+1}{=} q(w'_0) \subseteq q(w'_0 \circ w'_1) \overset{n+1}{=} q(w_0 \circ w_1)$$

holds by the non-expansiveness of q and the assumption that $\iota(q) \in \Phi(R)$, we obtain the required inclusion $r(w_0) \subseteq r(w_0 \circ w_1)$ by definition of r. $\square$

By Proposition 2 and the Banach theorem we can now define the hereditarily monotonic functions W as the uniquely determined fixed point of Φ, for which

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

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

$$\begin{aligned}&\tfrac{1}{2} \cdot W \to_{mon} \mathit{UPred}(A)\\ &\qquad\qquad \overset{def}{=} \{p : \tfrac{1}{2} \cdot X \to \mathit{UPred}(A) \mid \forall w_1, w_2 \in W.\; p(w_1) \subseteq p(w_1 \circ w_2)\} \;.\end{aligned}$$

Then, for our particular application of interest, we also have to ensure that all the operations restrict appropriately (*cf.* Section 4 below). Here, as a first step, we show 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 $p \otimes w$.

Lemma 4. *For all $n \in \mathbb{N}$, if $w_1, w_2 \in W$ then $w_1 \circ w_2 \in W_{[n]}$. In particular, since $W = \bigcap_n W_{[n]}$ it follows that $w_1, w_2 \in W$ implies $w_1 \circ w_2 \in W$.*

Proof. The proof is by induction on n. The base case is immediate as $W_{[0]} = X$. 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 $w'_1 \in W$, that w'_1 and w_1 are n-equal, and that w'_1 is such that n-equality of w, w' in $\frac{1}{2} \cdot X$ already implies $\iota^{-1}(w'_1)(w) = \iota^{-1}(w'_1)(w')$. Since w'_1 and w_1 are n-equivalent, the non-expansiveness of the composition operation implies $w_1 \circ w_2 \overset{n}{=} w'_1 \circ w_2$. Thus it suffices to show that $w'_1 \circ w_2 \in W = \Phi(W)$. To see this, let $w, w_0 \in W$ be arbitrary, and note that by induction hypothesis we have $w_2 \circ w \in W_{[n-1]}$. This means that there exists $w' \in W$ such that $w' \overset{n}{=} w_2 \circ w$ holds in $\frac{1}{2} \cdot X$, hence

$$\begin{aligned}
\iota^{-1}(w'_1 \circ w_2)(w) &= \iota^{-1}(w'_1)(w_2 \circ w) * \iota^{-1}(w_2)(w) && \text{by definition of } \circ\\
&= \iota^{-1}(w'_1)(w') * \iota^{-1}(w_2)(w) && \text{by } w' \overset{n}{=} w_2 \circ w\\
&\subseteq \iota^{-1}(w'_1)(w' \circ w_0) * \iota^{-1}(w_2)(w \circ w_0) && \text{by hereditariness}\\
&= \iota^{-1}(w'_1)((w_2 \circ w) \circ w_0) * \iota^{-1}(w_2)(w \circ w_0) && \text{by } w' \overset{n}{=} w_2 \circ w\\
&= \iota^{-1}(w'_1 \circ w_2)(w \circ w_0) && \text{by definition of } \circ.
\end{aligned}$$

Since w, w_0 were chosen arbitrarily, this calculation establishes $w'_1 \circ w_2 \in W$. $\square$

4 Step-indexed 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 $T_1, T_2 \in 1/2 \cdot X \to \mathit{UPred}(\mathit{Val} \times \mathit{Heap})$, $T_1 \to T_2$ in $\frac{1}{2} \cdot X \to \mathit{UPred}(\mathit{Val})$ is defined on $x \in X$ as

$$\begin{aligned} \{(k, \mathsf{fun}\, f(y){=}t) \mid \forall j < k.\ \forall w \in W.\ \forall r \in \mathit{UPred}(\mathit{Heap}). \\ \forall v, h.\ (j, (v, h)) \in T_1(x \circ w) * \iota^{-1}(x \circ w)(\mathit{emp}) * r \ \Rightarrow \\ (j, (t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathcal{E}(T_2 * r)(x \circ w)\} \ , \end{aligned} \tag{4}$$

where $\mathcal{E}(T)$ 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: $(k, (t, h)) \in \mathcal{E}(T)(x)$ if and only if for all $j \le k, t', h'$,

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

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 $T_1 \to T_2$ 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 $T_1 \to T_2 \le T_1 * C \to T_2 * C$ in the type system. The existential quantification over w', in the definition of $\mathcal{E}$, 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 $(T_1 \to T_2)(x)$ is uniform. There are three reasons 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 $T_1 \to T_2$ as a function $1/2 \cdot X \to \mathit{UPred}(\mathit{Val})$. Moreover, it lets us prove the typing rule for *recursive* functions by induction on k. Finally, it means that $\to$ is a contractive type constructor, which justifies the formal contractiveness assumption about arrow types that we made earlier. Intuitively, the use of $j < k$ for the arguments suffices since application consumes a step.

The function type constructor, as well as all the other type and capability constructors, restrict to Cap, VT and MT, respectively. With their help it becomes straightforward to define the interpretation $\llbracket C \rrbracket_\eta$ and $\llbracket \tau \rrbracket_\eta$ of capabilities and types, given 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 $\eta(\beta) \in \mathit{MT}$. The type equivalences can then be verified with respect to this interpretation. We state this for the case of arrow types:

Lemma 5. *Let T_1, T_2 non-expansive functions from $\frac{1}{2}\cdot X$ to $\mathit{UPred}(\mathit{Val} \times \mathit{Heap})$.*

1. *$T_1 \to T_2$ is non-expansive, and $(T_1 \to T_2)(x)$ is uniform for all $x \in X$.*
2. *$T_1 \to T_2 \in \mathit{VT}$.*
3. *The assignment of $T_1 \to T_2$ to T_1, T_2 is contractive.*
4. *Let $c \in \mathit{Cap}$ and $w \overset{\mathit{def}}{=} \iota(c)$. Then $(T_1 \to T_2) \otimes w = (T_1 \otimes w * c) \to (T_2 \otimes w * c)$.*

Recall that there are two kinds of typing judgments, one for typing of values and the other for the typing of expressions. The semantics of a value judgement simply establishes truth with respect to all worlds w, environments η, and $k \in \mathbb{N}$:

$$\models (\Delta \vdash v : \tau) \overset{\mathit{def}}{\iff} \forall \eta.\ \forall w.\ \forall k.\ \forall \rho.\ (k, \rho) \in \llbracket \Delta \rrbracket_\eta w \ \Rightarrow\ (k, \rho(v)) \in \llbracket \tau \rrbracket_\eta w\ .$$

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 $r \in \mathit{UPred}(\mathit{Heap})$ and an existential quantification over $w' \in W$ through the use of $\mathcal{E}$:

$$\models\ (\Gamma \Vdash t : \chi) \overset{\mathit{def}}{\iff} \forall \eta.\ \forall w.\ \forall k.\ \forall r \in \mathit{UPred}(\mathit{Heap}).$$
$$\forall \rho, h.\ (k, (\rho, h)) \in \llbracket \Gamma \rrbracket_\eta w * \iota^{-1}(w)(\mathit{emp}) * r \ \Rightarrow\ (k, (\rho(t), h)) \in \mathcal{E}(\llbracket \chi \rrbracket_\eta * r)(w).$$

Theorem 6 (Soundness). *If $\Delta \vdash v : \tau$ then $\models (\Delta \vdash v : \tau)$, and if $\Gamma \Vdash t : \chi$ then $\models (\Gamma \Vdash t : \chi)$.*

To prove the theorem, we show that each typing rule preserves the truth of judgements. Detailed proofs for the shallow and deep frame rules are included in the appendix. Here, we consider the anti-frame rule. Its proof employs so-called commutative pairs [14, 17], a property expressed by the following lemma.

Lemma 7. *For all worlds $w_0, w_1 \in W$, there exist $w_0', w_1' \in W$ such that*

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

Lemma 8 (Soundness of the anti-frame rule). *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 $w \in W$, η an environment, $r \in \mathit{UPred}(\mathit{Heap})$ and

$$(k, (\rho, h)) \in \llbracket \Gamma \rrbracket_\eta (w) * \iota^{-1}(w)(\mathit{emp}) * r\ .$$

We must prove $(k, (\rho(t), h)) \in \mathcal{E}(\llbracket \chi \rrbracket_\eta * r)(w)$. By Lemma 7,

$$w_1 = \iota(\iota^{-1}(w) \otimes w_2), \quad w_2 = \iota(\llbracket C \rrbracket_\eta \otimes w_1) \quad \text{and} \quad \iota(\llbracket C \rrbracket_\eta) \circ w_1 = w \circ w_2 \qquad (5)$$

holds for some worlds w_1, w_2 in W.

First, we find a superset of the precondition $\llbracket \Gamma \rrbracket_\eta (w) * \iota^{-1}(w)(\mathit{emp}) * r$ in the assumption above, replacing the first two $*$-conjuncts as follows:

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

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

Thus, by the monotonicity of separating conjunction, we have that

$$(k, (\rho, h)) \in \llbracket \Gamma \rrbracket_\eta (w) * \iota^{-1}(w)(\mathit{emp}) * r \;\subseteq\; \llbracket \Gamma \otimes C \rrbracket_\eta (w_1) * \iota^{-1}(w_1)(\mathit{emp}) * r \,. \quad (6)$$

By the assumed validity of the judgement $\Gamma \otimes C \Vdash t : \chi \otimes C * C$, (6) entails

$$(k, (\rho(t), h)) \in \mathcal{E}(\llbracket \chi \otimes C * C \rrbracket_\eta * r)(w_1) \,. \quad (7)$$

We need to show that $(k, (\rho(t), h)) \in \mathcal{E}(\llbracket \chi \rrbracket_\eta * r)(w)$, so assume $(\rho(t) \mid h) \longmapsto^j (t' \mid h')$ for some $j \leq k$ such that $(t' \mid h')$ is irreducible. From (7) we then obtain

$$(k-j, (t', h')) \in \textstyle\bigcup_{w'} \llbracket \chi \otimes C * C \rrbracket_\eta (w_1 \circ w') * \iota^{-1}(w_1 \circ w')(\mathit{emp}) * r \,. \quad (8)$$

Now observe that we have

$$\begin{array}{l}
\llbracket \chi \otimes C * C \rrbracket_\eta (w_1 \circ w') * \iota^{-1}(w_1 \circ w')(\mathit{emp}) \\
\quad = \llbracket \chi \rrbracket_\eta (\iota(\llbracket C \rrbracket_\eta) \circ w_1 \circ w') * \llbracket C \rrbracket_\eta (w_1 \circ w') * \iota^{-1}(w_1 \circ w')(\mathit{emp}) \\
\quad = \llbracket \chi \rrbracket_\eta (\iota(\llbracket C \rrbracket_\eta) \circ w_1 \circ w') * \iota^{-1}(\iota(\llbracket C \rrbracket_\eta) \circ w_1 \circ w')(\mathit{emp}) \\
\quad = \llbracket \chi \rrbracket_\eta (w \circ w'') * \iota^{-1}(w \circ w'')(\mathit{emp})
\end{array}$$

for $w'' \overset{\mathit{def}}{=} w_2 \circ w'$, since $w \circ w_2 = \iota(\llbracket C \rrbracket_\eta) \circ w_1$. Thus, (8) entails that $(k-j, (t', h'))$ is in $\bigcup_{w''} \llbracket \chi \rrbracket_\eta (w \circ w'') * \iota^{-1}(w \circ w'')(\mathit{emp}) * r$, and we are done. $\square$

5 Generalized Frame and Anti-frame Rules

The frame and anti-frame rules allow for hiding of *invariants*. However, to hide uses of local state, say for a function, it is, in general, not enough only to allow hiding of global invariants that are preserved across arbitrary sequences of calls and returns. For instance, consider the function f with local reference cell r:

$$\mathsf{let}\ r = \mathsf{ref}\ 0\ \mathsf{in}\ \mathsf{fun}\, f(g){=}(\mathit{inc}(r); g(); \mathit{dec}(r)) \quad (9)$$

If we write $\mathsf{int}\ n$ for the singleton integer type containing n, we may wish to hide the capability $I = \{\sigma : \mathsf{ref}\,(\mathsf{int}\ 0)\}$ to capture the intuition that the cell $r : [\sigma]$

stores 0 upon termination. However, there could well be re-entrant calls to f and $\{\sigma : \mathsf{ref}\,(\mathsf{int}\ 0)\}$ is not an invariant for those calls.

Thus Pottier [15] proposed two extensions to the anti-frame rule that allows for hiding of families of invariants. The first idea is that each invariant in the family is a *local* invariant that holds for one level of the recursive call of a function. This extension allows us to hide "well-bracketed" [10] uses of local state. For instance, the $\mathbb{N}$-indexed family of invariants $I\,n = \{\sigma : \mathsf{ref}\,(\mathsf{int}\ n)\}$ can be used for (9); see the examples in [15]. The second idea is to allow each local invariant to *evolve* in some monotonic fashion; this allows us to hide even more uses of local state. The idea is related to the notion of evolving invariants for local state in recent work on reasoning about contextual equivalence [1, 10]. (Space limitations preclude us from including examples; please see [15] for examples.)

In summary, we want to allow the hiding of a family of capabilities $(I\,i)_{i\in\kappa}$ indexed over a preordered set $(\kappa, \leq)$. The preorder is used to capture that the local invariants can evolve in a monotonic fashion, as expressed in the new definition of the action of $\otimes$ on function types (note that I on the right-hand side of $\otimes$ now has kind $\kappa \to \textsc{cap}$):

$$(\chi_1 \to \chi_2) \otimes I \;=\; \forall i.\ \big((\chi_1 \otimes I) * I\,i \to \exists j \geq i.\ ((\chi_2 \otimes I) * I\,j)\big)$$

Observe how this definition captures the intuitive idea: if the invariant $I\,i$ holds when the function is called then, upon return, we know that an invariant $I\,j$ (for $j \in \kappa$, $j \geq i$) holds. Different recursive calls may use different local invariants due to the quantification over i. The generalized frame and anti-frame rules are:

$$[GF]\frac{\Gamma \Vdash t : \chi}{\Gamma \otimes I * I\,i \Vdash t : \exists j \geq i.\,(\chi \otimes I) * I\,j} \qquad [GAF]\frac{\Gamma \otimes I \Vdash t : \exists i.\,(\chi \otimes I) * I\,i}{\Gamma \Vdash t : \chi}$$

We now show how to extend our model of the type and capability calculus to accomodate hiding of such more expressive families of invariants. Naturally, the first step is to refine our notion of world, since the worlds are used to describe hidden invariants.

Generalized worlds and generalized world extension. Suppose $\mathcal{K}$ is a (small) collection of preordered sets. We write $\mathcal{K}^*$ for the finite sequences over $\mathcal{K}$, ε for the empty sequence, and use juxtaposition to denote concatenation. For convenience, we will sometimes identify a sequence $\alpha = \kappa_1, \ldots, \kappa_n$ over $\mathcal{K}$ with the preorder $\kappa_1 \times \cdots \times \kappa_n$. As in Section 3, we define the worlds for the Kripke model in two steps, starting from an equation without any monotonicity requirements: CBUlt_{ne} has all non-empty coproducts, and there is a unique solution to the two equations

$$X \cong \sum_{\alpha\in\mathcal{K}^*} X_\alpha\ , \quad X_{\kappa_1,\ldots,\kappa_n} = (\kappa_1 \times \cdots \times \kappa_n) \to (\tfrac{1}{2}\cdot X \to \mathit{UPred}(\mathit{Heap}))\ , \quad (10)$$

with isomorphism $\iota : \sum_{\alpha\in\mathcal{K}^*} X_\alpha \to X$ in CBUlt_{ne}, where each $\kappa \in \mathcal{K}$ is equipped with the discrete metric. Each X_α consists of the α-indexed families of (world-dependent) predicates so that, in comparison to Section 3, X consists of all these families rather than individual predicates.

The composition operation $\circ : X \times X \to X$ is now given by $x_1 \circ x_2 = \iota(\langle \alpha_1\alpha_2, p\rangle)$ where $\langle \alpha_i, p_i\rangle = \iota^{-1}(x_i)$, and where $p \in X_{\alpha_1\alpha_2}$ is defined by

$$p(i_1 i_2)(x) \;=\; p_1(i_1)(x_2 \circ x) * p_2(i_2)(x)\ .$$

for $i_1 \in \alpha_1$, $i_2 \in \alpha_2$. That is, the combination of an α_1-indexed family p_1 and an α_2-indexed family p_2 is a family p over $\alpha_1\alpha_2$, but there is no interaction between the index components i_1 and i_2: they concern disjoint regions of the heap.

From here on we can proceed essentially as in Section 3: The composition operation can be shown associative, with a left and right unit given by $emp = \iota(\langle \varepsilon, \lambda\_, \_.I\rangle)$. For $f : \frac{1}{2}\cdot X \to Y$ the extension operation $(f \otimes x)(x') = f(x \circ x')$ is also defined as before (but with respect to the solution (10) and the new $\circ$ operation). We then carve out from X the subset of hereditarily monotonic functions W, which we again obtain as fixed point of a contractive function on the closed and non-empty subsets of X. Let us write $\sim$ for the (recursive) partial equivalence relation on X where $\iota(\langle \alpha_1\alpha_2, p\rangle) \sim \iota(\langle \alpha_2\alpha_1, q\rangle)$ holds if $p(i_1 i_2)(x_1) = q(i_2 i_1)(x_2)$ for all $i_1 \in \alpha_1$, $i_2 \in \alpha_2$ and $x_1 \sim x_2$. Then $w \in W$ iff $w \sim w$ and

$$\exists \alpha, p.\ w = \iota\langle \alpha, p\rangle \ \wedge\ \forall i \in \alpha. \forall w_1, w_2 \in W.\ p(i)(w_1) \subseteq p(i)(w_1 \circ w_2)\ .$$

Finally, the proof of Lemma 4 can be adapted to show that the operation $\circ$ restricts to the subset W.

Semantics of capabilities and types. The definition of function types changes as follows: given $x \in X$, $(k, \mathsf{fun}\, f(y){=}t) \in (T_1 \to T_2)(x)$ if and only if

$$\begin{array}{l}\forall j < k.\ \forall w \in W \text{ where } \iota^{-1}(x \circ w) = \langle \alpha, p\rangle.\ \forall r \in \mathit{UPred}(\mathit{Heap}).\ \forall i \in \alpha.\ \forall v, h.\\ (j, (v, h)) \in T_1(x \circ w) * p(i)(\mathit{emp}) * r \ \Rightarrow\\ \qquad (j, t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathcal{E}(T_2 * r, x \circ w, i)\ ,\end{array}$$

where the extension to expressions now depends on $i \in \alpha$: $(k, t) \in \mathcal{E}(T, x, i)$ if

$$\begin{array}{l}\forall j \le k, t', h'.\ (t \,|\, h) \longmapsto^{j} (t' \,|\, h') \ \wedge\ (t' \,|\, h') \text{ irreducible}\\ \qquad\qquad \Rightarrow\ (k - j, (t', h')) \in \bigcup_{w \in W,\, i_1 \in \alpha,\, i_2 \in \beta,\, i_1 \ge i} T(x \circ w) * q(i_1 i_2)(\mathit{emp})\end{array}$$

for $\langle \alpha\beta, q\rangle = \iota^{-1}(x \circ w)$.

Next, one proves the analogue of Lemma 5 which shows the well-definedness of $T_1 \to T_2$ and (a semantic variant of) the distribution axiom for generalized invariants: in particular, given $p \in \kappa \to \mathit{Cap}$ and setting $w \overset{def}{=} \iota(\langle \kappa, p\rangle)$,

$$(T_1 \to T_2) \otimes w = \forall_{i \in \kappa}\big((T_1 \otimes w) * p\, i) \to \exists_{j \ge i}((T_2 \otimes w) * p\, j)\big)$$

where $\forall$ and $\exists$ denote the pointwise intersection and union of world-indexed uniform predicates.

Once similar properties are proved for the other type and capability constructors (which do not change for the generalized invariants), we obtain:

Theorem 9 (Soundness). *The generalized frame and anti-frame rules* $[GF]$ *and* $[GAF]$ *are sound.*

In particular, this theorem shows that all the reasoning about the use of local state in the (non-trivial) examples considered by Pottier in [15] is sound.

6 Conclusion and Future Work

We have developed the first soundness proof of the anti-frame rule in the expressive type and capability system of Charguéraud and Pottier by constructing a Kripke model of the system. For our model, we have used a new approach to the construction of worlds by definining them as a recursive subset of a recursively defined metric space, thus avoiding a tedious explicit inverse-limit construction. We have shown that this approach scales, by also extending the model to show soundness of Pottier's generalized frame and anti-frame rules. Future work includes exploring some of the orthogonal extensions of the basic type and capability system: group regions [8] and fates & predictions [12].

References

1. Ahmed, A., Dreyer, D., Rossberg, A.: State-dependent representation independence. In: POPL (2009)
2. Ahmed, A., Fluet, M., Morrisett, G.: L3: A linear language with locations. Fundam. Inf. 77(4), 397–449 (2007)
3. America, P., Rutten, J.J.M.M.: Solving reflexive domain equations in a category of complete metric spaces. J. Comput. Syst. Sci. 39(3), 343–375 (1989)
4. Biering, B., Birkedal, L., Torp-Smith, N.: BI-hyperdoctrines, higher-order separation logic, and abstraction. ACM Trans. Program. Lang. Syst. 29(5) (2007)
5. Birkedal, L., Torp-Smith, N., Yang, H.: Semantics of separation-logic typing and higher-order frame rules for Algol-like languages. LMCS 2(5:1) (2006)
6. Birkedal, L., Reus, B., Schwinghammer, J., Støvring, K., Thamsborg, J., Yang, H.: Step-indexed Kripke models over recursive worlds. In: POPL (2011), to appear
7. Birkedal, L., Støvring, K., Thamsborg, J.: Realizability semantics of parametric polymorphism, general references, and recursive types. In: Proceedings of FOSSACS. pp. 456–470 (2009)
8. Charguéraud, A., Pottier, F.: Functional translation of a calculus of capabilities. In: Proceedings of ICFP. pp. 213–224 (2008)
9. Crary, K., Walker, D., Morrisett, G.: Typed memory management in a calculus of capabilities. In: Proceedings of POPL. pp. 262–275 (1999)
10. Dreyer, D., Neis, G., Birkedal, L.: The impact of higher-order state and control effects on local relational reasoning. In: Proceedings of ICFP (2010)
11. Pierce, B.C.: Types and Programming Languages. MIT Press (2002)
12. Pilkiewicz, A., Pottier, F.: The essence of monotonic state (July 2010), unpublished
13. Pitts, A.M.: Relational properties of domains. Inf. Comput. 127(2), 66–90 (1996)
14. Pottier, F.: Hiding local state in direct style: a higher-order anti-frame rule. In: Proceedings of LICS. pp. 331–340 (2008)
15. Pottier, F.: Generalizing the higher-order frame and anti-frame rules (July 2009), unpublished, available at `http://gallium.inria.fr/~fpottier`
16. Schwinghammer, J., Birkedal, L., Reus, B., Yang, H.: Nested Hoare triples and frame rules for higher-order store. In: Proceedings of CSL. pp. 440–454 (2009)
17. Schwinghammer, J., Yang, H., Birkedal, L., Pottier, F., Reus, B.: A semantic foundation for hidden state. In: Proceedings of FOSSACS. pp. 2–16 (2010)
18. Smyth, M.B.: Topology. In: Handbook of Logic in Computer Science, vol. 1. Oxford Univ. Press (1992)