A STEP-INDEXED KRIPKE MODEL OF HIDDEN STATE

JAN SCHWINGHAMMER, LARS BIRKEDAL, FRANÇOIS POTTIER, BERNHARD REUS, KRISTIAN STØVRING, AND HONGSEOK YANG

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 discuss the semantic foundations of frame and anti-frame rules, and present the first sound model for Charguéraud and Pottier's type and capability system including both of these rules. The model is a possible worlds model based on the operational semantics and step-indexed heap relations, and the worlds are given by 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 preorders. This generalization enables reasoning about some well-bracketed as well as (locally) monotone uses of local state.

1. Introduction

Information hiding, or *hidden state*, is one of the key design principles used by programmers in order to control the complexity of large-scale software systems. The idea is that an object (or function, or module) need not reveal in its interface the fact that it owns and maintains a private, mutable data structure. Hiding this internal invariant from the client has several beneficial effects. First, the complexity of the object's specification is slightly decreased. More importantly, the client is relieved from the need to thread the object's invariant through its own code. In particular, when an object has multiple clients, they are freed from the need to cooperate with one another in threading this invariant. Last, by hiding its internal state, the object escapes the restrictions on aliasing and ownership that are normally imposed on objects with mutable state.

The recently proposed anti-frame proof rule [21] enables hiding in the presence of higher-order store, i.e., memory cells containing (pointers to) procedures or code fragments. Thus, in combination with frame rules that allow the irrelevant parts of the state to be hidden during verification, the anti-frame rule can provide an important ingredient for modular, scalable program verification techniques. In this article, we study the semantic foundation of the anti-frame rule and give a soundness proof for it. Our proof involves an intricate recursive domain equation, and it helps identify some of the key ingredients for soundness.

1.1. **Information hiding with frame and anti-frame rules.**

Our results are in a line of work on logic-based approaches to information hiding. These approaches adopt a standard semantics of the programming language, and deal with information hiding on a logical basis, for instance by extending a Hoare calculus with special proof rules. These rules usually take the form of *frame rules* that allow the implementation of an object to ignore (hence implicitly preserve) some of the invariants provided by the context, and of *anti-frame rules*, which allow an object to hide its internal invariant from the context [10, 15, 21, 26].

It is worth emphasizing that *hiding* and *abstraction* (as studied, for instance, in separation logic [5, 14, 16, 17]) are distinct mechanisms, which may co-exist within a single program logic: Abstraction is often implemented in terms of assertion variables (called abstract predicates by Parkinson) that describe the private data structures of an object. These variables are exposed to a client, but their definitions are not, so that the object's internals are presented to the client in an abstract form. Hiding, on the other hand, conceals the object's internals completely.

In its simplest form, the frame rule [26] states that invariants R can be added to valid triples: if$\{P\}C\{Q\}$ is valid, then so is$\{P * R\}C\{Q * R\}$, where the separating conjunction $P * R$ indicates that P and R govern disjoint regions of the heap. In subsequent developments, the rule was extended to handle higher-order procedures [10, 15] and higher-order store [7, 27]. Moreover, it was argued that both extensions of the rule support information hiding: they allow one to hide the invariant of a module and to prove properties of clients, as long as the module is understood in continuation-passing style [15].

Thorough semantic analyses were required to determine the conditions under which these extensions of the frame rule are sound. Indeed, the soundness of these rules raises subtle issues. For instance, the frame rule for higher-order procedures turns out to be inconsistent with the conjunction rule, a standard rule of Hoare logic [10, 15]. Furthermore, seemingly innocent variants of the frame rule for higher-order store have been shown unsound [23, 27].

In the most recent development in this line of research, Pottier [21] proposed an anti-frame rule, which expresses the information hiding aspect of an object directly, instead of in continuation-passing style. Besides giving several extensive examples of how the anti-frame rule supports hidden state, Pottier argued that the anti-frame rule is sound by sketching a plausible syntactic argument. This argument, however, relied on several non-trivial assumptions about the existence of certain recursively defined types and recursively defined operations over types. In the present paper we justify these assumptions and give a complete soundness proof of Pottier's anti-frame rule.

1.2. **This paper.**

This article is an extended version of results that were presented in two papers at the FOSSACS 2010 and FOSSACS 2011 conferences [28, 29].

In the first of these papers we presented our results on a semantic foundation for the anti-frame rule in the context of a simple WHILE language with higher-order store, using a denotational semantics of the programming language. In the second paper we gave an alternative approach to constructing a model for the anti-frame rule and presented our results in the context of Charguéraud and Pottier's calculus of capabilities [11] that not only features higher-order store but also higher-order functions. In this latter paper we based the model on an operational semantics of the programming language, using the discovery that the metric approach to solving recursive possible world equations works both for denotationally- and operationally-based models [6].

In the present paper we describe our results in the context of the calculus of capabilities, using operational semantics. We detail both the original approach to constructing a model of the anti-frame rule from the FOSSACS 2010 paper (but adapted to operational semantics and step-indexing) and the alternative approach from the 2011 paper. We have chosen to use the capability calculus setup since Pottier has already shown how to reason about a range of applications with the anti-frame rule in this system [21]. Moreover, Pottier has also proposed generalized versions of the frame and anti-frame rules [22] for capabilities, and we show that our approach extends to these generalizations.

1.3. **Overview of the technical development.** 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.[1] More precisely, in the presence of the anti-frame rule, it turns out that the recursive domain equation for the worlds involves monotone 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 for the model without the anti-frame rule in [6], cannot be applied to define the worlds.

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 demonstrate how to extend our model to prove soundness of Pottier's *generalized* frame and anti-frame rules, which allow hiding of *families* of invariants [22]. 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. We present two alternative approaches that address this difficulty.

In the first approach, a solution to the recursive world equation is defined by an inverse-limit construction in a category of metric spaces; the approximants to this limit are defined simultaneously with suitably approximated order relations between worlds. This approach has originally been used by Schwinghammer et al. [29] for a separation logic variant of the anti-frame rule, for a simple WHILE language (untyped and without higher-order functions), and with respect to a denotational semantics of the programming language. In this article, the metrics that are employed to define the recursive worlds are linked to an operational semantics of the programming language instead, using the step-indexing idea [3, 6]. While the construction is laborious, it results in a set of worlds that evidently has the required properties.

The second approach can loosely be described as a metric space analogue of Pitts' approach to relational properties of domains [20] 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 monotone* functions. We show how to define this recursively specified subset as a fixed point of a suitable operator. While this second construction is considerably simpler than the inverse-limit construction, the resulting subset of monotone functions is, however, not a solution to the original recursive domain equation. Hence, we must verify that the semantic constructions that are used to justify the anti-frame rule restrict in a suitable way to the recursively defined subset of hereditarily monotone functions.

We show that our techniques scale, by extending the model to Pottier's generalized frame and anti-frame rules [22]. For this extension, capabilities denote

[1] The fact that ML-style untracked references can be encoded from strong references with the anti-frame rule [21] also indicates that a monotonicity condition is required: Kripke models of ML-style references involve monotonicity in the worlds [1, 8].

$$
\begin{aligned}
v &::= x \mid \langle\rangle \mid \mathsf{inj}^1\, v \mid \mathsf{inj}^2\, v \mid \langle v, v\rangle \mid \mathsf{fun}\, f(x){=}t \mid l \\
t &::= v \mid (v\, t) \mid \mathsf{case}(v, v, v) \mid \mathsf{proj}^1\, v \mid \mathsf{proj}^2\, v \mid \mathsf{ref}\, v \mid \mathsf{get}\, v \mid \mathsf{set}\, v
\end{aligned}
$$

Figure 1. Syntax

families of hereditarily monotone functions that are invariant under index reordering. The invariance property is expressed by considering a (recursively defined) partial equivalence relation on these families.

Outline. This paper is organised as follows. In the next section we give a brief overview of Charguéraud and Pottier's type and capability system with higher-order frame and anti-frame rules. In Section 3 we discuss the requirements that the frame and anti-frame rules place on the worlds of the Kripke model. Section 4 gives some background on metric spaces, and Sections 5 and 6 present the two approaches to constructing the recursive worlds for the possible worlds model. (Readers not interested in the details of these constructions can safely skip Sections 5 and 6.) The model is described and used to prove soundness of Charguéraud and Pottier's system in Section 7. In Section 8 we show how to extend the model to also prove soundness of the generalized frame and anti-frame rules.

2. A Calculus of Capabilities

Charguéraud and Pottier's calculus of capabilities uses (linear) capabilities and singleton types to track aliasing and ownership properties in a high-level, ML-like programming language [11]. Capabilities describe the shape of heap data structures, much like the assertions of separation logic. By introducing static names for values, the singleton types make it possible to refer within capabilities to the arguments and results of procedures.

We focus on the semantic foundations of the frame and anti-frame rules in this paper, and the exact details of the capability type system are therefore less important here: in this section we only give a brief overview of the calculus.[2] We refer to earlier work that motivates the design of the capability type system and gives detailed examples of its use [11, 19, 21, 22].

2.1. **Syntax and operational semantics.** The programming language that we consider is a standard call-by-value, higher-order language with general references, sum and product types, and polymorphic and recursive types. The grammar in Figure 1 gives the syntax of values and expressions, keeping close to the notation of [11]. 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 (Figure 2) 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.

2.2. **Types and capabilities.** Charguéraud and Pottier's type system uses *capabilities*, *value types*, and *computation types*. Figure 3 presents a subset of those. (The full syntax is given in Section 7.)

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 *singleton region*

[2] We only consider a fragment of the capability calculus. In particular, in this article we omit existential types and group regions.

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

FIGURE 2. Operational semantics

$$\begin{array}{ll}
\text{Capabilities} & C ::= \emptyset \mid \{\sigma : \mathsf{ref}\,\tau\} \mid C * C \mid \ \ldots \\
\text{Value types} & \tau ::= 1 \mid \mathsf{int} \mid \tau \times \tau \mid \tau + \tau \mid \chi \to \chi \mid [\sigma] \mid \ \ldots \\
\text{Computation types} & \chi ::= \tau \mid \chi * C \mid \exists\sigma.\chi \mid \ \ldots \\
\text{Value contexts} & \Delta ::= \varnothing \mid \Delta, x{:}\tau \mid \ \ldots \\
\text{Linear contexts} & \Gamma ::= \varnothing \mid \Gamma, x{:}\chi \mid \Gamma * C \mid \ \ldots
\end{array}$$

FIGURE 3. Capabilities and types

inhabited by one valid location that contains an integer value. More complex capabilities 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 like int, singleton types $[\sigma]$, and are closed under products, sums, and universal quantification. Computation types χ describe the result of computations. They include all types of the form $\exists\vec{\sigma}.\tau * C$, which describe both the value and the 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 as well as recursive value and computation types, provided the recursive definition is formally contractive [18], i.e., the recursion must go through one of the type constructors $+$, $\times$, $\to$, or ref .

Since Charguéraud and Pottier's system tracks aliasing, strong (i.e., not necessarily type preserving) updates can be admitted. A possible type for such an update operation is $\forall\sigma, \sigma'.([\sigma] \times [\sigma']) * \{\sigma : \mathsf{ref}\,\tau\} \to 1 * \{\sigma : \mathsf{ref}\,[\sigma']\}$. 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 the unit value $\langle\rangle$, but as a side-effect σ' will be stored at the location σ.

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. (Since values cannot be reduced, there is no need for pre- and postconditions in the value typing judgement.) Some of the inference rules that define the two typing judgements are given in Figure 4.

2.3. **Invariant extension, frame and anti-frame rules.** As in Pottier's work [21], following the approach to higher-order frame rules in [10], each of the syntactic categories is equipped with an *invariant extension* operation, $\cdot \otimes C$. Intuitively, this

$$\frac{\Delta,\, f : \chi_1 \to \chi_2,\, x : \chi_1 \Vdash t : \chi_2}{\Delta \vdash \mathsf{fun}\, f(x){=}t : \chi_1 \to \chi_2} \qquad \frac{\Delta \vdash v : \chi_1 \to \chi_2 \qquad \Delta, \Gamma \Vdash t : \chi_1}{\Delta, \Gamma \Vdash (v\, t) : \chi_2}$$

$$\frac{\Gamma \Vdash v : \tau}{\Gamma \Vdash \mathsf{ref}\, v : \exists \sigma.[\sigma] * \{\sigma : \mathsf{ref}\, \tau\}} \qquad \frac{\Gamma \Vdash v : [\sigma] * \{\sigma : \mathsf{ref}\, \tau\}}{\Gamma \Vdash \mathsf{get}\, v : \tau * \{\sigma : \mathsf{ref}\, \tau\}}$$

$$\frac{\Gamma \Vdash v : ([\sigma] \times \tau_2) * \{\sigma : \mathsf{ref}\, \tau_1\}}{\Gamma \Vdash \mathsf{set}\, v : 1 * \{\sigma : \mathsf{ref}\, \tau_2\}}$$

FIGURE 4. Some typing rules for values and expressions

Monoid structures on capabilities

$$C_1 \circ C_2 \overset{def}{=} (C_1 \otimes C_2) * C_2 \qquad C_1 * C_2 = C_2 * C_1 \tag{1}$$

$$(C_1 \circ C_2) \circ C_3 = C_1 \circ (C_2 \circ C_3) \qquad (C_1 * C_2) * C_3 = C_1 * (C_2 * C_3) \tag{2}$$

$$C \circ \emptyset = C \qquad C * \emptyset = C \tag{3}$$

Monoid actions

$$(\cdot \otimes C_1) \otimes C_2 = \cdot \otimes (C_1 \circ C_2) \qquad \cdot \otimes \emptyset = \cdot \tag{4}$$

$$(\cdot * C_1) * C_2 = \cdot * (C_1 * C_2) \qquad \cdot * \emptyset = \cdot \tag{5}$$

Action by $*$ on linear environments

$$(\Gamma, x{:}\chi) * C = \Gamma, x{:}(\chi * C) = (\Gamma * C), x{:}\chi \tag{6}$$

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

$$(\cdot * \cdot) \otimes C = (\cdot \otimes C) * (\cdot \otimes C) \tag{7}$$

$$\{\sigma : \mathsf{ref}\, \tau\} \otimes C = \{\sigma : \mathsf{ref}\, \tau \otimes C\} \tag{8}$$

$$1 \otimes C = 1 \tag{9}$$

$$\mathsf{int} \otimes C = \mathsf{int} \tag{10}$$

$$(\tau_1 + \tau_2) \otimes C = (\tau_1 \otimes C) + (\tau_2 \otimes C) \tag{11}$$

$$(\tau_1 \times \tau_2) \otimes C = (\tau_1 \otimes C) \times (\tau_2 \otimes C) \tag{12}$$

$$(\chi_1 \to \chi_2) \otimes C = (\chi_1 \circ C) \to (\chi_2 \circ C) \tag{13}$$

$$[\sigma] \otimes C = [\sigma] \tag{14}$$

$$(\exists \sigma.\chi) \otimes C = \exists \sigma.(\chi \otimes C) \qquad \text{if } \sigma \notin \mathit{RegNames}(C) \tag{15}$$

$$\varnothing \otimes C = \varnothing \tag{16}$$

$$(\Gamma, x{:}\chi) \otimes C = (\Gamma \otimes C), x{:}(\chi \otimes C) \tag{17}$$

FIGURE 5. Some axioms of the structural equivalence relation

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 [10]. The two key cases of the structural equivalence are the distribution axioms for arrow types, $(\chi_1 \to \chi_2) \otimes C = (\chi_1 \otimes$

$$\textsc{Shallow frame}\quad \frac{\Gamma \Vdash t : \chi}{\Gamma * C \Vdash t : \chi * C} \qquad \textsc{Deep frame (comp)}\quad \frac{\Gamma \Vdash t : \chi}{(\Gamma \otimes C) * C \Vdash t : (\chi \otimes C) * C}$$

$$\textsc{Deep frame (val)}\quad \frac{\Delta \vdash v : \tau}{\Delta \otimes C \vdash v : \tau \otimes C} \qquad \textsc{Anti-frame}\quad \frac{\Gamma \otimes C \Vdash t : (\chi \otimes C) * C}{\Gamma \Vdash t : \chi}$$

Figure 6. Frame and anti-frame rules

$C * C) \to (\chi_2 \otimes C * C)$, 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 $(C_1 \otimes C_2) * C_2$. Figure 5 shows some of the axioms that define the structural equivalence. The operations $*$ and $\circ$ form two monoid structures on the capabilities (equations 1–3), and $*$ and the invariant extension operation $\otimes$ are actions of these monoids (equations 4–17). The structural equivalence also includes the unfolding equations for recursive capabilities and types.

The view of capabilities as the assertions of a program logic provides some intuition for the "shallow" and "deep" frame rules, and for the (essentially dual) anti-frame rule given in Figure 6. 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 Shallow frame and the deep variant Deep frame 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 [10], 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 $\cdot \otimes C$.

2.4. **Example: typing Landin's knot.** Pottier [21] 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., using back-patching and recursion through the heap: employing the standard let notation as syntactic sugar, *fix* can be written as

$$\begin{aligned} \mathsf{fun}\, \mathit{fix}(f) = \mathsf{let}\ & r = \mathsf{ref}\,\langle\rangle \\ & h = \lambda y.\,(f\,(\lambda x.(\mathsf{get}\, r)\ x))\ y \\ & \_ = \mathsf{set}\,\langle r, h\rangle \\ \mathsf{in}\ & h \end{aligned}$$

Every time the combinator is called with a functional $f : (\chi_1 \to \chi_2) \to (\chi_1 \to \chi_2)$, a new reference cell σ is allocated in order to set up the recursion required for the resulting fixed point $\mathit{fix}\, f$. Subsequent calls to $\mathit{fix}\, f$ rely on this cell, and one needs to know that the code stored in σ preserves the properties of this cell. More precisely, the invariant for the cell is the (recursive) capability $I = \{\sigma : \mathsf{ref}\,(\chi_1 \to \chi_2) \otimes I\}$. When type-checking the body of *fix* in the context $\Gamma = f{:}(\chi_1 \to \chi_2) \to (\chi_1 \to \chi_2), \ldots$ we have

$$(\Gamma, r{:}[\sigma]) \otimes I \vdash \lambda x.(\mathsf{get}\, r)\ x : (\chi_1 \to \chi_2) \otimes I$$

and therefore

$$(\Gamma, r{:}[\sigma]) \otimes I \vdash \lambda y.\,(f\,(\lambda x.(\mathsf{get}\, r)\ x))\ y : (\chi_1 \to \chi_2) \otimes I$$

Thus, the strong update in the third line of the definition of *fix* establishes the invariant I:

$$(\Gamma, r{:}[\sigma]) \otimes I, h : (\chi_1 \to \chi_2) \otimes I * \{\sigma : \mathsf{ref}\,1\} \Vdash \mathsf{set}\,\langle r, h\rangle : 1 * \{\sigma : \mathsf{ref}\,(\chi_1 \to \chi_2) \otimes I\}$$

From this we obtain $(\Gamma, r{:}[\sigma]) \otimes I * \{\sigma : \mathsf{ref}\,1\} \Vdash \mathsf{let}\ h = \ldots\ \mathsf{in}\ h : (\chi_1 \to \chi_2) \otimes I * I$ which, by applying structural equivalence axioms 17, 14, and 6 can be rewritten as

$$(\Gamma, r{:}[\sigma] * \{\sigma : \mathsf{ref}\,1\}) \otimes I \Vdash \mathsf{let}\ h = \ldots\ \mathsf{in}\ h : (\chi_1 \to \chi_2) \otimes I * I$$

At this point, the anti-frame rule allows us to hide the reliance of the result on σ:

$$\Gamma, r{:}[\sigma] * \{\sigma : \mathsf{ref}\,1\} \Vdash \mathsf{let}\ h = \ldots\ \mathsf{in}\ h : \chi_1 \to \chi_2$$

This leads to a purely functional interface of the fixed point combinator: after hiding I, we can ascribe *fix* the type $((\chi_1 \to \chi_2) \to (\chi_1 \to \chi_2)) \to (\chi_1 \to \chi_2)$. Thus 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. Kripke Semantics of Frame and Anti-frame Rules

Our soundness proof of the frame and anti-frame rules is based on two key ideas. The first idea is an interpretation of arrow types which explicates the universal and existential quantifications that are implicit in the anti-frame rule. Recall that $\cdot \circ C = \cdot \otimes C * C$ abbreviates the operation of combining two capabilities. Roughly speaking, in our model, an arrow type $\chi_1 \to \chi_2$ consists of the procedures that have type

$$\forall C.\ \big(\chi_1 \circ C \ \to\ \exists C'.\ \chi_2 \circ (C \circ C')\big)$$

in a standard interpretation. Pottier [21] showed how the anti-frame rule allows encoding ML-like weak references in terms of strong references. Readers who are familiar with Kripke models of ML references (see, e.g., [13]) may thus find the above interpretation natural, by reading the type as *for all worlds C, if the procedure is given an argument of type χ_1 in world C, then, for some future world $C \circ C'$ (an extension of C), the procedure returns a result of type χ_2 in world $C \circ C'$.*

Intuitively, as indicated earlier, capabilities are like assertions in a Hoare-style program logic and thus describe heaps. However, to formalize the above meaning of arrow types, the second key idea of our model is that capabilities (as well as types and type contexts) are parameterized by invariants. This parameterization will make it easy to interpret the invariant extension operation $\otimes$, as in earlier work [10, 27]. 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. Essentially, invariant extension of $C \otimes C'$ is then interpreted by applying $[\![C]\!]$ to (the interpretation of) the given invariant C'. 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 Figure 6 suggest, 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 $W = W \to \mathit{Pred}(\mathit{Heap})$.

In fact, in order to prove the soundness of the anti-frame rule, we will also need to consider a preorder on W and ensure that the interpretation of capabilities and types is monotone, which leads to the equation

$$W \ = \ W \to_{mon} \mathit{Pred}(\mathit{Heap})\ . \tag{18}$$

The preorder on W is induced by a monoid structure on W. More precisely, $w_1 \sqsubseteq w_1'$ holds if w_1' is $w_1 \circ w_2$ for some w_2, where

$$(w_1 \circ w_2)(w) \;=\; (w_1 \otimes w_2)(w) * w_2(w) \tag{19}$$

reflects the syntactic operation $C_1 \circ C_2$, and where

$$(w_1 \otimes w_2)(w) \;=\; w_1(w_2 \circ w) \tag{20}$$

is the semantic analogue of invariant extension. In particular, the monotonicity condition in (18) states that $\llbracket C \rrbracket\,(\llbracket C_1 \rrbracket) \subseteq \llbracket C \rrbracket\,(\llbracket C_1 \circ C_2 \rrbracket)$ holds for any capability C, which means that additional invariants (here C_2, appearing in the combined invariant $C_1 \circ C_2$) cannot invalidate C with respect to a given invariant (here C_1). Intuitively, this property is necessary since C_1 may have been hidden by the anti-frame rule, i.e., C_1 is not visible in the program logic at the point where the frame rule is applied to introduce C_2.

Note that the operations (19) and (20) are mutually recursive, that w_2 on the right-hand side of (19) is used both as an element in W and as a function on W, and that the monotonicity condition in (18) refers to the operation in (19). In Sections 5 and 6 we will construct sets of worlds W that satisfy a suitable variant of (18), using ultrametric spaces. To this end, we recall some basic definitions and results about metric spaces in the next section.

4. Ultrametric Spaces and Uniform Relations

This section summarizes some basic notions from the theory of metric spaces, and introduces "uniform relations" which will be used as building blocks for the interpretation in the following sections. For a less condensed introduction we refer to Smyth [30] and Birkedal et al. [9].

4.1. **Ultrametric spaces.** 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 $d(x, y) \leq \max\{d(x, z), d(z, y)\}$, for all $x, y, z \in X$. A *Cauchy sequence* is a sequence $(x_n)_{n\in\mathbb{N}}$ of elements in X such that for every $k \in \mathbb{N}$ there exists an index n and for all $n_1, n_2 \geq n$, $d(x_{n_1}, x_{n_2}) \leq 2^{-k}$. A metric space is *complete* if every Cauchy sequence $(x_n)_{n\in\mathbb{N}}$ has a limit $\lim_n x_n$.

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 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**. The terminal object **1** is given by any singleton space, and 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 $d(x, y) \leq 2^{-n}$. Each relation $\overset{n}{=}$ is an equivalence relation because of the ultrametric inequality, and 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 $x = y$; this observation allows us to prove equalities by induction on n.

If X is *bisected*, i.e., if all distances in X are of the form 2^{-n} for some n, then a function $f : X \to Y$ is non-expansive if and only if $x \overset{n}{=} x'$ implies $f(x) \overset{n}{=} f(x')$. In the following, all the metric spaces that we consider have this property.

4.2. **Uniform relations.** In order to rephrase the (informal) requirement (18) in **CBUlt**, we consider uniform relations in place of arbitrary predicates on *Heap*. More generally, let $(A, \leq)$ be a partially ordered set. An *(upwards closed) uniform relation* 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:

$$(k, a) \in p \ \wedge \ j \leq k \ \wedge \ a \leq b \ \Rightarrow \ (j, b) \in p \ .$$

We write $\mathit{URel}(A)$ for the set of all such relations on A, and for $k \in \mathbb{N}$ we define $p_{[k]} = \{(j, a) \mid j < k\}$. Note that $p_{[k]} \in \mathit{URel}(A)$ if $p \in \mathit{URel}(A)$, and that $p \subseteq p'$ implies $p_{[n]} \subseteq p'_{[n]}$. We equip $\mathit{URel}(A)$ with the distance function $d(p, q) = \inf\{2^{-n} \mid p_{[n]} = q_{[n]}\}$, which makes $(\mathit{URel}(A), d)$ an object of **CBUlt**. Moreover, $\mathit{URel}(A)$ forms a complete Heyting algebra.

Proposition 1. *$\mathit{URel}(A)$, ordered by inclusion, forms a complete Heyting algebra. Meets and joins are given by set-theoretic intersections and unions, resp., and implication $p \Rightarrow q$ is given by the uniform relation such that $(k, a) \in (p \Rightarrow q)$ holds if and only if for all $j \leq k$ and all $b \geq a$, $(j, b) \in p$ implies $(j, b) \in q$.*

Meets, joins and implication are non-expansive operations on $\mathit{URel}(A)$ with respect to the distance function d defined above.

In our model, we use $\mathit{URel}(A)$ with the following concrete instances for the partial order $(A, \leq)$:

(1) *heaps* $(\mathit{Heap}, \leq)$, where $h \leq h'$ if and only if $h' = h \cdot h_0$ for some $h_0 \# h$,
(2) *values* $(\mathit{Val}, \leq)$, where $v \leq v'$ if and only if $v = v'$,
(3) *stateful values* $(\mathit{Val} \times \mathit{Heap}, \leq)$, where $(v, h) \leq (v', h')$ if and only if $v = v'$ and $h \leq h'$, and
(4) *stateful expressions* $(\mathit{Exp} \times \mathit{Heap}, \leq)$, where $(t, h) \leq (t', h')$ if and only if $t = t'$ and $h = h'$.

We also use variants of (2) and (3) where the set *Val* is replaced by the set of value substitutions, *Env*.

In the case of uniform relations on *Heap* we have a complete BI algebra structure [25]: a separating conjunction and separating implication as its right adjoint, are given by

$$\begin{aligned} (k, h) \in (p_1 * p_2) \ &\Leftrightarrow \ \exists h_1, h_2.\ h = h_1 \cdot h_2 \ \wedge \ (k, h_1) \in p_1 \wedge \ (k, h_2) \in p_2 \\ (k, h) \in (p \mathrel{-\!\!*} q) \ &\Leftrightarrow \ \forall j \leq k.\ \forall h' \# h.\ (j, h') \in p \ \Rightarrow \ (j, h \cdot h') \in q \end{aligned}$$

The unit for $*$ is given by $I = \mathbb{N} \times \mathit{Heap}$. Up to the natural number indexing, this is just the standard intuitionistic (in the sense that it is not "tight") heap model of separation logic [26]. Both separating conjunction and separating implication are non-expansive operations on $\mathit{URel}(\mathit{Heap})$.

In the following, when interpreting types and capabilities, we will not need to use all of the algebraic structure on uniform relations. Nevertheless, the fact that the uniform relations form a complete Heyting (BI) algebra suggests that Charguéraud and Pottier's system could, in principle, be extended to a full-blown program logic by including all of the logical connectives of separation logic assertions in the syntax of capabilities.

4.3. **Preordered metric spaces.** The uniform relations $\mathit{URel}(A)$, ordered by inclusion, form an example of a preordered metric space. More generally, a *preordered, complete, 1-bounded ultrametric space* is an object $(X, d) \in \mathbf{CBUlt}$ equipped with a preorder $\leq$ such that for all Cauchy sequences $(x_n)_{n\in\mathbb{N}}$ and $(y_n)_{n\in\mathbb{N}}$, if $x_n \leq y_n$ holds for all $n \in \mathbb{N}$ then $\lim_n x_n \leq \lim_n y_n$.

For preordered, complete, 1-bounded ultrametric spaces X_1 and X_2 we write $X_1 \to_{mon} X_2$ for the set of non-expansive and monotone functions between X_1 and X_2. When equipped with the sup-distance, $d(f, g) = \sup\{d_2(f\,x, g\,x) \mid x \in X_1\}$, the set $X_1 \to_{mon} X_2$ becomes an object of $\mathbf{CBUlt}$.

Proposition 2. *For any preordered, complete, 1-bounded ultrametric spaces X and Y, $X \to_{mon} Y$ equipped with the pointwise order forms a complete Heyting algebra where the operations are non-expansive. Meets and joins are given by the pointwise extension of the corresponding operations on Y, and $f \Rightarrow g$ is defined by $(f \Rightarrow g)(x) = \bigwedge_{x' \geq x}\big(f(x') \Rightarrow g(x')\big)$.*

In the case where Y is $\mathit{URel}(\mathit{Heap})$, $X \to_{mon} \mathit{URel}(\mathit{Heap})$ is a complete BI algebra where $*$ and $-\!\!*$ are non-expansive operations. Separating conjunction $f * g$ and its unit I are defined pointwise, and the separating implication $f -\!\!* g$ is defined by $(f -\!\!* g)(x) = \bigwedge_{x' \geq x}\big(f(x') -\!\!* g(x')\big)$.

5. Monotone Recursive Worlds

In this section we prove the following existence theorem:

Theorem 3 (Existence of monotone recursive worlds)**.** *There exists a preordered monoid $(W, \sqsubseteq, \circ, e)$ where W is an object of* $\mathbf{CBUlt}$ *with a (non-expansive) isomorphism ι from $\big(\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})\big)$ to W, such that the following conditions hold:*

1. *The preorder on W is given by $w \sqsubseteq w' \Leftrightarrow \exists w_0.\ w' = w \circ w_0$.*
2. *The operation $\circ : W \times W \to W$ is non-expansive.*
3. *For all $w_1, w_2, w \in W$, $\iota^{-1}(w_1 \circ w_2)(w) = \iota^{-1}(w_1)(w_2 \circ w) * \iota^{-1}(w_2)(w)$.*

In condition (3), the operation $*$ is the separating conjunction on uniform heap relations described in Proposition 1. By Proposition 2, $\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ is a complete BI algebra.

This theorem asserts the existence of a suitable set of worlds for the interpretation of the capability calculus. In particular, if we define $(f \otimes w_1)(w) = f(w_1 \circ w)$ for $f \in \big(\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})\big)$ and $w_1, w \in W$ we have

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

Apart from the insertion of the isomorphism in this equation, the difference between the statement of the theorem and the informal requirements (18–20) on page 8 are the use of uniform relations instead of arbitrary predicates over heaps, the restriction to non-expansive functions, and the scaling factor $\frac{1}{2}$. (Note that a non-expansive function $\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ is the same as a contractive function $W \to_{mon} \mathit{URel}(\mathit{Heap})$ with contraction factor $\frac{1}{2}$.)

We prove Theorem 3 by constructing $W \cong \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ explicitly, as (inverse) limit

$$W = \Big\{x \in \textstyle\prod_{k \geq 0} W_k \mid \forall k \geq 0.\ x_k = \epsilon_k^\circ(x_{k+1})\Big\}$$

of a sequence of "approximations" W_k of W,

$$\begin{tikzcd} W_0 \arrow[r, shift left, "\epsilon_0"] & W_1 \arrow[l, shift left, "\epsilon_0^\circ"] \arrow[r, shift left, "\epsilon_1"] & W_2 \arrow[l, shift left, "\epsilon_1^\circ"] \arrow[r, shift left, "\epsilon_2"] & \ldots \arrow[l, shift left, "\epsilon_2^\circ"] \arrow[r, shift left, "\epsilon_k"] & W_{k+1} \arrow[l, shift left, "\epsilon_k^\circ"] \arrow[r, shift left, "\epsilon_{k+1}"] & \ldots \arrow[l, shift left, "\epsilon_{k+1}^\circ"] \end{tikzcd} \tag{21}$$

Each W_k is a complete, 1-bounded, ultrametric space with distance function d_k, and comes equipped with a non-expansive operation $\circ_k : W_k \times W_k \to W_k$ and a preorder $\sqsubseteq_k$. This sequence will be defined inductively, so that $W_{k+1} = \frac{1}{2} \cdot W_k \to_{mon} \mathit{URel}(\mathit{Heap})$ are the non-expansive and monotone functions with respect to $\sqsubseteq_k$.

5.1. **Cauchy tower of approximants.** We define preordered, complete, 1-bounded ultrametric spaces $(W_k, \sqsubseteq_k)$, binary operations $\circ_k$ on W_k, and functions

$$\begin{tikzcd} W_k \arrow[r, shift left, "\epsilon_k"] & \left(\frac{1}{2} \cdot W_k \to_{mon} \mathit{URel}(\mathit{Heap})\right) \arrow[l, shift left, "\epsilon_k^\circ"] \end{tikzcd}$$

by induction on k as follows.

- $W_0 = \{\star\}$ is a one-point space;
- $\circ_0$ is given by $\star \circ_0 \star = \star$;
- $\sqsubseteq_0$ is the trivial order, $\star \sqsubseteq_0 \star$;
- $\epsilon_0(w) = I$, the unit of the BI algebra structure on $\frac{1}{2} \cdot W_k \to_{mon} \mathit{URel}(\mathit{Heap})$;
- $\epsilon_0^\circ(f) = \star$.

For $k \geq 0$,

- $W_{k+1} = \frac{1}{2} \cdot W_k \to_{mon} \mathit{URel}(\mathit{Heap})$;
- $\circ_{k+1}$ is given by $(f \circ_{k+1} g)(w) = f((\epsilon_k^\circ\, g) \circ_k w) * g(w)$;
- $f \sqsubseteq_{k+1} g$ holds if $g \overset{k+1}{=} f \circ_{k+1} f_0$ for some $f_0 \in W_{k+1}$;
- $\epsilon_{k+1}(f)$ sends $g \in \frac{1}{2} \cdot W_{k+1}$ to $(f(\epsilon_k^\circ\, g))_{[k+2]} \in \mathit{URel}(\mathit{Heap})$;
- $\epsilon_{k+1}^\circ(F)$ sends $w \in \frac{1}{2} \cdot W_k$ to $(F(\epsilon_k\, w))_{[k+1]} \in \mathit{URel}(\mathit{Heap})$.

In the rest of this subsection we show that $\sqsubseteq_k$ indeed defines a preorder on W_k, and that ϵ_k and ϵ_k° are non-expansive and map into the space of non-expansive and monotone functions. One technical inconvenience in these proofs is that the operations $\circ_k$ are not preserved by ϵ_k and ϵ_{k-1}°, and that they are not associative. However, associativity holds "up to approximation k," which explains the definition of $\sqsubseteq_k$ above.

Lemma 4 (Well-definedness)**.** *For all $k \geq 0$,*

1. *ϵ_k and ϵ_k° are non-expansive functions between W_k and $\frac{1}{2} \cdot W_k \to \mathit{URel}(\mathit{Heap})$.*
2. *For all $w, w' \in W_k$, $w \circ_k w' \in W_k$.*
3. *$\circ_k$ is a non-expansive operation on W_k.*
4. *For all $w, w', w'' \in W_k$, $(w \circ_k w') \circ_k w'' \overset{k}{=} w \circ_k (w' \circ_k w'')$.*
5. *For all $w \in W_{k+1}$, $I \circ_{k+1} w = w$ and $w \circ_{k+1} I \overset{k+1}{=} w$, where I is the unit of the BI algebra structure on $\frac{1}{2} \cdot W_k \to_{mon} \mathit{URel}(\mathit{Heap})$.*
6. *The relation $\sqsubseteq_k$ is a preorder on W_k.*
7. *For all $w \in W_k$ and $F \in W_{k+2}$, $\epsilon_k(w)$ and $\epsilon_{k+1}^\circ(F)$ are monotone functions $\frac{1}{2} \cdot W_k \to_{mon} \mathit{URel}(\mathit{Heap})$.*
8. *For all $w \in W_k$, $\epsilon_k^\circ(\epsilon_k\, w) \overset{k}{=} w$. For all $g \in W_{k+1}$, $\epsilon_k(\epsilon_k^\circ\, g) \overset{k}{=} g$.*
9. *For all $w, w' \in W_k$, $\epsilon_k(w \circ_k w') \overset{k}{=} (\epsilon_k\, w) \circ_{k+1} (\epsilon_k\, w')$. For all $g, g' \in W_{k+1}$, $\epsilon_k^\circ(g \circ_{k+1} g') \overset{k}{=} (\epsilon_k^\circ\, g) \circ_k (\epsilon_k^\circ\, g')$.*

Proof. The properties are proved simultaneously by induction on k. The case $k = 0$ follows directly from the definitions. We give the key ideas for the case $k > 0$:

1. The claimed non-expansiveness properties are consequences of the non-expansiveness of ϵ_{k-1} and ϵ_{k-1}°, obtained from the induction hypothesis, and from the non-expansiveness of function composition.
2. By induction hypothesis, $\circ_{k-1}$ and ϵ_{k-1}° are non-expansive functions; the non-expansiveness of $w \circ_k w'$ then follows with the non-expansiveness of $*$.

The monotonicity of $w \circ_k w'$ follows from the definition of $\sqsubseteq_k$, the approximate associativity of $\circ_{k-1}$ given by part (4) of the induction hypothesis, and the monotonicity of $*$ on $\mathit{URel}(\mathit{Heap})$.

(3) For non-expansiveness of $\circ_k$, note that $(\epsilon^\circ_{k-1}\, w_2) \circ_{k-1} w \overset{n}{=} (\epsilon^\circ_{k-1}\, w_2') \circ_{k-1} w$ holds for all $w \in W_{k-1}$ and all $w_2, w_2' \in W_k$ with $w_2 \overset{n}{=} w_2'$ by parts (1) and (3) of the induction hypothesis. Thus, for all w_1, w_1' with $w_1 \overset{n}{=} w_1'$, the non-expansiveness of $*$, w_1 and w_1' yields $(w_1 \circ_k w_2)(w) \overset{n}{=} (w_1' \circ_k w_2')(w)$. Since w is chosen arbitrarily, the sup-metric on W_k shows $w_1 \circ_k w_2 \overset{n}{=} w_1' \circ_k w_2'$.

(4) Given any $x \in \frac{1}{2} \cdot W_{k-1}$, $((w \circ_k w') \circ_k w'')(x) \overset{k}{=} (w \circ_k (w' \circ_k w''))(x)$ follows from parts (4) and (9) of the induction hypothesis. Thus, the claim follows with the sup-metric on W_k.

(5) That $I \circ_{k+1} w = I$ follows easily from the definition of I. For the second claim, first note that $(\epsilon^\circ_k\, I) \overset{k}{=} I$ holds in W_k. Thus, $(\epsilon^\circ_k\, I) \circ_k x \overset{k+1}{=} I \circ_k x = I$ holds in $\frac{1}{2} \cdot W_k$ for any x, by the non-expansiveness of $\circ_k$ and by the first claim. From this observation, the $k+1$-equivalence of $w \circ_{k+1} I$ and w follows with the non-expansiveness of w and the definition of $\circ_{k+1}$.

(6) That $\sqsubseteq_k$ is a preorder follows from its definition using parts (4) and (5).

(7) That $\epsilon_k(w)(w_1) \subseteq \epsilon_k(w)(w_2)$ holds for any $w, w_1, w_2 \in W_k$ with $w_1 \sqsubseteq_k w_2$ is a consequence of the monotonicity of w, part (9) of the induction hypothesis and the definition of ϵ_k. For the second claim observe that, whenever $w_1 \sqsubseteq_k w_2$, the definition of $\sqsubseteq_k$ and part (9) of the induction hypothesis yield $\epsilon_k(w_2) \overset{k}{=} \epsilon_k(w_1) \circ_{k+1} \epsilon_k(w_0)$ for some w_0. Hence,

$$\begin{aligned}\epsilon^\circ_{k+1}(F)(w_2) &= (F(\epsilon\, w_2))_{[k+1]}\\ &= (F(\epsilon_k(w_1) \circ_{k+1} \epsilon_k(w_0)))_{[k+1]} \supseteq (F(\epsilon\, w_1))_{[k+1]} = \epsilon^\circ_{k+1}(F)(w_1)\end{aligned}$$

by the contractiveness and monotonicity of F and the definition of ϵ°_{k+1}.

(8) The claims follow from part (8) of the induction hypothesis, using the fact that function composition is non-expansive and that functions in W_k for $k > 0$ are contractive with contraction factor $\frac{1}{2}$, due to the scaling in the definition of W_k.

(9) By part (9) of the induction hypothesis and by property (8) that we have just established, $\epsilon^\circ_{k-1}(\epsilon^\circ_k(\epsilon_k\, w') \circ_k w_0) \overset{k-1}{=} \epsilon^\circ_{k-1}(w') \circ_{k-1} \epsilon^\circ_{k-1}(w_0)$ holds for all $w', w_0 \in W_k$. Thus, using the definition of $\circ_k$ and ϵ_k, the contractiveness of $w \in W_k$, and the non-expansiveness of $*$,

$$\begin{aligned}&(\epsilon_k\, w \circ_{k+1} \epsilon_k\, w')(w_0)\\ &\quad = (w(\epsilon^\circ_{k-1}(\epsilon^\circ_k(\epsilon_k\, w') \circ_k w_0)))_{[k+1]} * (w'(\epsilon^\circ_{k-1}\, w_0))_{[k+1]}\\ &\quad \overset{k}{=} (w(\epsilon^\circ_{k-1}(w') \circ_{k-1} \epsilon^\circ_{k-1}(w_0)) \,*\, w'(\epsilon^\circ_{k-1}\, w_0))_{[k+1]}\\ &\quad = (w \circ_k w')(\epsilon^\circ_{k-1}\, w_0)_{[k+1]}\ ,\end{aligned}$$

which is just $\epsilon_k(w \circ_k w')(w_0)$. Since this approximate equality holds for all w_0, the claim follows by definition of the sup-metric on W_k.

The second claim is proved similarly.

□

Part (8) of Lemma 4 states that diagram (21) forms a Cauchy tower [9], meaning that $\sup_w d_{k+1}(w, \epsilon_k(\epsilon^\circ_k\, w))$ as well as $\sup_w d_k(w, \epsilon^\circ_k(\epsilon_k\, w))$ become arbitrarily small as k increases. This ensures that $W = \{x \in \prod_{k \geq 0} W_k \mid \forall k \geq 0.\ x_k = \epsilon^\circ_k(x_{k+1})\}$, equipped with the sup-distance, is an object of $\mathbf{CBUlt}$. Limits of Cauchy chains in W are given componentwise.

5.2. **Monoid structure on the inverse limit.** For all $0 \le k < l$, we define the functions $\epsilon_{k,l} : W_k \to W_l$ and $\epsilon^\circ_{k,l} : W_l \to W_k$ by

$$\epsilon_{k,l} = \epsilon_{l-1} \cdot \ldots \cdot \epsilon_{k+1} \cdot \epsilon_k \qquad\qquad \epsilon^\circ_{k,l} = \epsilon^\circ_k \cdot \epsilon^\circ_{k+1} \cdot \ldots \cdot \epsilon^\circ_{l-1}$$

which are non-expansive by Lemma 4(1).

Next, we equip W with an operation $\circ : W \times W \to W$ defined by

$$(x_k)_{k\ge 0} \circ (y_k)_{k\ge 0} \;=\; \big(\lim_{j>k} \epsilon^\circ_{k,j}(x_j \circ_j y_j)\big)_{k\ge 0}.$$

Note that the limits exist: $\epsilon^\circ_j(x_{j+1} \circ_{j+1} y_{j+1}) \overset{j}{=} \epsilon^\circ_j(x_{j+1}) \circ_j \epsilon^\circ_j(y_{j+1}) = x_j \circ_j y_j$ by Lemma 4(9), and so $(\epsilon^\circ_{k,j}(x_j \circ_j y_j))_{j>k}$ forms a Cauchy sequence in W_k by the non-expansiveness of $\epsilon^\circ_{k,j}$. Moreover, we have $\epsilon^\circ_k(\lim_{j>k+1} \epsilon^\circ_{k+1,j}(x_j \circ_j y_j)) = \lim_{j>k+1} \epsilon^\circ_{k,j}(x_j \circ_j y_j)$ which shows that $x \circ y$ is a sequence in W. We also define $e \overset{def}{=} (e_k)_{k\ge 0} \in W$ by $e_k = I$.

Lemma 5. $(W, \circ, e)$ *is a monoid with non-expansive multiplication* $\circ$.

Proof. From the definition of e and $\circ$, we have $e \circ w = w \circ e = w$ for all $w \in W$. To see the associativity of $\circ$, suppose $x, y, z \in W$. Lemma 4(4) shows for all j: $(x_j \circ_j y_j) \circ_j z_j \overset{j}{=} x_j \circ_j (y_j \circ_j z_j)$. We obtain

$$\begin{aligned}
x_j \circ_j (y \circ z)_j &= x_j \circ_j (\lim_{l>j} \epsilon^\circ_{j,l}(y_l \circ_l z_l)) && \\
&= \lim_{l>j} x_j \circ_j \epsilon^\circ_{j,l}(y_l \circ_l z_l)) && \text{by non-expansiveness of } \circ_j \\
&\overset{j}{=} \lim_{l>j} x_j \circ_j (\epsilon^\circ_{j,l}(y_l) \circ_j \epsilon^\circ_{j,l}(z_l)) && \text{by Lemma 4(9)} \\
&= \lim_{l>j} x_j \circ_j (y_j \circ_j z_j) && \text{since } y, z \in W \\
&\overset{j}{=} \lim_{l>j} (x_j \circ_j y_j) \circ_j z_j && \text{by Lemma 4(4)} \\
&\overset{j}{=} (x \circ y)_j \circ_j z_j .
\end{aligned}$$

Thus, for any real number $\varepsilon > 0$ there exists $n \ge 0$ sufficiently large such that

$$\forall j \ge n.\ d_{W_j}((x \circ y)_j \circ_j z_j,\ x_j \circ_j (y \circ z)_j) < \varepsilon.$$

Since $\epsilon^\circ_{k,j}$ is non-expansive, this yields for all k

$$((x \circ y) \circ z)_k = \lim_{j>k}\ \epsilon^\circ_{k,j}((x \circ y)_j \circ_j z_j) = \lim_{j>k}\ \epsilon^\circ_{k,j}(x_j \circ_j (y \circ z)_j) = (x \circ (y \circ z))_k$$

which proves $(x \circ y) \circ z = x \circ (y \circ z)$.

Finally, $\circ$ is non-expansive since each $\circ_j$ and $\epsilon^\circ_{k,j}$ is non-expansive. □

5.3. **Isomorphism between W and monotone functions on W.** As shown in the preceding Lemma 5, $\circ$ is associative and has e as a unit. Therefore we can consider the induced preorder on W, $w \sqsubseteq w' \Leftrightarrow \exists w_0.\ w' = w \circ w_0$. It remains to establish an isomorphism $W \cong \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ in **CBUlt** (where the monotonicity refers to this preorder $\sqsubseteq$ on W) that satisfies condition (3) from Theorem 3.

To this end, first note that if $w' = w \circ w''$ then $w'_k \overset{k}{=} w_k \circ_k w''_k$ for all k, and therefore we obtain

$$\forall w, w' \in W.\ w \sqsubseteq w' \;\Rightarrow\; \forall k.\ w_k \sqsubseteq_k w'_k\ . \tag{22}$$

Now note that for each k and for all sequences $(w_k)_{k\ge 0}$ and $(w'_k)_{k\ge 0}$ in W we have

$$w_{k+1}(w'_k) \;=\; \epsilon^\circ_{k+1}(w_{k+2})(\epsilon^\circ_k\, w'_{k+1}) \;=\; (w_{k+2}(\epsilon_k(\epsilon^\circ_k\, w'_{k+1})))_{[k+1]} \overset{k+1}{=} w_{k+2}(w'_{k+1})$$

by Lemma 4(8) and the contractiveness of w_{k+2}. Hence, $(\lambda w'.w_{k+1}(w'_k))_{k\geq 0}$ is a Cauchy sequence in $\frac{1}{2}\cdot W \to \mathit{URel}(\mathit{Heap})$. In fact, it is a sequence in the (complete) subspace of monotone maps, by (22) and the fact that each w_k is monotone, and therefore this sequence has a limit in $\frac{1}{2}\cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$. We may thus define

$$\iota^\bullet(w) \;=\; \lim_k(\lambda w' \in W.\, w_{k+1}(w'_k))$$

For $g \in \frac{1}{2}\cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ we define $\iota(g)_k \in W_k$ by the following two cases:

$$\begin{aligned}\iota(g)_0 \;&=\; \star\\ \iota(g)_{k+1} \;&=\; \lambda w \in \tfrac{1}{2}\cdot W_k.\,\big(g\big(\lim_{l>\max\{i,k\}}\ \epsilon^\circ_{i,l}(\epsilon_{k,l}\, w)\big)_{i\geq 0}\big)_{[k+1]}\end{aligned}$$

For this definition, one first checks that the sequence $(\lim_{l>\max\{i,k\}}\ \epsilon^\circ_{i,l}(\epsilon_{k,l}\, w))_{i\geq 0}$ is an element of W, so that g can be applied. Next, each $\iota(g)_{k+1}$ is monotone. To see this, let $w_1 \sqsubseteq_k w_2$, so by definition of $\sqsubseteq_k$ there exists $w_0 \in W_k$ such that w_2 and $w_1 \circ_k w_0$ are k-equivalent in W_k; we must show that $(\iota\, g)_{k+1}(w_1) \subseteq (\iota\, g)_{k+1}(w_2)$. Let $x_j = (\lim_{l>\max\{i,k\}}\ \epsilon^\circ_{i,l}(\epsilon_{k,l}\, w_j))_{i\geq 0}$ for $j = 0, 1, 2$. Then, $x_2 \overset{k}{=} x_1 \circ x_0$ holds in W. From the non-expansiveness and monotonicity of g it follows that $g(x_2) \overset{k+1}{=} g(x_1 \circ x_0) \supseteq g(x_1)$, and therefore $(g\, x_1)_{[k+1]} \subseteq (g\, x_2)_{[k+1]}$, which yields the claimed monotonicity of $\iota(g)_{k+1}$. Finally, $\iota\, g \in W$ holds since the definition of ι satisfies $\epsilon^\circ_k(\iota\, g)_{k+1} = (\iota\, g)_k$ for all k.

Lemma 6. *The assignment of g to $\iota(g)$ determines a non-expansive function from $\frac{1}{2}\cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ to W, with a non-expansive inverse given by $\iota^\bullet$.*

Proof sketch. The non-expansiveness of ϵ and ϵ° is easy to see. To show that $\iota^\bullet$ is a right-inverse to ι one first proves that $(\iota(\iota^\bullet\, w))_n \overset{n}{=} w_n$ holds for all $w \in W$ and $n \in \mathbb{N}$. This yields the required equality, since

$$(\iota(\iota^\bullet\, w))_l \;=\; \lim_{n>l} \epsilon^\circ_{l,n}((\iota(\iota^\bullet\, w))_n) \;=\; \lim_{n>l} \epsilon^\circ_{l,n}(w_n) \;=\; w_l$$

follows for each l by the non-expansiveness of the $\epsilon^\circ_{n,l}$'s.

That $\iota^\bullet$ is also a left-inverse to ι can be seen by a similar calculation. □

To finish the proof of Theorem 3 we need to establish the relationship between the monoid multiplication and the isomorphism:

Lemma 7. *For all $w_1, w_2, w \in W$, $\iota^{-1}(w_1 \circ w_2)(w) \;=\; \iota^{-1}(w_1)(w_2 \circ w) * \iota^{-1}(w_2)(w)$.*

Proof sketch. One first shows that $g(w) = \lim_k \lim_{j>k+1} (\iota\, g)_j(\epsilon^\circ_{k,j-1}\, w_k)$ for all g in $\frac{1}{2}\cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ and $w \in W$. Using this equation, the claim is then established by unfolding the definitions of $\circ$ and ι^{-1}. □

6. Hereditarily Monotone Recursive Worlds

In this section we present an alternative construction of a set of recursive worlds, which differs from the one defined in the previous section in some respects. Either set is suitable for the interpretation of the capability calculus.

6.1. **Recursive worlds.** The first step in this construction is the definition of recursive worlds without monotonicity condition. It is well-known that one can solve recursive domain equations in **CBUlt**, given by locally contractive functors, by an adaptation of the inverse-limit method from classical domain theory [2]. In particular, by considering the space of contractive but not necessarily monotone functions in the domain equation (18) above, America and Rutten's existence theorem applies.

Proposition 8. *There exists a unique (up to isomorphism) metric space $(X, d) \in$* **CBUlt** *and an isomorphism ι from $\frac{1}{2} \cdot X \to URel(Heap)$ to X.*

Proof. X is obtained by America and Rutten's existence theorem for fixed points of locally contractive functors [2], applied to the functor $F : \mathbf{CBUlt}^{op} \longrightarrow \mathbf{CBUlt}$, $F(X) = \frac{1}{2} \cdot X \to URel(Heap)$. □

The next step is to define the composition operation $\circ$ on X.

Lemma 9. *There exists a non-expansive operation $\circ : X \times X \to X$ such that*

$$\forall x_1, x_2, x \in X.\ \iota^{-1}(x_1 \circ x_2)(x) \ = \ \iota^{-1}(x_1)(x_2 \circ x) \ * \ \iota^{-1}(x_2)(x)\ ,$$

This operation is associative, and has emp $= \iota(I)$ as left and right unit, for $I(w) = \mathbb{N} \times Heap$ the unit of the lifted separating conjunction described in Proposition 2.

Proof. The operation $\circ$ can be defined by a straightforward application of Banach's fixed point theorem on the complete ultrametric space $X \times X \to X$. The proof that *emp* is a left and right unit is easy, for associativity one proves $x_1 \circ (x_2 \circ x_3) \overset{n}{=} (x_1 \circ x_2) \circ x_3$ for all $n \in \mathbb{N}$ by induction. See [27]. □

We define $f \otimes x$, for $f : \frac{1}{2} \cdot X \to URel(Heap)$ and $x \in X$, as the non-expansive function $\frac{1}{2} \cdot X \to URel(Heap)$ given by $(f \otimes x)(x') = f(x \otimes x')$.

Since $\circ$ defines a monoid structure on X there is an induced preorder on X given by $x \sqsubseteq y \ \Leftrightarrow\ \exists x_0.\ y = x \circ x_0$. We will now "carve out" a subset of functions in $\frac{1}{2} \cdot X \to URel(Heap)$ that are monotonic with respect to this preorder. This subset needs to be defined recursively.

6.2. **Relations on ultrametric spaces.** For $X \in$ **CBUlt** let $\mathcal{R}(X)$ be the collection of all non-empty and closed relations $R \subseteq X$; we will just write $\mathcal{R}$ when X is clear from the context. 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 10. *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 $R \in \mathcal{R}$.*

Proof. First, $\mathcal{R}$ is non-empty since it contains X itself, and d is well-defined since $R_{[0]} = S_{[0]}$ holds for any $R, S \in \mathcal{R}$. 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 $\mathcal{R}$. 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 is non-empty, and therefore $R \in \mathcal{R}$, 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 $x_0, \dots, x_n$, 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, $x \in R_{[k]}$. □

6.3. **Hereditarily monotone recursive worlds.** We will now define the set of hereditarily monotonic functions W as a recursive predicate on the space X from Proposition 8. Let the function $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ on subsets of X be given by

$$\Phi(R) = \{\iota(g) \mid \forall x, x_0 \in R.\ g(x) \subseteq g(x \circ x_0)\} \ .$$

The function restricts to a contractive function on $\mathcal{R}$:

Lemma 11. *If $R \in \mathcal{R}$ then $\Phi(R)$ is non-empty and closed, and $R \stackrel{n}{=} S$ implies $\Phi(R) \stackrel{n+1}{=} \Phi(S)$.*

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

We now show that Φ is contractive. To this end, let $n \geq 0$ and assume $R \stackrel{n}{=} S$. Let $\iota(g) \in \Phi(R)_{[n+1]}$. We must show that $\iota(g) \in \Phi(S)_{[n+1]}$. By definition of the closure operation there exists $\iota(f) \in \Phi(R)$ such that g and f are $(n{+}1)$-equal. Set $h(w) = f(w)_{[n+1]}$. Then h and g are also $(n{+}1)$-equal, hence it suffices to show that $\iota(h) \in \Phi(S)$. 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 $w_0' \stackrel{n}{=} w_0$ and $w_1' \stackrel{n}{=} w_1$ holds in 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

$$f(w_0) \ \stackrel{n+1}{=} \ f(w_0') \ \subseteq \ f(w_0' \circ w_1') \ \stackrel{n+1}{=} \ f(w_0 \circ w_1)$$

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

By Proposition 10 and the Banach theorem we can now define the hereditarily monotonic functions W as the uniquely determined fixed point of Φ.

Theorem 12 (Existence of hereditarily monotone recursive worlds)**.** *There exists a non-empty and closed subset $W \subseteq X$ satisfying the condition*

$$w \in W \;\Leftrightarrow\; \exists g.\; w = \iota(g) \;\wedge\; \forall w_1, w_2 \in W.\; g(w_1) \subseteq g(w_1 \circ w_2)\;.$$

Note that W thus constructed does not quite satisfy the conditions stated in Theorem 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

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

Then, for our particular application of interest, we also have to ensure that all the operations restrict appropriately (*cf.* Section 7 below). Here, as a first step, we show that the composition operation $\circ$ restricts to W.

Lemma 13. *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 the latter, 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 $w_1' \circ w_2 \in W$. □

Moreover, the BI algebra structure that exists on $\frac{1}{2}\cdot X \to \mathit{URel}(\mathit{Heap})$ by Proposition 2 restricts to the hereditarily monotone functions.

Proposition 14. *$\frac{1}{2}\cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$ forms a complete BI algebra where the operations are non-expansive. Meets and joins are given by the pointwise extension of intersection and union on $\mathit{URel}(\mathit{Heap})$, and $f \Rightarrow g$ is defined by $(f \Rightarrow g)(x) = \bigcap_{x_0 \in X}\big(f(x \circ x_0) \Rightarrow g(x \circ x_0)\big)$. Separating conjunction $f * g$ and its unit I are defined pointwise, and the separating implication $f \mathrel{-\!\!*} g$ is defined by $(f \mathrel{-\!\!*} g)(x) = \bigcap_{x_0 \in X}\big(f(x \circ x_0) \mathrel{-\!\!*} g(x \circ x_0)\big)$ from the separating implication on $\mathit{URel}(\mathit{Heap})$.*

7. Step-indexed Possible World Semantics of Capabilities

In this section we prove the soundness of the calculus of capabilities. After defining the semantic domains for the interpretation of types and capabilities, we give the full syntax and typing rules for the system presented in Section 2. Then, using the hereditarily monotone recursive worlds W, we construct a model of types and capabilities based on the operational semantics.

Alternatively, it is possible to use the monotone recursive worlds from Section 5 instead. This would require only minor and straightforward modifications of the interpretation below.

7.1. **Semantic domains and constructors.** Let $X \in \mathbf{CBUlt}$ denote the solution to the ultrametric equation $X \cong \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Heap})$ from Proposition 8, and let $W \in \mathcal{R}(X)$ denote the subset of hereditarily monotone recursive worlds (Theorem 12).

We define semantic domains for the capabilities and the value and memory types,

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

so that $g \in \mathit{Cap}$ if and only if $\iota(g) \in W$.

To define operations on the semantic domains that correspond to the syntactic type and capability constructors, we consider the lifting of (memory) types from values to expressions.

Definition 15 (Expression typing)**.** Consider $f : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$. The function $\mathcal{E}(f) : X \to \mathit{URel}(\mathit{Exp} \times \mathit{Heap})$ is defined by $(k, (t, h)) \in \mathcal{E}(f)(x)$ iff

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

Note that it is here where the indexing by natural numbers that is used in uniform relations (and which, in particular, induces the distance between uniform relations) is linked to the operational semantics of the programming language.

Also note that in this definition, f is a contractive function on X whereas $\mathcal{E}(f)$ is merely non-expansive. This is because the conclusion uses the world x as a heap predicate, qua $\iota^{-1}(x \circ w)(\mathit{emp})$, i.e. the scaling by 1/2 is undone, and the number j of steps taken in the reduction sequence may in fact be 0.

Lemma 16. *Let $f : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$. Then $\mathcal{E}(f)$ is non-expansive, and for all $x \in X$, $\mathcal{E}(f)(x) \in \mathit{URel}(\mathit{Exp} \times \mathit{Heap})$ is uniform. Moreover, the assignment of $\mathcal{E}(f)$ to f is non-expansive.*

Proof. Observe that $f \overset{n}{=} f'$ and $x \overset{n}{=} x'$ in X implies $f(x \circ w) \overset{n}{=} f'(x' \circ w)$ and $\iota^{-1}(x \circ w)(\mathit{emp}) \overset{n}{=} \iota^{-1}(x' \circ w)(\mathit{emp})$, for any $w \in W$, by the non-expansiveness of f, f' and $\circ$. Thus $\mathcal{E}(f)(x) \overset{n}{=} \mathcal{E}(f')(x')$. In particular, for $f = f'$ we obtain the non-expansiveness of $\mathcal{E}(f)$, and for $x = x'$ we obtain the non-expansiveness of $\mathcal{E}$ by definition of the sup metric. □

Definition 17 (Capability and type constructors)**.** In addition to separating conjunction and its unit, given in Proposition 14, we define the following operations.

Invariant extension: Let $g : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ and $w \in W$. We define $g \otimes w : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ by

$$(g \otimes w)(x) = g(w \circ x)$$

Separation: Let $p \in \mathit{URel}(A \times \mathit{Heap})$ and $r \in \mathit{URel}(\mathit{Heap})$. We define $p * r \in \mathit{URel}(A \times \mathit{Heap})$ by

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

This operation can be lifted pointwise, $(g{*}c)(x) = g(x){*}c(x)$ for $g : \frac{1}{2} \cdot X \to \mathit{URel}(A \times \mathit{Heap})$ and $c : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Heap})$. For notational convenience we will sometimes view $r \in \mathit{URel}(\mathit{Heap})$ as the constant function that maps any $x \in X$ to r, and thus write $g * r$ for this pointwise lifting.

Singleton capabilities: Let $v \in \mathit{Val}$ and $g : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$. We define $\{v : g\} : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Heap})$ by

$$\{v : g\}(x) = \{(k, h) \mid (k, (v, h)) \in g(x)\}$$

Name abstraction: Let $F : \mathit{Val} \to (\frac{1}{2} \cdot X \to \mathit{URel}(A))$, where Val is viewed as a discrete ultrametric space. Then $\exists F : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ is defined by

$$(\exists F)(x) = \bigcup_{v \in \mathit{Val}} F(v)(x)$$

Universal quantification: Let S be a set (viewed as an object of **CBUlt** with discrete metric), and let $F : S \to (\frac{1}{2} \cdot X \to \mathit{URel}(A))$. We define $\forall F : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ by

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

Recursion: Let $F : (\frac{1}{2} \cdot X \to \mathit{URel}(A)) \to (\frac{1}{2} \cdot X \to \mathit{URel}(A))$ be a contractive function. We define $\mathit{fix}\, F : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ by

$$\mathit{fix}\, F = \text{ the unique } g : \tfrac{1}{2} \cdot X \to \mathit{URel}(A) \text{ such that } g = F(g)$$

which exists by the Banach fixed point theorem.

Sum types: Let $g_1, g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val})$. We define $g_1 + g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val})$ by

$$(g_1 + g_2)(x) = \{(k, \mathsf{inj}^i v) \mid \forall j < k.\ (j, v) \in g_i(x)\}$$

Similarly, for $g_1, g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$ we define $g_1 + g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$ by

$$(g_1 + g_2)(x) = \{(k, (\mathsf{inj}^i v, h)) \mid \forall j < k.\ (j, (v, h)) \in g_i(x)\}$$

Product types: Let $g_1, g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val})$. We define $g_1 \times g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val})$ by

$$(g_1 \times g_2)(x) = \{(k, \langle v_1, v_2 \rangle) \mid \forall j < k.\ (j, v_i) \in g_i(x)\}$$

Similarly, for $g_1, g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$ we define $g_1 \times g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$ by

$$(g_1 \times g_2)(x) = \{(k, (\langle v_1, v_2 \rangle, h_1 \cdot h_2)) \mid \forall j < k.\ (j, (v_i, h_i)) \in g_i(x)\}$$

Arrow types: Let $g_1, g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$. We define $g_1 \to g_2 : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val})$ on $x \in X$ by

$$\begin{aligned} \{(k, \mathsf{fun}\, f(y){=}t) \mid \forall j < k.\ \forall w \in W.\ \forall r \in \mathit{URel}(\mathit{Heap}).& \\ \forall v, h.\ (j, (v, h)) \in g_1(x \circ w) * \iota^{-1}(x \circ w)(emp) * r \ \Rightarrow& \\ (j, (t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathcal{E}(g_2 * r)(x \circ w)\}& \end{aligned}$$

Reference types: Let $g : \frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$. We define $\mathit{ref}(g)$ in $\frac{1}{2} \cdot X \to \mathit{URel}(\mathit{Val} \times \mathit{Heap})$ by

$$\mathit{ref}(g)(x) \ = \{(k, (l, h \cdot [l \mapsto v])) \mid \forall j < k.\ (j, (v, h)) \in g(x)\}$$

The case for arrow types realizes the key ideas of our model that we have described in Section 3 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 $g_1 \to g_2$ is parametric in (and hence preserves) invariants that have been added by the procedure's context. In particular, the definition states that procedure application preserves this invariant, when viewed as the predicate $\iota^{-1}(x \circ w)(emp)$. 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 $\mathcal{E}$, allows us to "absorb" a part of the local heap description into the world. Finally, the quantification over indices $j < k$ in the definition of $g_1 \to g_2$ achieves that $(g_1 \to g_2)(x)$ is uniform. There are three reasons why we require that j be *strictly* less than k. Technically, as for the definition of $\mathcal{E}$, the use of $\iota^{-1}(x \circ w)$ in the definition undoes the scaling by $1/2$, and $j < k$ ensures the non-expansiveness of $g_1 \to g_2$ as a function $1/2 \cdot X \to \mathit{URel}(\mathit{Val})$. 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 use of $j < k$ in sum, product, and reference types instead of $j \leq k$ ensures that these constructors are contractive in their arguments and not merely non-expansive.

Lemma 18 (Well-definedness). *The operations given in Definition 17 are well-defined, i.e., each operation is a non-expansive function that maps into uniform relations of the right kind. Moreover, they restrict to non-expansive operations on monotonic functions:*

- *If $g : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$ then $g \otimes w : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$. The operation $g, w \mapsto g \otimes w$ is non-expansive in g and contractive in w.*
- *If $g : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A \times \mathit{Heap})$ and $c \in \mathit{Cap}$ then $g * c : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A \times \mathit{Heap})$. The operation $g, c \mapsto g * c$ is non-expansive in g and c.*
- *If $g \in MT$ then $\{v : g\} \in \mathit{Cap}$. The operation $g \mapsto \{v : g\}$ is non-expansive.*
- *If $F : \mathit{Val} \to (\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A))$ then $\exists F : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$. The operation $F \mapsto \exists F$ is non-expansive.*
- *If $F : S \to (\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A))$ then $\forall F : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$. The operation $F \mapsto \forall F$ is non-expansive.*
- *If $F : (\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)) \to (\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A))$ is contractive then $\mathit{fix}\, F : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$. The operation $F \mapsto \mathit{fix}\, F$ is non-expansive.*
- *If $g_1, g_2 \in VT$ then $g_1 + g_2 \in VT$, and if $g_1, g_2 \in MT$ then $g_1 + g_2 \in MT$. The operations $g_1, g_2 \mapsto g_1 + g_2$ are contractive in g_1 and g_2.*
- *If $g_1, g_2 \in VT$ then $g_1 \times g_2 \in VT$, and if $g_1, g_2 \in MT$ then $g_1 \times g_2 \in MT$. The operations $g_1, g_2 \mapsto g_1 \times g_2$ are contractive in g_1 and g_2.*
- *If $g_1, g_2 \in MT$ then $g_1 \to g_2 \in VT$. The operation $g_1, g_2 \mapsto g_1 \to g_2$ is contractive in g_1 and g_2.*
- *If $g \in MT$ then $\mathit{ref}(g) \in MT$. The operation $g \mapsto \mathit{ref}(g)$ is contractive.*

Proof. We consider the cases of invariant extension and sum types in detail.

- Let $g : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ and $w \in W$. Then, by definition, $(g \otimes w)(x) = g(w \circ x)$ is a uniform relation on A for any $x \in X$. By the non-expansiveness of g and $\circ$ (cf. Lemma 9), $g \otimes w$ is a non-expansive function.

  Next, we show that $\otimes$ restricts to the monotone functions. Assume $g : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$. To show $g \otimes w : \frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$ we must prove $(g \otimes w)(w_1) \subseteq (g \otimes w)(w_1 \circ w_2)$ for all $w_1, w_2 \in W$. Note that $w \in W$, and thus Lemma 13 shows $w \circ w_1 \in W$. Hence, $g(w \circ w_1) \subseteq g(w \circ w_1 \circ w_2)$

by the assumption $g : \frac{1}{2} \cdot W \to_{mon} URel(A)$, and the claim follows from the definition of $g \otimes w$.

We show that $\otimes$ is non-expansive in its first and contractive in its second argument. If $g \overset{n}{=} g'$ then $(g \otimes w)(x) \overset{n}{=} (g' \otimes w)(x)$ by definition of the sup-metric, which means that $g \mapsto g \otimes w$ is non-expansive. Finally, assuming that we have $w \overset{n}{=} w'$ for $w, w' \in W$, then $w \circ x \overset{n+1}{=} w' \circ x$ holds in $\frac{1}{2} \cdot X$ for any $x \in X$ by the non-expansiveness of $\circ$ and the scaling operation. Thus $(g \otimes w)(x) \overset{n+1}{=} (g \otimes w')(x)$ follows from the non-expansiveness of g, and since x was chosen arbitrarily the definition of the sup-metric yields $g \otimes w \overset{n+1}{=} g \otimes w'$ which shows that $w \mapsto g \otimes w$ is contractive.

- Let $g_1, g_1', g_2, g_2' : \frac{1}{2} \cdot X \to URel(Val)$, and assume $g_1 \overset{n}{=} g_1'$ and $g_2 \overset{n}{=} g_2'$. Then, for any $x, x' \in X$ such that $x \overset{n}{=} x'$ holds with respect to the metric on $\frac{1}{2} \cdot X$, the non-expansiveness of g_1, g_2 yields $g_1(x) \overset{n}{=} g_1(x')$ and $g_2(x) \overset{n}{=} g_2(x')$. Hence, $(j, v) \in g_1(x)$ if and only if $(j, v) \in g_1'(x')$ for any $j < n$, and $(j, v) \in g_2'(x)$ if and only if $(j, v) \in g_2'(x')$ for any $j < n$. By definition of $g_1 + g_2$ and $g_1' + g_2'$ it follows that $(g_1 + g_2)(x)_{[n+1]} = (g_1' + g_2')(x')_{[n+1]}$, i.e., that $(g_1 + g_2)(x) \overset{n+1}{=} (g_1 + g_2)(x')$. From this observation, taking $g_1 = g_1'$ and $g_2 = g_2'$, it follows immediately that $g_1 + g_2$ is non-expansive. Moreover, taking $x = x'$, the definition of the sup-metric shows that the assignment $g_1, g_2 \mapsto g_1 + g_2$ is contractive.

  Since $g_1(x)$ and $g_2(x)$ are uniform relations, it is easy to see that $(k, v) \in (g_1 + g_2)(x)$ implies $(j, v) \in (g_1 + g_2)(x)$ for all $j \leq k$. Finally, from the definition of $g_1 + g_2$ it follows that $g_1, g_2 \in VT$ implies $g_1 + g_2 \in VT$.

The remaining cases are similar. □

7.2. **Type system and soundness.** The syntax and typing rules of Charguéraud and Pottier's capability type system are given in Figures 7 and 8. In addition to the typing rules given earlier, the capability type system also features subtype and subcapability relations. Figure 9 shows some of the axioms that induce these relations. Axiom (23) is a variant of the first-order (shallow) frame rule from Figure 6.[3] Axiom (24) allows us to "garbage-collect" capabilities for parts of the heap that are no longer needed. This axiom only holds in a "non-tight" interpretation of assertions like we use it here. Axioms (25) and (26) permit to translate back and forth between a value type τ and a singleton type $[\sigma]$ (together with a capability for σ). The relation $\leq$ is defined inductively by inference rules (not shown here) which state that all type and capability constructors are covariant,[4] with two exceptions: as usual, arrow types are contravariant in their first argument, and $\otimes$ is invariant in its second argument.

Using the operations given in Definition 17, the interpretation of capabilities and types is defined in Figure 10 by induction on the syntax. The interpretation depends on an environment η, which maps region names $\sigma \in RegName$ to closed values $\eta(\sigma) \in Val$, capability variables γ to semantic capabilities $\eta(\gamma) \in Cap$, and type variables α and β to semantic types $\eta(\alpha) \in VT$ and $\eta(\beta) \in MT$. By Lemma 18 we obtain interpretations $[\![C]\!]_\eta \in Cap$, $[\![\tau]\!]_\eta \in VT$, and $[\![\theta]\!]_\eta \in MT$. Moreover, Lemma 18 shows that whenever C is formally contractive in ξ then

[3] The deep variant of this axiom, $\chi_1 \to \chi_2 \leq (\chi_1 \circ C) \to (\chi_2 \circ C)$, is not sound in the capability calculus. Pottier [23] gives a counterexample based on this axiom and the anti-frame rule, and a similar counterexample that does not use the anti-frame rule can be constructed along the lines of [27, Proposition 1].

[4] Unusually, even the reference types are covariant in Charguéraud and Pottier's system.

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

Figure 7. Syntax of capabilities and types

$$\frac{(x : \tau) \in \Delta}{\Delta \vdash x : \tau} \qquad \frac{}{\Delta \vdash \langle\rangle : 1} \qquad \frac{\Delta \vdash v : \tau_i}{\Delta \vdash (\mathsf{inj}^i\ v) : (\tau_1 {+} \tau_2)} \qquad \frac{\Delta \vdash v_1 : \tau_1 \qquad \Delta \vdash v_2 : \tau_2}{\Delta \vdash \langle v_1, v_2 \rangle : (\tau_1 \times \tau_2)}$$

$$\frac{\Delta \vdash v : \tau}{\Delta \Vdash v : \tau} \qquad \frac{\Delta \vdash v : \chi_1 \rightarrow \chi_2 \qquad \Delta, \Gamma \Vdash t : \chi_1}{\Delta, \Gamma \Vdash (v\,t) : \chi_2} \qquad \frac{\Gamma \Vdash v : [\sigma] * \{\sigma : \tau_1 \times \theta_2\}}{\Gamma \Vdash \mathsf{proj}^1\ v : \tau_1 * \{\sigma : \tau_1 \times \theta_2\}}$$

$$\frac{\Gamma \Vdash v : [\sigma] * \{\sigma : \theta_1 \times \tau_2\}}{\Gamma \Vdash \mathsf{proj}^2\ v : \tau_2 * \{\sigma : \theta_1 \times \tau_2\}} \qquad \frac{\Delta,\ f : \chi_1 \rightarrow \chi_2,\ x : \chi_1 \Vdash t : \chi_2}{\Delta \vdash \mathsf{fun}\, f(x){=}t : \chi_1 \rightarrow \chi_2} \qquad \frac{\Delta \vdash v : \tau}{\Delta \vdash v : \forall \xi . \tau}\, \xi \notin \Delta$$

$$\frac{\begin{array}{c} \Delta \vdash v_1 : (\exists \sigma_1 . [\sigma_1] * \{\sigma : [\sigma_1] + 0\} * \{\sigma_1 : \theta_1\} * C) \rightarrow \chi \\ \Delta \vdash v_2 : (\exists \sigma_2 . [\sigma_2] * \{\sigma : 0 + [\sigma_2]\} * \{\sigma_2 : \theta_2\} * C) \rightarrow \chi \\ \Delta, \Gamma \Vdash v : [\sigma] * \{\sigma : \theta_1 + \theta_2\} * C \end{array}}{\Delta, \Gamma \Vdash \mathsf{case}(v_1, v_2, v) : \chi}$$

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

$$\frac{\Gamma \Vdash t : \chi}{\Gamma * C \Vdash t : \chi * C} \qquad \frac{\Gamma \Vdash t : \chi}{(\Gamma \otimes C) * C \Vdash t : (\chi \otimes C) * C} \qquad \frac{\Delta \vdash v : \tau}{\Delta \otimes C \vdash v : \tau \otimes C}$$

$$\frac{\Gamma \otimes C \Vdash t : (\chi \otimes C) * C}{\Gamma \Vdash t : \chi} \qquad \frac{\Delta \vdash v : \tau' \qquad \tau' \leq \tau}{\Delta \vdash v : \tau} \qquad \frac{\Gamma \leq \Gamma' \qquad \Gamma' \Vdash t : \theta' \qquad \theta' \leq \theta}{\Gamma \vdash t : \theta}$$

Figure 8. Typing of values and expressions

$$\chi_1 \rightarrow \chi_2 \ \leq\ (\chi_1 * C) \rightarrow (\chi_2 * C) \tag{23}$$
$$C \ \leq\ \emptyset \tag{24}$$
$$\tau \ \leq\ \exists \sigma . [\sigma] * \{\sigma : \tau\} \tag{25}$$
$$[\sigma] * \{\sigma : \tau\} \ \leq\ \tau * \{\sigma : \tau\} \tag{26}$$

Figure 9. Some subtyping axioms

$g \mapsto [\![C]\!]_{\eta[\xi:=g]}$ is contractive (and similarly for formally contractive types τ and χ), which guarantees that the fixed points in Figure 10 are well-defined.

The structural equivalences given in Figure 5 can be verified with respect to this interpretation. The monoid equations follow since $*$ and $\circ$ define monoid structures

Capabilities, $[\![C]\!]_\eta : 1/2 \cdot W \to_{mon} \mathit{URel}(\mathit{Heap})$

$$\begin{array}{rlrl}
[\![C_1 \otimes C_2]\!]_\eta &= [\![C_1]\!]_\eta \otimes \iota([\![C_2]\!]_\eta) & [\![\emptyset]\!]_\eta &= I \\
[\![C_1 * C_2]\!]_\eta &= [\![C_1]\!]_\eta * [\![C_2]\!]_\eta & [\![\{\sigma : \theta\}]\!]_\eta &= \{\eta(\sigma) : [\![\theta]\!]_\eta\} \\
[\![\gamma]\!]_\eta &= \eta(\gamma) & [\![\exists\sigma.C]\!]_\eta &= \exists(\lambda v \in \mathit{Val}.\ [\![C]\!]_{\eta[\sigma:=v]}) \\
[\![\mu\gamma.C]\!]_\eta &= \mathit{fix}(\lambda c \in \mathit{Cap}.\ [\![C]\!]_{\eta[\gamma:=c]}) & [\![\forall\sigma.C]\!]_\eta &= \forall(\lambda v \in \mathit{Val}.\ [\![C]\!]_{\eta[\sigma:=v]})
\end{array}$$

Value types, $[\![\tau]\!]_\eta : 1/2 \cdot W \to_{mon} \mathit{URel}(\mathit{Val})$

$$\begin{array}{rlrl}
[\![\tau \otimes C]\!]_\eta &= [\![\tau]\!]_\eta \otimes \iota([\![C]\!]_\eta) & [\![0]\!]_\eta &= \lambda w.\,\emptyset \\
[\![1]\!]_\eta &= \lambda w.\,\mathbb{N} \times \{\langle\rangle\} & [\![\mathsf{int}]\!]_\eta &= \lambda w.\,\mathbb{N} \times \{\underline{n} \mid n \in \mathbb{Z}\} \\
[\![[\sigma]]\!]_\eta &= \lambda w.\,\mathbb{N} \times \{\eta(\sigma)\} & [\![\tau_1 + \tau_2]\!]_\eta &= [\![\tau_1]\!]_\eta + [\![\tau_2]\!]_\eta \\
[\![\tau_1 \times \tau_2]\!]_\eta &= [\![\tau_1]\!]_\eta \times [\![\tau_2]\!]_\eta & [\![\chi_1 \to \chi_2]\!]_\eta &= [\![\chi_1]\!]_\eta \to [\![\chi_2]\!]_\eta \\
[\![\alpha]\!]_\eta &= \eta(\alpha) & [\![\mu\alpha.\tau]\!]_\eta &= \mathit{fix}(\lambda g \in \mathit{VT}.\ [\![\tau]\!]_{\eta[\alpha:=g]}) \\
[\![\forall\sigma.\tau]\!]_\eta &= \forall(\lambda v \in \mathit{Val}.\ [\![\tau]\!]_{\eta[\sigma:=v]}) & &
\end{array}$$

Memory types, $[\![\theta]\!]_\eta : 1/2 \cdot W \to_{mon} \mathit{URel}(\mathit{Val} \times \mathit{Heap})$

$$\begin{array}{rlrl}
[\![\theta \otimes C]\!]_\eta &= [\![\theta]\!]_\eta \otimes \iota([\![C]\!]_\eta) & [\![\theta_1 + \theta_2]\!]_\eta &= [\![\theta_1]\!]_\eta + [\![\theta_2]\!]_\eta \\
[\![\tau]\!]_\eta &= \lambda w.\{(k,(v,h)) \mid (k,v) \in [\![\tau]\!]_\eta\, w\} & [\![\theta_1 \times \theta_2]\!]_\eta &= [\![\theta_1]\!]_\eta \times [\![\theta_2]\!]_\eta \\
[\![\mathsf{ref}\,\theta]\!]_\eta &= \mathit{ref}\,[\![\theta]\!]_\eta & [\![\theta * C]\!]_\eta &= [\![\theta]\!]_\eta * [\![C]\!]_\eta \\
[\![\beta]\!]_\eta &= \eta(\beta) & [\![\exists\sigma.\theta]\!]_\eta &= \exists(\lambda v \in \mathit{Val}.\ [\![\theta]\!]_{\eta[\sigma:=v]}) \\
[\![\mu\beta.\theta]\!]_\eta &= \mathit{fix}(\lambda g \in \mathit{MT}.\ [\![\theta]\!]_{\eta[\beta:=g]}) & [\![\forall\sigma.\theta]\!]_\eta &= \forall(\lambda v \in \mathit{Val}.\ [\![\theta]\!]_{\eta[\sigma:=v]})
\end{array}$$

Value contexts, $[\![\Delta]\!]_\eta : 1/2 \cdot W \to_{mon} \mathit{URel}(\mathit{Env})$

$$\begin{array}{rl}
[\![\Delta \otimes C]\!]_\eta &= [\![\Delta]\!]_\eta \otimes \iota([\![C]\!]_\eta) \\
[\![\varnothing]\!]_\eta &= \lambda w.\mathbb{N} \times \{[\,]\} \\
[\![\Delta, x{:}\tau]\!]_\eta &= \lambda w.\{(k, \rho[x \mapsto v]) \mid (k,\rho) \in [\![\Delta]\!]_\eta\, w \ \wedge\ (k,v) \in [\![\tau]\!]_\eta\, w\}
\end{array}$$

Linear contexts, $[\![\Gamma]\!]_\eta : 1/2 \cdot W \to_{mon} \mathit{URel}(\mathit{Env} \times \mathit{Heap})$

$$\begin{array}{rl}
[\![\Gamma \otimes C]\!]_\eta &= [\![\Gamma]\!]_\eta \otimes \iota([\![C]\!]_\eta) \\
[\![\varnothing]\!]_\eta &= \lambda w.\mathbb{N} \times (\{[\,]\} \times \mathit{Heap}) \\
[\![\Gamma, x{:}\chi]\!]_\eta\, w &= \lambda w.\{(k, (\rho[x \mapsto v], h \cdot h')) \mid \\
&\qquad\qquad (k,(\rho,h)) \in [\![\Gamma]\!]_\eta\, w \ \wedge\ (k,(v,h')) \in [\![\chi]\!]_\eta\, w\} \\
[\![\Gamma * C]\!]_\eta &= [\![\Gamma]\!]_\eta * [\![C]\!]_\eta
\end{array}$$

Figure 10. Interpretation of capabilities and types

on *Cap*; the latter via the bijection ι between W and *Cap*. We consider the case of associativity of $\circ$:

Lemma 19. *For all* C_1, C_2, C_3, $[\![C_1 \circ (C_2 \circ C_3)]\!] = [\![(C_1 \circ C_2) \circ C_3]\!]$.

Proof. We prove the following claim: for all C, C', $\iota\,[\![C \circ C']\!] = \iota\,[\![C]\!] \circ \iota\,[\![C']\!]$. It suffices to show $[\![C \circ C']\!]_\eta\, w = \iota^{-1}(\iota\,[\![C]\!]_\eta \circ \iota\,[\![C']\!]_\eta)(w)$ for all η and w, and this follows from the defining equation for $\circ$:

$$\begin{aligned}\iota^{-1}(\iota\,[\![C]\!]_\eta \circ \iota\,[\![C']\!]_\eta)(w) &= \iota^{-1}(\iota\,[\![C]\!]_\eta)(\iota\,[\![C']\!]_\eta \circ w) * \iota^{-1}(\iota\,[\![C']\!]_\eta)(w)\\ &= ([\![C]\!]_\eta \otimes \iota\,[\![C']\!]_\eta)(w) * [\![C']\!]_\eta\,(w)\\ &= [\![C \otimes C' * C']\!]_\eta\,(w) = [\![C \circ C']\!]_\eta\,(w)\end{aligned}$$

To prove the lemma, it suffices to prove $\iota\,[\![C_1 \circ (C_2 \circ C_3)]\!] = \iota\,[\![(C_1 \circ C_2) \circ C_3]\!]$. By the above claim, this is a consequence of the associativity of $\circ$ on X. □

Most of the remaining equations in Figure 5 (as well as other equivalences that appear in [11, 21]) are easy consequences of the pointwise definition of the operations in Definition 17. We consider the distribution axiom for arrow types, which is more involved:

Lemma 20. *For all* χ_1, χ_2 *and* C, $[\![(\chi_1 \to \chi_2) \otimes C]\!] = [\![(\chi_1 \circ C) \to (\chi_2 \circ C)]\!]$.

Proof. The lemma follows from the following claim.

$$\forall g_1, g_2 \in \mathit{MT}.\ \forall c \in \mathit{Cap}.\ (g_1 \to g_2) \otimes \iota(c) = (g_1 \otimes \iota(c) * c) \to (g_2 \otimes \iota(c) * c)$$

We prove the inclusion from left to right. For the proof, let $x \in X$, $k \in \mathbb{N}$ and assume $(k, (\mathsf{fun}\, f(y){=}t)) \in ((g_1 \to g_2) \otimes \iota(c))(x) = (g_1 \to g_2)(\iota(c) \circ x)$. We must show that $(k, (\mathsf{fun}\, f(y){=}t)) \in (g_1 \otimes \iota(c) * c) \to (g_2 \otimes \iota(c) * c)$. To this end, let $j < k$, $w \in W$, $r \in \mathit{URel}(\mathit{Heap})$, and suppose

$$\begin{aligned}(j, (v, h)) &\in (g_1 \otimes \iota(c) * c)(x \circ w) * \iota^{-1}(x \circ w)(\mathit{emp}) * r\\ &= g_1(\iota(c) \circ x \circ w) * c(x \circ w) * \iota^{-1}(x \circ w)(\mathit{emp}) * r\\ &= g_1(\iota(c) \circ x \circ w) * \iota^{-1}(\iota(c) \circ x \circ w)(\mathit{emp}) * r\ .\end{aligned}$$

Then, by assumption, $(j, (t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathcal{E}(g_2 * r)(\iota(c) \circ x \circ w)$. By unfolding the definition of $\mathcal{E}$, the latter is seen to be equivalent to

$$(j, (t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathcal{E}(g_2 \otimes \iota(c) * c * r)(x \circ w)\ ,$$

and thus $(k, (\mathsf{fun}\, f(y){=}t)) \in (g_1 \otimes \iota(c) * c) \to (g_2 \otimes \iota(c) * c)$.

The other inclusion is proved similarly. □

We give the semantics of typing judgements next. The semantics of a typing judgement for values simply establishes truth with respect to all worlds w, environments η, and indices $k \in \mathbb{N}$:

$$\models (\Delta \vdash v : \tau) \iff \forall \eta.\ \forall w.\ \forall k.\ \forall \rho.\ (k, \rho) \in [\![\Delta]\!]_\eta\, w \ \Rightarrow\ (k, \rho(v)) \in [\![\tau]\!]_\eta\, w$$

Here $\rho(v)$ means the application of the substitution ρ to v.

The semantics of the typing judgement for expressions mirrors the interpretation of the arrow case for value types, in that there is also a quantification over heap predicates $r \in \mathit{URel}(\mathit{Heap})$ and an existential quantification over $w' \in W$ through the use of $\mathcal{E}$:

$$\begin{aligned}\models\ (\Gamma \Vdash t : \chi) \iff\ & \forall \eta.\ \forall w \in W.\ \forall k.\ \forall \rho.\ \forall h.\ \forall r \in \mathit{URel}(\mathit{Heap}).\\ & (k, (\rho, h)) \in [\![\Gamma]\!]_\eta\, w * \iota^{-1}(w)(\mathit{emp}) * r\\ & \quad \Rightarrow\ (k, (\rho(t), h)) \in \mathcal{E}([\![\chi]\!]_\eta * r)(w)\end{aligned}$$

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. The existential quantifier plays an important part in the verification of the anti-frame rule below.

In the remainder of this section we prove soundness of the calculus of capabilities.

Theorem 21 (Soundness)**.**

- *If* $\Delta \vdash v : \tau$ *then* $\models (\Delta \vdash v : \tau)$.
- *If* $\Gamma \Vdash t : \chi$ *then* $\models (\Gamma \Vdash t : \chi)$.

In particular, if $\varnothing \vdash t : \chi$ *is a closed program that does not contain any locations, and if* $(t \,|\, h) \longmapsto^* (t' \,|\, h')$ *where* $(t' \,|\, h')$ *is irreducible, then* t' *is a value.*

To prove the theorem, we show that each typing rule preserves the truth of judgements. The proof of the frame rules is straightforward.

Lemma 22 (Soundness of the shallow frame rule)**.** *Suppose* $\models (\Gamma \Vdash t : \chi)$*. Then* $\models (\Gamma * C \Vdash t : \chi * C)$*.*

Proof. Assume $\models (\Gamma \Vdash t : \chi)$. We prove $\models (\Gamma * C \Vdash t : \chi * C)$. Let η be an environment, let $w \in W$, $k \in \mathbb{N}$, $r \in \mathit{URel}(\mathit{Heap})$ and assume

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

We can now instantiate the universally quantified r in the assumption $\models (\Gamma \Vdash t : \chi)$ with $[\![C]\!]_\eta\,(w) * r$, and obtain $(k, (\rho(t), h)) \in \mathcal{E}([\![\chi]\!]_\eta * ([\![C]\!]_\eta\,(w) * r))(w)$. Since $[\![C]\!]_\eta \in \mathit{Cap}$ we have $[\![C]\!]_\eta\,(w) \subseteq [\![C]\!]_\eta\,(w \circ w')$ for any $w' \in W$, and hence we obtain $(k, (\rho(t), h)) \in \mathcal{E}([\![\chi * C]\!]_\eta * r)(w)$ by unfolding the definition of $\mathcal{E}$. □

Lemma 23 (Soundness of the deep frame rule for expressions)**.** *Suppose* $\models (\Gamma \Vdash t : \chi)$*. Then* $\models (\Gamma \otimes C * C \Vdash t : \chi \otimes C * C)$*.*

Proof. Assume $\models (\Gamma \Vdash t : \chi)$. We prove $\models (\Gamma \otimes C * C \Vdash t : \chi \otimes C * C)$. Let η be an environment, let $w \in W$, $k \in \mathbb{N}$, $r \in \mathit{URel}(\mathit{Heap})$ and

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

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

Next, we consider the 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 [21], and has been worked out in more detail in [29].

Lemma 24 (Existence of commutative pairs)**.** *For all worlds* $w_0, w_1 \in W$*, there exist* $w_0', w_1' \in W$ *such that*

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

Proof. Fix $w_0, w_1 \in W$, and consider the 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 second argument. Also, F restricts to a function on the non-empty and closed subset $W \times W$ of $X \times X$. Thus, by Banach's fixpoint theorem, F has a unique fixpoint $(w_0', w_1') \in W \times W$. This means that

$$w_0' \;=\; \iota(\iota^{-1}(w_0) \otimes w_1') \qquad \text{and} \qquad w_1' \;=\; \iota(\iota^{-1}(w_1) \otimes w_0'). \tag{27}$$

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 $w \in X$.

$$\begin{aligned}
\iota^{-1}(w_0 \circ w_1')(w) &= \iota^{-1}(w_0)(w_1' \circ w) * \iota^{-1}(w_1')(w) && \text{(by definition of } \circ\text{)}\\
&= (\iota^{-1}(w_0) \otimes w_1')(w) * \iota^{-1}(w_1')(w) && \text{(by definition of } \otimes\text{)}\\
&= \iota^{-1}(w_0')(w) * (\iota^{-1}(w_1) \otimes w_0')(w) && \text{(by (27))}\\
&= \iota^{-1}(w_0')(w) * \iota^{-1}(w_1)(w_0' \circ w) && \text{(by definition of } \otimes\text{)}\\
&= \iota^{-1}(w_1)(w_0' \circ w) * \iota^{-1}(w_0')(w) && \text{(by commutativity of } *\text{)}\\
&= \iota^{-1}(w_1 \circ w_0')(w) && \text{(by definition of } \circ\text{).}
\end{aligned}$$

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 25 (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, $r \in \mathit{URel}(\mathit{Heap})$ and

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

We must prove $(k, (\rho(t), h)) \in \mathcal{E}([\![\chi]\!]_\eta * r)(w)$. By Lemma 24,

$$w_1 = \iota(\iota^{-1}(w) \otimes w_2), \quad w_2 = \iota([\![C]\!]_\eta \otimes w_1) \quad \text{and} \quad \iota([\![C]\!]_\eta) \circ w_1 = w \circ w_2 \tag{28}$$

holds for some worlds w_1, w_2 in W.

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

$$\begin{aligned}
[\![\Gamma]\!]_\eta\,(w) &\subseteq [\![\Gamma]\!]_\eta\,(w \circ w_2) && \text{by monotonicity of } [\![\Gamma]\!]_\eta \text{ and } w_2 \in W\\
&= [\![\Gamma]\!]_\eta\,(\iota([\![C]\!]_\eta) \circ w_1) && \text{since } \iota([\![C]\!]_\eta) \circ w_1 = w \circ w_2\\
&= [\![\Gamma \otimes C]\!]_\eta\,(w_1) && \text{by definition of } \otimes.
\end{aligned}$$

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

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

$$(k, (\rho, h)) \in [\![\Gamma]\!]_\eta\,(w) * \iota^{-1}(w)(\mathit{emp}) * r \ \subseteq\ [\![\Gamma \otimes C]\!]_\eta\,(w_1) * \iota^{-1}(w_1)(\mathit{emp}) * r\ . \tag{29}$$

By the assumed validity of the judgement $\Gamma \otimes C \Vdash t : \chi \otimes C * C$, (29) entails

$$(k, (\rho(t), h)) \in \mathcal{E}([\![\chi \otimes C * C]\!]_\eta * r)(w_1)\ . \tag{30}$$

We need to show that $(k, (\rho(t), h)) \in \mathcal{E}([\![\chi]\!]_\eta * r)(w)$, so assume $(\rho(t) \mid h) \longmapsto^j (t' \mid h')$ for some $j \leq k$ such that $(t' \mid h')$ is irreducible. From (30) we then obtain

$$(k{-}j, (t', h')) \in \textstyle\bigcup_{w'} [\![\chi \otimes C * C]\!]_\eta\,(w_1 \circ w') * \iota^{-1}(w_1 \circ w')(\mathit{emp}) * r\ . \tag{31}$$

Now observe that we have

$$\begin{aligned}
&[\![\chi \otimes C * C]\!]_\eta\,(w_1 \circ w') * \iota^{-1}(w_1 \circ w')(\mathit{emp})\\
&\quad = [\![\chi]\!]_\eta\,(\iota([\![C]\!]_\eta) \circ w_1 \circ w') * [\![C]\!]_\eta\,(w_1 \circ w') * \iota^{-1}(w_1 \circ w')(\mathit{emp})\\
&\quad = [\![\chi]\!]_\eta\,(\iota([\![C]\!]_\eta) \circ w_1 \circ w') * \iota^{-1}(\iota([\![C]\!]_\eta) \circ w_1 \circ w')(\mathit{emp})\\
&\quad = [\![\chi]\!]_\eta\,(\iota(w \circ w_2 \circ w') * \iota^{-1}(\iota(w \circ w_2 \circ w')(\mathit{emp})
\end{aligned}$$

since $\iota(\llbracket C \rrbracket_\eta) \circ w_1 = w \circ w_2$. Setting $w'' \overset{def}{=} w_2 \circ w'$ one obtains

$$\llbracket \chi \rrbracket_\eta (w \circ w'') * \iota^{-1}(w \circ w'')(emp) \ .$$

Thus, (31) entails that $(k{-}j, (t', h'))$ is in $\bigcup_{w''} \llbracket \chi \rrbracket_\eta (w \circ w'') * \iota^{-1}(w \circ w'')(emp) * r$, and we are done. □

Remark 26 (Monotonicity)**.** Note that it is in the above proof for the anti-frame rule where the monotonicity condition of the recursive worlds is exploited to establish (29). Monotonicity of $\llbracket C \rrbracket$ is also used to prove the shallow frame rule in Lemma 22 (and the first-order frame axiom in Proposition 27 below). However, this is only necessary because of the existential quantifier that is implicitly used in the postcondition, via the definition of $\mathcal{E}(\cdot)$. In a system without anti-frame rule, the quantifier can be dropped from the definition of $\mathcal{E}(\cdot)$ and no monotonicity condition of $\llbracket C \rrbracket$ is needed [6, 27].

We omit the proofs for the remaining typing rules. Using the model, we can also show that subtyping is sound. Recall that $\leq$ is an inductively defined relation on syntactic type expressions, defined by axioms (as shown in Figure 9) and rules that propagate those axioms through type constructors (omitted for brevity). One can show that syntactic subtyping is sound:

Proposition 27 (Soundness of subtyping)**.** *The three kinds of subtyping relations are sound. More precisely, for all η and w:*

(1) $C \leq C'$ *implies* $\llbracket C \rrbracket_\eta w \subseteq \llbracket C' \rrbracket_\eta w$,
(2) $\tau \leq \tau'$ *implies* $\llbracket \tau \rrbracket_\eta w \subseteq \llbracket \tau' \rrbracket_\eta w$,
(3) $\theta \leq \theta'$ *implies* $\llbracket \theta \rrbracket_\eta w \subseteq \llbracket \theta' \rrbracket_\eta w$.

Proof. The three statements are proved simultaneously by induction on the derivation of the subtyping judgement in question. One must show that the axioms in Figure 9 hold with respect to the interpretation given in Figure 10, and that all of the inference rules that define the subtyping judgements preserve these inclusions. We show three sample cases:

Axiom (23) is sound. We have to show that for all η and $w \in W$, $\llbracket \chi_1 \to \chi_2 \rrbracket_\eta w \subseteq \llbracket (\chi_1 * C) \to (\chi_2 * C) \rrbracket_\eta w$.
Assume $(k, \mathsf{fun}\, f(x){=}t) \in \llbracket \chi_1 \to \chi_2 \rrbracket_\eta w$. To see that $(k, \mathsf{fun}\, f(x){=}t)$ is also in the set $\llbracket (\chi_1 * C) \to (\chi_2 * C) \rrbracket_\eta w$, suppose that $j < k$, $w_0 \in W$ and $r \in \mathit{URel}(\mathit{Heap})$, and let

$$\begin{aligned}(j, (v, h)) \in \llbracket \chi_1 * C \rrbracket_\eta (w \circ w_0) * \iota^{-1}(w \circ w_0)(emp) * r \\ = \llbracket \chi_1 \rrbracket_\eta (w \circ w_0) * \iota^{-1}(w \circ w_0)(emp) * (r * \llbracket C \rrbracket_\eta (w \circ w_0))\end{aligned}$$

We must show that $(j, (t[f{:=}\mathsf{fun}\, f(x){=}t, x{:=}v], h)) \in \mathcal{E}(\llbracket \chi_2 * C \rrbracket_\eta * r)(w \circ w_0)$. So assume that $(t[f{:=}\mathsf{fun}\, f(x){=}t, x{:=}v] \,|\, h) \longmapsto^i (t' \,|\, h')$ for some $i \leq j$ and some irreducible configuration $(t' \,|\, h')$. By unfolding the definition of $\llbracket \chi_1 \to \chi_2 \rrbracket_\eta w$, we obtain

$$(j, (t[f{:=}\mathsf{fun}\, f(x){=}t, x{:=}v], h)) \in \mathcal{E}(\llbracket \chi_2 \rrbracket_\eta * (r * \llbracket C \rrbracket_\eta (w \circ w_0)))(w \circ w_0)$$

and hence that there exists $w_1 \in W$ such that

$$(j - i, (t', h')) \in \llbracket \chi_2 \rrbracket_\eta (w \circ w_0 \circ w_1) * \iota^{-1}(w \circ w_0 \circ w_1)(emp) * r * \llbracket C \rrbracket_\eta (w \circ w_0)$$

Since $w \circ w_0 \sqsubseteq w \circ w_0 \circ w_1$ entails $\llbracket C \rrbracket_\eta (w \circ w_0) \subseteq \llbracket C \rrbracket_\eta (w \circ w_0 \circ w_1)$ by monotonicity of $\llbracket C \rrbracket_\eta$, one obtains

$$(j - i, (t', h')) \in \llbracket \chi_2 * C \rrbracket_\eta (w \circ w_0 \circ w_1) * \iota^{-1}(w \circ w_0 \circ w_1)(emp) * r$$

which yields $(j, (t[f{:=}\mathsf{fun}\, f(x){=}t, x{:=}v], h)) \in \mathcal{E}(\llbracket \chi_2 * C \rrbracket_\eta * r)(w \circ w_0)$.

Axiom 24 is sound. We have to prove for all η and $w \in W$, $\llbracket C \rrbracket_\eta\, w \subseteq \llbracket \emptyset \rrbracket_\eta\, w$. This follows simply from the definition $\llbracket \emptyset \rrbracket_\eta\, w = \mathbb{N} \times \mathit{Heap}$.

The rule for covariant subtyping of $\otimes$, concluding $\tau \otimes C \leq \tau' \otimes C$ from $\tau \leq \tau'$, is sound. Assume that $\llbracket \tau \rrbracket_\eta\, w \subseteq \llbracket \tau' \rrbracket_\eta\, w$ holds for all η and $w \in W$. Then we have to show that $\llbracket \tau \otimes C \rrbracket_\eta\, w \subseteq \llbracket \tau' \otimes C \rrbracket_\eta\, w$ for all η and $w \in W$.
By definition, $\llbracket \tau \otimes C \rrbracket_\eta\, w = \llbracket \tau \rrbracket_\eta\, (\iota\, \llbracket C \rrbracket_\eta \circ w)$ and $\llbracket \tau' \otimes C \rrbracket_\eta\, w = \llbracket \tau' \rrbracket_\eta\, (\iota\, \llbracket C \rrbracket_\eta \circ w)$. Thus, the statement follows by instantiating the universally quantified world in the assumption by $\iota\, \llbracket C \rrbracket_\eta \circ w$. □

The soundness of the two subsumption rules given in the last line of Figure 8 is an immediate consequence of Proposition 27.

8. 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:

$$\mathsf{let}\ r = \mathsf{ref}\ 0\ \mathsf{in}\ \mathsf{fun}\, f(g){=}(\mathit{inc}(r); g\langle\rangle; \mathit{dec}(r)) \tag{32}$$

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 such that $\{\sigma : \mathsf{ref}\,(\mathsf{int}\ 0)\}$ is not an invariant for those calls.

Thus Pottier [22] 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" [12] 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 (32); see the examples in [22]. 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. For instance, for f defined by

$$\mathsf{let}\ r = \mathsf{ref}\ 1\ \mathsf{in}\ \mathsf{fun}\, f(g){=}(\mathsf{set}\,\langle r, 0\rangle\,; g\langle\rangle; \mathsf{set}\,\langle r, 1\rangle\,; g\langle\rangle)$$

we may wish to capture the fact that the cell $r : [\sigma]$ stores 1 after $f(g)$ returns. Intuitively, this holds since the calls to g may at most bump up the value of r from 0 to 1 (through recursive calls to f), and this fact can be captured in the type system by considering the $\{0,1\}$-indexed family of invariants $I\, n = \{\sigma : \mathsf{ref}\,(\mathsf{int}\ n)\}$ once we allow that calls with $I\, i$ may return with $I\, j$ for $j \geq i$. The idea is related to the notion of evolving invariants for local state in recent work on reasoning about contextual equivalence [1, 12].

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 $\kappa \to \textsc{cap}$):

$$(\chi_1 \to \chi_2) \otimes I \ = \ \forall i.\ \big((\chi_1 \otimes I) * I\, i \to \exists j \geq i.\ ((\chi_2 \otimes I) * I\, j)\big) \tag{33}$$

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

$$\textsc{Generalized frame}\quad \frac{\Gamma \Vdash t : \chi}{\Gamma \otimes I * I\,i \Vdash t : \exists j \geq i.\, (\chi \otimes I) * I\,j} \qquad \textsc{Generalized anti-frame}\quad \frac{\Gamma \otimes I \Vdash t : \exists i.\, (\chi \otimes I) * I\,i}{\Gamma \Vdash t : \chi}$$

Figure 11. Generalized frame and anti-frame rules

to the quantification over i. The generalized frame and anti-frame rules are given in Figure 11.

We now show how to extend our model of the type and capability calculus to accommodate 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.

8.1. **Generalized recursive 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 6, we define the worlds for the Kripke model in two steps, starting from an equation without any monotonicity requirements:[5] **CBUlt** has all non-empty coproducts, and there is a unique solution to the two equations

$$X \cong \sum_{\alpha \in \mathcal{K}^*} X_\alpha\ , \qquad X_{\kappa_1,\ldots,\kappa_n} = (\kappa_1 \times \cdots \times \kappa_n) \to (\tfrac{1}{2} \cdot X \to \mathit{URel}(\mathit{Heap}))\ , \tag{34}$$

with isomorphism $\iota : \sum_{\alpha \in \mathcal{K}^*} X_\alpha \to X$ in **CBUlt**, 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 6, X consists of all these families rather than individual predicates.

Note that, by definition of the metric on X, if $x \overset{n}{=} x'$ holds for $n > 0$ and $x = \iota\langle \alpha, g\rangle$ and $x' = \iota\langle \alpha', g'\rangle$, then $\alpha = \alpha'$ and $g\,i \overset{n}{=} g'\,i$ for all $i \in \alpha$.

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

$$g(i_1 i_2)(x) \ = \ g_1(i_1)(x_2 \circ x) * g_2(i_2)(x)\ .$$

for $i_1 \in \alpha_1$, $i_2 \in \alpha_2$. That is, the combination of an α_1-indexed family g_1 and an α_2-indexed family g_2 is a family g 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. The composition operation is defined as the fixed point of a contractive function as in Lemma 9, it can be shown associative, and it has a left and right unit given by $\mathit{emp} = \iota(\langle \varepsilon, I\rangle)$. For $g : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ we define the extension operation $(g \otimes x)(x') = f(x \circ x')$

8.2. **Generalized hereditarily monotone recursive worlds.** We will proceed as in Section 6, and carve out a subset of recursive worlds that satisfy a monotonicity condition.

To prove soundness of the anti-frame rule, and more specifically to establish the existence of commutative pairs, we need to know that the order in which the invariant families appear is irrelevant for the semantics of types and capabilities. The requirement is made precise by considering a partial equivalence relation $\sim$ on X, where $\iota(\langle \alpha_1\alpha_2, g\rangle) \sim \iota(\langle \alpha_2\alpha_1, h\rangle)$ holds if $g(i_1 i_2)(x_1) = h(i_2 i_1)(x_2)$ for all $i_1 \in \alpha_1$, $i_2 \in \alpha_2$ and $x_1 \sim x_2$, and insisting that semantic operations respect this

[5] We believe that a variant of the inverse-limit construction in Section 5 could also be used to construct the worlds, but we have not checked all the details.

relation. Note that the relation $\sim$ is recursive; we define it as the fixed point of a function Ψ on the non-empty and closed subsets of $X \times X$.

Definition 28. Let $\Psi : \mathcal{R}(X \times X) \to \mathcal{R}(X \times X)$ be defined as follows. For all $x, y \in X$ where $x = \iota\langle \alpha, g\rangle$ and $y = \iota\langle \beta, h\rangle$, $(x, y) \in \Psi(R)$ if and only if

- there exists $n \in \mathbb{N}$ and a permutation π of $1, \ldots, n$ such that $\alpha = \alpha_1 \ldots \alpha_n$ and $\beta = \alpha_{\pi(1)} \ldots \alpha_{\pi(n)}$; and
- for all $i_1 \in \alpha_1, \ldots, i_n \in \alpha_n$ and all $z, z' \in X$, if $(z, z') \in R$ then $g(i_1 \ldots i_n)(z) = h(i_{\pi(1)} \ldots i_{\pi(n)})(z')$.

The function Ψ is contractive, and we define $\sim \subseteq X \times X$ as its unique fixed point in $\mathit{URel}(X \times X)$, by the Banach fixed point theorem.

Lemma 29. *$\sim$ is a partial equivalence relation on X:*

(1) *$x \sim y$ implies $y \sim x$;*
(2) *$x \sim y$ and $y \sim z$ implies $x \sim z$.*

Proof. Since $(\sim_{[n]})_n$ is a Cauchy chain in $\mathcal{R}(X \times X)$ with limit $\sim$ given as the intersection of the $\sim_{[n]}$, part (1) of the lemma follows from the claim:

$$\forall n \in \mathbb{N}.\ \forall xy \in X.\ x \sim y \ \Rightarrow\ (y, x) \in \sim_{[n]}\ ,$$

which is proved by induction on n.

The case $n = 0$ is immediate since $\sim_{[0]} = X \times X$. For the case $n > 0$ let $x \sim y$. For simplicity, we assume $x = \iota\langle \alpha_1\alpha_2, p\rangle$ and $y = \iota\langle \alpha_2\alpha_1, q\rangle$. To prove $(y, x) \in \sim_{[n]}$ it suffices to show that $y' \sim x'$ holds for $y' = \iota\langle \alpha_2\alpha_1, q'\rangle$ and $x = \iota\langle \alpha_1\alpha_2, p'\rangle$ with $q'(i_2i_1)(z) = q(i_2i_1)(z)_{[n]}$ and $p'(i_1i_2)(z) = p(i_1i_2)(z)_{[n]}$, since $(y, x) \overset{n}{=} (y', x')$. To this end, let $i_2 \in \alpha_2$, $i_1 \in \alpha_1$, and suppose that $z \sim z'$; we must prove $q'(i_2i_1)(z) = p'(i_1i_2)(z')$. By induction hypothesis, $(z', z) \in \sim_{[n-1]}$, i.e., there exists $u' \sim u$ with $u' \overset{n-1}{=} z'$ and $u \overset{n-1}{=} z$ in X. Note that this means $u' \overset{n}{=} z'$ and $u \overset{n}{=} z$ holds in $\frac{1}{2} \cdot X$. Thus

$$q(i_2i_1)(z) \overset{n}{=} q(i_2i_1)(u) = p(i_1i_2)(u') \overset{n}{=} p(i_1i_2)(z')$$

by the non-expansiveness of p, q, and by the assumption $x \sim y$. It follows that

$$q'(i_2i_1)(z) = q(i_2i_1)(z)_{[n]} = p(i_1i_2)(u')_{[n]} = p'(i_1i_2)(z')$$

i.e., we have shown $y' \sim x'$.

Part (2) follows from a similar argument, proving that for all n, $x \sim y$ and $y \sim z$ implies $(x, z) \in \sim_{[n]}$. □

The composition operation respects this partial equivalence relation.

Lemma 30. *If $x \sim x'$ and $y \sim y'$ then $x \circ y \sim x' \circ y'$.*

Proof sketch. Similar to the proof of Lemma 13: We prove by induction that for all $n \in \mathbb{N}$, if $x \sim x'$ and $y \sim y'$ then $(x \circ y, x' \circ y') \in \sim_{[n]}$, and use that $\sim$ is the intersection of all the $\sim_{[n]}$. □

Next, we define the hereditarily monotone worlds. We ensure that these worlds w respect $\sim$ by requiring that they be self-related. The set $W \subseteq X$ of these worlds is again defined as fixed point of a contractive function, on the closed and non-empty subsets of X.

Definition 31 (Generalized hereditarily monotone worlds)**.** Let $\Phi : \mathcal{R}(X) \to \mathcal{R}(X)$ be defined as follows. For all $w \in X$ where $w = \iota\langle \alpha, g\rangle$, $w \in \Phi(R)$ if and only if

- $w \sim w$; and
- for all $i \in \alpha$ and all $w_1, w_2 \in R$, $g(i)(w_1) \subseteq g(i)(w_1 \circ w_2)$.

The function Φ is contractive, and we define the hereditarily monotone functions $W = \mathit{fix}(\Phi) = \Phi(W)$ by the Banach fixed point theorem.

Using Lemmas 29 and 30 it is not difficult to see that W is closed under the relation $\sim$. Moreover, as in Section 6, the composition operation restricts to the subset of hereditary monotone worlds.

Lemma 32. *If* $w_1, w_2 \in W$ *then* $w_1 \circ w_2 \in W$.

Proof sketch. As in the proof of Lemma 13, we show that $x, y \in W$ implies $x \circ y \in W_{[n]}$ for all $n \in \mathbb{N}$ by induction on n. Lemma 30 is used to show the additional requirement that the composition of $x, y \in W$ is self-related, $x \circ y \sim x \circ y$. □

8.3. **Semantics of capabilities and types.** The semantic domains for the interpretation of capabilities and types, with respect to the generalized worlds, now consist of the world-dependent functions that are both monotonic (with respect to the generalized hereditarily monotone worlds) and respect the relation $\sim$. More precisely, for a preordered set A we define $\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(A)$ to consist of all those $g : \frac{1}{2} \cdot X \to \mathit{URel}(A)$ where

- $\forall x, x' \in X.\ x \sim x' \ \Rightarrow\ g(x) = g(x')$;
- $\forall w_1, w_2 \in W.\ g(w_1) \subseteq g(w_1 \circ w_2)$.

Then we write

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

Note that with this definition, $g \in \kappa \to \mathit{Cap}$ if and only if $\iota(\langle \kappa, g \rangle) \in W$.

To define the interpretation of types, we first consider the following extension of memory types from values to expressions. Compared to the corresponding Definition 15 in Section 7, the extension now depends on the parameter $i \in \alpha$.

Definition 33 (Expression typing)**.** Let f in $\frac{1}{2} \cdot W \to_{mon} \mathit{URel}(\mathit{Val} \times \mathit{Heap})$. Let $x \in X$ and $\langle \alpha, p \rangle = \iota^{-1}(x)$. Let $i \in \alpha$. Then $\mathcal{E}(f, x, i) \subseteq \mathit{Exp} \times \mathit{Heap}$ is defined by $(k, (t, h)) \in \mathcal{E}(f, x, i)$ if and only if

$$\begin{aligned} &\forall j \leq k, t', h'.\ (t \,|\, h) \longmapsto^j (t' \,|\, h') \ \wedge\ (t' \,|\, h') \text{ irreducible} \\ &\quad \Rightarrow\ (k{-}j, (t', h')) \in \bigcup_{w \in W,\ \langle \alpha\beta, q \rangle = \iota^{-1}(x \circ w),\ i_1 \geq i,\ i_2 \in \beta} f(x \circ w) * q(i_1 i_2)(\mathit{emp})\ . \end{aligned}$$

This definition is well-behaved, in the sense that $\mathcal{E}(f, x, i) \subseteq \mathit{Exp} \times \mathit{Heap}$ is a uniform subset (with respect to the discrete order on $\mathit{Exp} \times \mathit{Heap}$, that it is non-expansive as a function in x, and that $x \sim x'$ implies $\mathcal{E}(f, x, i) = \mathcal{E}(f, x', i')$ for a suitable reordering i' of the parameters i.

Corresponding to the distribution axiom (33), the interpretation of arrow types bakes in the property that state changes on local state are captured by the local invariants: given $x \in X$, $(k, \mathsf{fun}\, f(y){=}t) \in (f_1 \to f_2)(x)$ if and only if

$$\begin{aligned} &\forall j < k.\ \forall w \in W \text{ where } \iota^{-1}(x \circ w) = \langle \alpha, p \rangle.\ \forall r \in \mathit{URel}(\mathit{Heap}).\ \forall i \in \alpha.\ \forall v, h. \\ &(j, (v, h)) \in f_1(x \circ w) * p(i)(\mathit{emp}) * r \ \Rightarrow \\ &\qquad (j, t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathcal{E}(f_2 * r, x \circ w, i)\ . \end{aligned}$$

Semantic operations corresponding to the other capability and type constructors can be defined analogous to Definition 17. It is easy to see that these operations respect the relation $\sim$. In fact, the only case that makes direct use of the parameter $x \in W$ is the case of arrow types above where one quantifies universally over the elements of all instances of its precondition p and (via $\mathcal{E}$) existentially over the elements of instances of its postcondition q; by definition of $\sim$ all these instances do not depend on reordering of the parameter.

As in Section 7, (semantic variants of) the distribution axioms for generalized invariants can be justified with respect to these operations. In particular, the axiom (33) holds since, given $c \in \kappa \to \mathit{Cap}$ and setting $w \overset{\mathit{def}}{=} \iota(\langle \kappa, c\rangle)$,

$$(f_1 \to f_2) \otimes w = \forall_{i \in \kappa}\big((f_1 \otimes w) * c\, i) \to \exists_{j \geq i}((f_2 \otimes w) * c\, j)\big)$$

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

The semantics of value judgements $\Delta \vdash v : \tau$ looks as before. The semantics of the expression typing judgement mirrors the new interpretation of arrow types, in the sense that there is now also a universal quantification over all possible instances i of the invariant family p represented by a world $w \in W$:

$$\begin{aligned}\models\ (\Gamma \Vdash t : \chi) \quad \iff \quad & \forall \eta.\ \forall w \in W \text{ where } w = \langle \alpha, p\rangle.\ \forall k \in \mathbb{N}.\\ & \forall i \in \alpha.\ \forall r \in \mathit{URel}(\mathit{Heap}).\forall (k, (\rho, h)) \in [\![\Gamma]\!]_\eta\, w * p(i)(\mathit{emp}) * r.\\ & \qquad\qquad (k, (\rho(t), h)) \in \mathcal{E}([\![\chi]\!]_\eta * r, w, i).\end{aligned}$$

We can now prove soundness of the generalized rules.

Theorem 34 (Soundness)**.** *The generalized frame and anti-frame rules 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 [22] is sound.

Proof sketch. The case of the generalized frame rule is similar to the proof of Lemma 23.

The soundness proof for the generalized anti-frame rule rests again on the existence of commutative pairs. Compared to the earlier Lemma 24, however, we can only prove a variant which states that commutativity holds up to the relation $\sim$: Let $w_0, w_1 \in W$ be families indexed over α_0 and α_1, i.e., $\iota^{-1}(w_0) = \langle \alpha_0, p_0\rangle$ and $\iota^{-1}(w_1) = \langle \alpha_1, p_1\rangle$ for some p_0 and p_1. Then there exist $w_0', w_1' \in W$ such that

$$\begin{aligned}& w_0' = \iota\langle \alpha_0, \lambda i.(p_0\, i) \otimes w_1'\rangle,\\ & w_1' = \iota\langle \alpha_1, \lambda i.(p_1\, i) \otimes w_0'\rangle, \text{ and}\\ & w_0 \circ w_1' \sim w_1 \circ w_0\ .\end{aligned}$$

Since we insisted that the interpretations of types and capabilities respect $\sim$, this variant is sufficient to prove the soundness of the generalized anti-frame rule analogously to the proof of Lemma 25. □

9. Conclusion and Future Work

We have developed a soundness proof of the frame and anti-frame rules in the expressive type and capability system of Charguéraud and Pottier, by constructing a Kripke model of the system. For our model, we have presented two novel approaches to construct the recursively defined set of worlds.[6] The first approach is a (tedious) construction of an inverse limit in the category of complete, 1-bounded ultrametric spaces. In the second approach one defines the worlds as a recursive subset of a recursively defined metric space. This construction is simpler than the inverse limit construction, but requires an additional argument to show that the semantic operations restrict to this subset. We have demonstrated that this approach scales, by also extending the model to show soundness of Pottier's generalized frame and

[6] An interesting challenge would be to find a general existence theorem for solutions of recursive domain equations that can deal with the recursive monotonic worlds.

anti-frame rules. More generally, we believe that the recursive worlds constructed in Sections 5 and 6 can be used, possibly in variations, to model various type system and program logics with hidden (higher-order) state.

Future work includes exploring some of the orthogonal extensions of the basic type and capability system that have been proposed in the literature: group regions [11], and fates and predictions [19]. The model that we have presented suggests to include separation logic assertions in the syntax of capabilities, and it would be interesting to work out such a program logic in detail.

Recently, Pottier has given an alternative soundness proof for a slightly different language, including group regions but not the generalized frame and anti-frame rules. This proof is based on progress and preservation properties, and has been formalized in the Coq proof assistant [24]. While we have not attempted a formalization of our model, we believe that this is possible based on the results of Varming et al. [4].

References

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

[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] A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.

[4] N. Benton, L. Birkedal, A. Kennedy, and C. Varming. Formalizing domains, ultrametric spaces and semantics of programming languages. 2010. Draft.

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

[6] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. In *Proceedings of POPL*, pages 119–132, 2011.

[7] L. Birkedal, B. Reus, J. Schwinghammer, and H. Yang. A simple model of separation logic for higher-order store. In *Proceedings of ICALP*, pages 348–360, 2008.

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

[9] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations. *Theor. Comput. Sci.*, 411(47):4102–4122, 2010.

[10] 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.

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

[12] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. In *Proceedings of ICFP*, 2010.

[13] P. B. Levy. Possible world semantics for general storage in call-by-value. In *Proceedings of CSL*, pages 232–246, 2002.

[14] A. Nanevski, A. Ahmed, G. Morrisett, and L. Birkedal. Abstract predicates and mutable ADTs in Hoare type theory. In *Proceedings of ESOP*, pages 189–204, 2007.

[15] P. W. O'Hearn, H. Yang, and J. C. Reynolds. Separation and information hiding. In *Proceedings of POPL*, pages 268–280, 2004.

[16] M. Parkinson and G. Bierman. Separation logic and abstraction. In *Proceedings of POPL*, pages 247–258, 2005.

[17] M. Parkinson and G. Bierman. Separation logic, abstraction and inheritance. In *Proceedings of POPL*, pages 75–86, 2008.

[18] B. C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

[19] A. Pilkiewicz and F. Pottier. The essence of monotonic state. In *Proceedings of TLDI*, pages 73–86, 2011.

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

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

[22] F. Pottier. Generalizing the higher-order frame and anti-frame rules. Unpublished note, available at `http://gallium.inria.fr/~fpottier`, July 2009.

[23] F. Pottier. Three comments on the anti-frame rule. Unpublished note, available at `http://gallium.inria.fr/~fpottier`, July 2009.

[24] F. Pottier. Syntactic soundness proof of a type-and-capability system with hidden state. Unpublished note, May 2011.

[25] D. J. Pym, P. W. O'Hearn, and H. Yang. Possible worlds and resources: the semantics of BI. *Theor. Comput. Sci.*, 315(1):257–305, 2004.

[26] J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of LICS*, pages 55–74, 2002.

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

[28] J. Schwinghammer, L. Birkedal, and K. Støvring. A step-indexed Kripke model of hidden state via recursive properties on recursively defined metric spaces. In *Proceedings of FOSSACS*, pages 305–319, 2011.

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

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

Saarland University, Saarbrücken

IT University of Copenhagen

INRIA

University of Sussex, Brighton

University of Copenhagen

Queen Mary University, London