Information and Computation 249 (2016) 160–189

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

A Kripke logical relation for effect-based program transformations ☆

Lars Birkedal [a,*], Guilhem Jaber [b,1], Filip Sieczkowski [a,2,*], Jacob Thamsborg [c]

[a] Aarhus University, Denmark
[b] Ecole des Mines de Nantes, France
[c] IT University of Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 2 July 2013
Received in revised form 22 October 2014
Available online 3 May 2016

Keywords:
Effect type system
Kripke logical relation
Program transformation

ABSTRACT

We present a Kripke logical relation for showing the correctness of program transformations based on a region-polymorphic type-and-effect system for an ML-like programming language with higher-order store and dynamic allocation. We also show how to use our model to verify a number of interesting program transformations that rely on effect annotations.

In building the model, we extend earlier work by Benton *et al.* that treated, respectively dynamically allocated first-order references, and higher-order store for global variables. We utilize ideas from region-based memory management, and on Kripke logical relations for higher-order store.

One of the key challenges that we overcome in the construction of the model is treatment of *masking* of regions (conceptually similar to deallocation). Our approach bears similarities to the one used in Ahmed's unary model of a region calculus in her Ph.D. thesis.

1. Introduction

A type system for a programming language classifies programs according to properties that the programs satisfy. An effect system is a type system that, in particular, classifies programs according to which side effects the programs may have. A variety of effect systems have been proposed for higher-order programming languages, e.g., [2–4], see [5] for a recent overview. Effect systems can often be understood as specifying the results of a static analysis, in the sense that it is possible to automatically infer types and effects. Effect systems can be used for different purposes: they were originally proposed by Gifford and Lucassen in [2] as a means of combining functional and imperative features of a language, and for parallelization purposes, but since then they have also been used as the basis for implementing ML using a stack of regions for memory management [4,6] and for region-based memory management in Cyclone, a safe dialect of C [7]. Lately, they have also been used for termination analysis for higher-order concurrent programs [8,9], and for ensuring determinism of parallel

☆ An earlier version of this paper was presented at the ICFP 2011 conference [1]. In addition to including more proofs and examples, the present paper also includes a treatment of an extension of the type system with region polymorphism, which was not covered in [1].

* Corresponding authors. Postal address: Dept. of Comp. Science, Aarhus University, Aabogade 34, DK-8200 Aarhus N, Denmark.
E-mail addresses: birkedal@cs.au.dk (L. Birkedal), guilhem.jaber@mines-nantes.fr (G. Jaber), filips@cs.au.dk (F. Sieczkowski), thamsborg@itu.dk (J. Thamsborg).
[1] Current affiliation: Université Paris 7, France.
[2] Current affiliation: INRIA Paris, France.

http://dx.doi.org/10.1016/j.ic.2016.04.003
0890-5401/

programs [10,11]. In a recent series of papers, Benton et al. have argued that another important point of effect systems is that they can be used as the basis for effect-based program transformations, in particular compiler optimizations [12–15]. The idea is that certain program transformations are only sound under additional assumptions about which effects program phrases may, or rather may not, have. For example, in a higher-order language with references it is only sound to hoist an expression out of a lambda abstraction if it is known that the expression neither allocates new references, nor reads or writes references.

While it is intuitively clear that effect information is important for validating program transformations, it is surprisingly challenging to develop semantic models that can be used to rigorously justify effect-based transformations. In earlier work, Benton et al. developed semantic relational models of effect systems for a higher-order language with dynamically allocated first-order references (ground store) [14] and for a higher-order language with global variables for higher-order store (but no dynamic allocation) [15].

In this paper we present a Kripke logical relations model of a region-based effect system for a higher-order language with higher-order store *and* dynamic allocation, i.e., with general ML-like references. As pointed out in [15], this is a particularly challenging extension and one that is important for soundness of effect-based transformations for realistic ML-like languages. We now explain what the main challenges are; in Section 3 we give an intuitive overview of how we address these challenges.

The main challenge arises from effect masking: Our region-based effect system includes an effect masking rule that enables hiding local uses of effects. This makes it possible to view a computation as pure even if it uses effects locally and makes the effect system stronger (it can justify more program transformations). To model effect masking we need to model references that, conceptually speaking, become dangling because they point into a region that is masked away. We say "conceptually speaking" because in the operational semantics there is no deallocation of references (the operational semantics is completely standard). However, in our model we have to reason *as if* regions could actually be allocated and deallocated.

Here is a simple concrete example: Consider the following expression e, typed as indicated[3] (where 1 denotes the unit type):

$$\texttt{let x} = \texttt{ref}_\rho\ 7\ \texttt{in let y} = \texttt{ref}_\sigma\ \texttt{x in}$$
$$\lambda z.\texttt{y} := \texttt{x} : 1 \to^{\{wr_\sigma\}} 1, \{al_\sigma\}$$

The program allocates two references, binds them to x and y, and then returns a trivial function from unit to unit that assigns x to y. The types indicate that x will be bound to a location in region ρ, say location 0, and that y will be bound to a location in region σ, say location 1. At location 0 the value 7 is stored and at location 1 the location 0 is stored. The so-called latent effect $\{wr_\sigma\}$ of the function type of the whole expression describes which effects the function may have when called. Finally the entire expression has the effect $\{al_\sigma\}$ as it allocates a location in region σ. It performs allocation in region ρ as well, but this effect has been masked out: since no ρ appears in the return type, the expression cannot leak location 0, and we can, conceptually, deallocate all locations in region ρ once the computation has run; this is what the masking rule captures. This particular example, however, was chosen to stress test the masking rule, as location 0 is in fact leaked: it is in the function closure that the expression returns. The function neither reads not writes location 0, but it does write it to the heap; our model must be able to cope with such conceptually dangling pointers. On the level of types, the meaning of the type $\mathsf{ref}_\rho\mathsf{int}$ of x changes: after the allocation of values for x and y, it contains location 0, but what should it contain after region ρ has been masked out? If, e.g., $\mathsf{ref}_\rho\mathsf{int}$ is taken to be empty, then the function is most likely no longer well-typed. We explain our answer to this question in Section 3. Since we have to reason as if regions are deallocated, it is, in hindsight, not so surprising that our approach is closely related to the one used by Ahmed in her unary model of a region calculus with region deallocation [16].

One could argue that the program above should in fact be treated as semantically pure, even if the type system cannot infer this fact, since the location bound to y is never made accessible to the context. Thus, one might want to provide a stronger masking principle on the semantic side. We do not pursue that goal in the current paper and hence our semantic treatment of masking mimics the syntactic masking rule. In future work, it will be interesting to investigate a more liberal notion of semantic masking.

Note that the effect annotations in the types are just annotations; the operational semantics is completely standard and regions only exist in our semantic model, not in the operational semantics. Further note that the issues in the above example do not arise for a language with only ground store.

Another challenge arises from the fact that since our language includes dynamically allocated general references, the existence of the logical relation is non-trivial; in particular, the set of worlds must be recursively defined. Here we define the worlds as a solution to a recursive metric-space equation, building on our earlier work [17], which gives a unified account of models based on step-indexing [16] and on domain theory.

Yet another challenge arises from the fact that our language includes a new form of region polymorphism, inspired by [3], but not covered before in semantic models for validating effect-based program equivalences. Region polymorphism makes

[3] Note that conceptually the whole expression could be considered pure: none of its effects can actually be observed by any context. However, our model and type-and-effect system are not fine-grained enough to express this.

the type system significantly more expressive (fewer regions are collapsed, so more program transformations will hold). However, combining region polymorphism and program equivalence is not straightforward. Indeed, consider the following example:

$$\lambda x.\lambda y.\, x := 0;\ y := 1 \approx \lambda x.\lambda y.\, y := 1;\ x := 0 : \mathsf{ref}_\rho \mathsf{int} \to \mathsf{ref}_\sigma \mathsf{int} \to^{wr_\rho, wr_\sigma} 1$$

Both sides take two references as arguments, one in region ρ and one in region σ, and assigns to them. The order of assignments is different though, but as the references belong to different regions, they cannot be identical and so the two sides are indeed equivalent. We would like this equivalence also to hold at region-polymorphic type $\forall \rho, \sigma.\mathsf{ref}_\rho \mathsf{int} \to \mathsf{ref}_\sigma \mathsf{int} \to^{wr_\rho, wr_\sigma} 1$. But this would be true only if ρ and σ cannot be instantiated to the same region. To express this, we use a restricted form of region polymorphism, $\forall \rho \notin \Pi.\, \tau^\varepsilon$, for which region variable ρ can only be instantiated by regions not in Π. For the example above, the equivalence will then hold at the type $\forall \rho. \forall \sigma \notin \{\rho\}.\mathsf{ref}_\rho \mathsf{int} \to \mathsf{ref}_\sigma \mathsf{int} \to^{wr_\rho, wr_\sigma} 1$.

We should stress the practical importance of region polymorphism for typing the impure library functions. Consider the function

$$\mathtt{swap} \equiv \lambda x.\lambda y.\, \mathtt{let}\ v = !x\ \mathtt{in}\ x := !y;\ y := v,$$

and the application $\mathtt{swap}\, r_1\, r_2;\ \mathtt{swap}\, r_3\, r_4$. Without the region polymorphism, we can type the function as $\mathtt{swap} : \mathsf{ref}_\rho \mathsf{int} \to \mathsf{ref}_\rho \mathsf{int} \to^{rd_\rho, wr_\rho} 1$. However, this requires the references r_1 and r_3, which appear at different call sites to the library function, to be in the same region ρ. In fact, this treatment leads to an unnecessary collapsing of many regions together, which can lead to program transformations being inapplicable. If, however, we can give the library functions types that are polymorphic in the memory regions they access, the different call sites can use different instantiations and the regions can be much more fine-grained, allowing more program transformations.

One may worry whether our resulting model now becomes much more complicated than the earlier, already non-trivial, models of Benton et al. [14,15]. As we shall explain in Section 8, that is not the case. Indeed, our model is arguably a bit simpler even though it applies to a richer language. Moreover, it also becomes simpler to verify equivalences using the model.

Our relational model is built directly over the operational semantics, using our metric approach to step-indexing. We could also have defined the model using a denotational model of the programming language; see the discussion in Section 8 — here we preferred the operational approach because it is perhaps more widely accessible.

We use our model to validate a number of interesting effect-based program transformations that, to the best of our knowledge, have not been proved correct before, see Section 7.

To focus the presentation on the core challenges, we here consider a monomorphically typed higher-order programming language with general references, but leave out universal and existential types as well as recursive types. However, we want to stress that since our semantic techniques (step-indexed Kripke logical relations over recursively defined worlds) do indeed scale well to universal, existential, and recursive types, e.g. [16,18,17], it is straightforward to extend our model to a language with such types.[4]

2. Language and effect system

2.1. Syntax

The expressions and values of our model language are defined in the grammar below. We use e to range over the set of expressions $\mathcal{E}$, v to range over the set of values $\mathcal{V}$, l to range over a countably infinite set of locations $\mathcal{L}$, and n to range over integers.

$$\begin{aligned}
e &::= x \mid l \mid n \mid () \mid \langle e_1, e_2\rangle \mid \mathsf{proj}_1\, e \mid \mathsf{proj}_2\, e \mid \mathsf{fix}\, f(x).e \mid e_1\, e_2 \\
&\quad \mid \mathsf{susp}\, e \mid \mathsf{force}\, e \mid \mathsf{ref}\, e \mid e_1 := e_2 \mid !e \\
v &::= l \mid n \mid () \mid \langle v_1, v_2\rangle \mid \mathsf{fix}\, f(x).e \mid \mathsf{susp}\, e \\
E &::= [\,] \mid \langle E, e\rangle \mid \langle v, E\rangle \mid \mathsf{proj}_1\, E \mid \mathsf{proj}_2\, E \mid E\, e \mid v\, E \mid \mathsf{force}\, E \\
&\quad \mid \mathsf{ref}\, E \mid E := e \mid v := E \mid !E
\end{aligned}$$

The only non-standard constructs are $\mathsf{susp}\, e$ and $\mathsf{force}\, e$. Intuitively, $\mathsf{susp}\, e$ suspends the computation e, and $\mathsf{force}\, e$ forces the suspension that e evaluates to. These constructs are used in connection with region abstraction and instantiation, as explained in Section 2.3.

[4] With type abstractions as values, we would have a latent effect on a polymorphic type, which would thus be of the form $\forall \alpha.\tau^\varepsilon$.

2.2. Dynamic semantics

$$\langle E[\mathsf{proj}_1\,\langle v_1, v_2\rangle]\,|\,h\rangle \to \langle E[v_1]\,|\,h\rangle$$
$$\langle E[\mathsf{proj}_2\,\langle v_1, v_2\rangle]\,|\,h\rangle \to \langle E[v_2]\,|\,h\rangle$$
$$\langle E[(\mathsf{fix}\,f(x).e)\,v]\,|\,h\rangle \to \langle E[e[\mathsf{fix}\,f(x).e/f, v/x]]\,|\,h\rangle$$
$$\langle E[\mathsf{force}\,(\mathsf{susp}\,e)]\,|\,h\rangle \to \langle E[e]\,|\,h\rangle$$
$$\langle E[\mathsf{ref}\,v]\,|\,h\rangle \to \langle E[l]\,|\,h[l \mapsto v]\rangle \quad \text{if } l \notin \mathrm{dom}(h)$$
$$\langle E[l := v]\,|\,h\rangle \to \langle E[()]\,|\,h[l \mapsto v]\rangle \quad \text{if } l \in \mathrm{dom}(h)$$
$$\langle E[!\,l]\,|\,h\rangle \to \langle E[h(l)]\,|\,h\rangle \qquad \text{if } l \in \mathrm{dom}(h)$$

Heaps $\mathcal{H}$ are finite maps from locations to closed values, and, for $e \in \mathcal{E}$, $h \in \mathcal{H}$ and $j \geq 0$, we write $\langle e\,|\,h\rangle \xrightarrow{j} \langle e'\,|\,h'\rangle$ if $\langle e\,|\,h\rangle$ reduces to $\langle e'\,|\,h'\rangle$ in j steps, counting all reduction steps. We write $\mathrm{irr}\langle e\,|\,h\rangle$ to state that no further reductions of $\langle e\,|\,h\rangle$ are possible, either because e is a value or because we are stuck. Runtime errors (such as trying to look up a nonexistent location or trying to apply a number as a function) are modeled by the evaluation getting stuck.

2.3. Type and effect system

Our effect system ensures that programs are well-typed in the standard sense of not getting stuck and, moreover, tracks dependencies and side-effects of computations on the heap.

The typing rules for our effect system are given in Fig. 1. Types are defined by the following grammar:

$$\delta ::= rd_\rho \mid wr_\rho \mid al_\rho$$
$$\varepsilon ::= \delta_1, \ldots, \delta_n$$
$$\tau ::= 1 \mid \mathsf{int} \mid \tau_1 \times \tau_2 \mid \mathsf{ref}_\rho\,\tau \mid \tau_1 \to^\varepsilon \tau_2 \mid \forall \rho \notin \Pi.\,\tau^\varepsilon$$

where ρ ranges over a countably infinite set of region variables $\mathcal{RV}$, Π is a finite subset of $\mathcal{RV}$ and ε ranges over the set of effects. An effect is a finite set of primitive effects, and a primitive effect is of one of the forms rd_ρ, wr_ρ, or al_ρ. The primitive effect rd_ρ specifies a read effect on region ρ. Likewise wr_ρ specifies a write effect and al_ρ specifies an allocation effect (that locations may be allocated in region ρ). Note that, as usual, the function type $\tau_1 \to^\varepsilon \tau_2$ includes a *latent* effect ε; this is the effect that the function may have when called. The main novelty of this system is the restricted form of region polymorphism $\forall \rho \notin \Pi.\,\tau^\varepsilon$, which restricts the possible instantiations of ρ to regions which are not in Π. In practice, as we will see, we restrict the choice of Π to be $\mathrm{FRV}(\tau, \varepsilon)\backslash\{\rho\}$.

We define the *free region variables* $\mathrm{FRV}(\tau)$ of a type τ:

$$\mathrm{FRV}(\delta) ::= \{\rho\} \text{ with } \delta \in \{rd_\rho, wr_\rho, al_\rho\}$$
$$\mathrm{FRV}(\delta_1, \ldots, \delta_n) ::= \mathrm{FRV}(\delta_1) \cup \ldots \cup \mathrm{FRV}(\delta_n)$$
$$\mathrm{FRV}(1) ::= \emptyset$$
$$\mathrm{FRV}(\mathsf{int}) ::= \emptyset$$
$$\mathrm{FRV}(\tau_1 \times \tau_2) ::= \mathrm{FRV}(\tau_1) \cup \mathrm{FRV}(\tau_2)$$
$$\mathrm{FRV}(\mathsf{ref}_\rho\,\tau) ::= \{\rho\} \cup \mathrm{FRV}(\tau)$$
$$\mathrm{FRV}(\tau_1 \to^\varepsilon \tau_2) ::= \mathrm{FRV}(\tau_1) \cup \mathrm{FRV}(\varepsilon) \cup \mathrm{FRV}(\tau_2)$$
$$\mathrm{FRV}(\forall \rho \notin \Pi.\,\tau^\varepsilon) ::= (\mathrm{FRV}(\tau) \cup \mathrm{FRV}(\varepsilon)) \setminus \{\rho\}$$

Thus ρ is bound in $\forall \rho \notin \Pi.\,\tau^\varepsilon$. Substitution in a type τ of a region variable σ for another region variable ρ is written $\tau[\sigma/\rho]$ and is defined as usual, avoiding capture of bound region variables in region polymorphic types.

Furthermore, we define a notion of *well-formed* types, denoted $\mathrm{wf}(\tau)$, which restricts the region quantification that can occur in τ. Intuitively, any region-polymorphic type within τ that binds a region ρ has to ensure that ρ is different from its free region variables. Formally, for any $\forall \rho \notin \Pi.\,\sigma^\varepsilon$ within τ, Π has to be the set $(\mathrm{FRV}(\tau) \cup \mathrm{FRV}(\varepsilon)) \setminus \{\rho\}$. The full definition follows.

$$\text{T-Ax}\ \frac{\mathrm{wf}(\tau), \mathrm{wf}(\Gamma) \qquad \mathrm{FRV}(\tau, \Gamma) \subseteq \Pi}{\Pi \mid \Gamma, x{:}\tau \vdash x : \tau, \emptyset}$$

$$\text{T-Unit}\ \frac{\mathrm{wf}(\Gamma) \qquad \mathrm{FRV}(\Gamma) \subseteq \Pi}{\Pi \mid \Gamma \vdash () : 1, \emptyset} \qquad \frac{\mathrm{wf}(\Gamma) \qquad \mathrm{FRV}(\Gamma) \subseteq \Pi}{\Pi \mid \Gamma \vdash n : \mathsf{int}, \emptyset}\ \text{T-Int}$$

$$\text{T-Pair}\ \frac{\Pi \mid \Gamma \vdash e_1 : \tau_1, \varepsilon_1 \qquad \Pi \mid \Gamma \vdash e_2 : \tau_2, \varepsilon_2}{\Pi \mid \Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2, \varepsilon_1 \cup \varepsilon_2} \qquad \frac{\Pi \mid \Gamma \vdash e : \tau_1 \times \tau_2, \varepsilon}{\Pi \mid \Gamma \vdash \mathsf{proj}_i\, e : \tau_i, \varepsilon}\ \text{T-Proj}_i$$

$$\text{T-Fix}\ \frac{\Pi \mid \Gamma, f : \tau_1 \to^{\varepsilon} \tau_2, x : \tau_1 \vdash e : \tau_2, \varepsilon}{\Pi \mid \Gamma \vdash \mathsf{fix}\, f(x).e : \tau_1 \to^{\varepsilon} \tau_2, \emptyset}$$

$$\frac{\Pi \mid \Gamma \vdash e_1 : \tau_1 \to^{\varepsilon} \tau_2, \varepsilon_1 \qquad \Pi \mid \Gamma \vdash e_2 : \tau_1, \varepsilon_2}{\Pi \mid \Gamma \vdash e_1\, e_2 : \tau_2, \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon}\ \text{T-App}$$

$$\text{T-Alloc}\ \frac{\Pi \mid \Gamma \vdash e : \tau, \varepsilon \qquad \rho \in \Pi}{\Pi \mid \Gamma \vdash \mathsf{ref}\, e : \mathsf{ref}_\rho \tau, \varepsilon \cup \{al_\rho\}}$$

$$\text{T-Deref}\ \frac{\Pi \mid \Gamma \vdash e : \mathsf{ref}_\rho \tau, \varepsilon}{\Pi \mid \Gamma \vdash\, !e : \tau, \varepsilon \cup \{rd_\rho\}} \qquad \frac{\Pi \mid \Gamma \vdash e_1 : \mathsf{ref}_\rho \tau, \varepsilon_1 \qquad \Pi \mid \Gamma \vdash e_2 : \tau, \varepsilon_2}{\Pi \mid \Gamma \vdash e_1 := e_2 : 1, \varepsilon_1 \cup \varepsilon_2 \cup \{wr_\rho\}}\ \text{T-Assign}$$

$$\text{T-RegGen}\ \frac{\Pi, \rho \mid \Gamma \vdash e : \tau, \varepsilon \qquad \rho \notin \mathrm{FRV}(\Gamma) \text{ and } \Delta = \mathrm{FRV}(\tau, \varepsilon) \backslash \{\rho\}}{\Pi \mid \Gamma \vdash \mathsf{susp}\, e : \forall \rho \notin \Delta.\, \tau^{\varepsilon}, \emptyset}$$

$$\text{T-RegInst}\ \frac{\Pi \mid \Gamma \vdash e : \forall \rho \notin \Delta.\, \tau^{\varepsilon}, \varepsilon' \qquad \sigma \in \Pi \backslash \Delta}{\Pi \mid \Gamma \vdash \mathsf{force}\, e : \tau[\sigma/\rho], \varepsilon[\sigma/\rho] \cup \varepsilon'}$$

$$\text{T-Masking}\ \frac{\Pi, \rho \mid \Gamma \vdash e : \tau, \varepsilon \qquad \rho \notin \mathrm{FRV}(\Gamma, \tau)}{\Pi \mid \Gamma \vdash e : \tau, \varepsilon - \rho}$$

$$\text{T-Sub}\ \frac{\Pi \mid \Gamma \vdash e : \tau_1, \varepsilon_1 \qquad \Pi \vdash \tau_1 \leq \tau_2 \qquad \varepsilon_1 \subseteq \varepsilon_2}{\Pi \mid \Gamma \vdash e : \tau_2, \varepsilon_2}\ (\mathrm{FRV}(\varepsilon_2) \subseteq \Pi)$$

Fig. 1. Type and effect system.

$$\begin{aligned}
\mathrm{wf}(1) &::= \mathit{true} \\
\mathrm{wf}(\mathsf{int}) &::= \mathit{true} \\
\mathrm{wf}(\tau_1 \times \tau_2) &::= \mathrm{wf}(\tau_1) \wedge \mathrm{wf}(\tau_2) \\
\mathrm{wf}(\mathsf{ref}_\rho \tau) &::= \mathrm{wf}(\tau) \\
\mathrm{wf}(\tau_1 \to^{\varepsilon} \tau_2) &::= \mathrm{wf}(\tau_1) \wedge \mathrm{wf}(\tau_2) \\
\mathrm{wf}(\forall \rho \notin \Pi.\, \tau^{\varepsilon}) &::= \mathrm{FRV}(\tau, \varepsilon) \setminus \{\rho\} = \Pi \wedge \mathrm{wf}(\tau)
\end{aligned}$$

Well-formedness is extended straightforwardly to typing contexts Γ.

Judgments are of the form: $\Pi \mid \Gamma \vdash e : \tau, \varepsilon$. Here Π is a region variable context (a finite set of region variables), Γ is a variable context (a finite map from variables to types), e is an expression, τ is the result type and ε is the effect of the computation. The typing rules are given in Fig. 1.

The effect system is fairly standard, except for the region polymorphism. Note the inclusion of the effect masking rule T-Masking; here $\varepsilon - \rho$ is defined as $\varepsilon - \{rd_\rho, wr_\rho, al_\rho\}$ and Π, ρ denotes the union of Π and $\{\rho\}$. We write $\mathrm{FRV}(\Gamma, \tau)$ for the region variables in the types of Γ and in τ. The masking rule expresses that to mask out primitive effects on ρ from ε, the region variable ρ must not be free in Γ, nor in τ. The idea is that in this case the remaining part of the computation cannot access ρ and thus we may hide the effects on ρ.

The rule T-RegGen forces the forbidden regions in Δ of the restricted polymorphism $\forall \rho \notin \Delta.\, \tau^{\varepsilon}$ to be equal to the free region variables of τ and ε. This ensures the well-formedness of $\forall \rho \notin \Delta.\, \tau^{\varepsilon}$. As we could expect, the rule T-RegInst prevents the instantiation of ρ by a region in Δ. Note the use of suspend ($\mathsf{susp}\, e$) and force ($\mathsf{force}\, e$) for region abstraction and instantiation; alternatively we could have used a value restriction.

Also note that there is a standard notion of effect subtyping, given by finite set inclusion, which also yields a notion of subtyping, defined by the rules in Fig. 2.

Consider the example expression e from the introduction. Note that it is well-typed (`let`-expressions are definable as usual) since the judgment

$$\{\sigma\} \mid \emptyset \vdash e : 1 \to^{wr_\sigma} 1, \{al_\sigma\}$$

$$\frac{}{\Pi \vdash \tau \leq \tau}\ (\mathrm{FRV}(\tau) \subseteq \Pi) \qquad \frac{\Pi \vdash \tau_1 \leq \tau_1' \qquad \Pi \vdash \tau_2 \leq \tau_2'}{\Pi \vdash \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'}$$

$$\frac{\Pi \vdash \tau_1' \leq \tau_1 \qquad \Pi \vdash \tau_2 \leq \tau_2' \qquad \varepsilon_1 \subseteq \varepsilon_2}{\Pi \vdash \tau_1 \to^{\varepsilon_1} \tau_2 \leq \tau_1' \to^{\varepsilon_2} \tau_2'}\ (\mathrm{FRV}(\varepsilon_2) \subseteq \Pi)$$

$$\frac{\Pi, \rho \vdash \tau_1 \leq \tau_2 \qquad \varepsilon_1 \subseteq \varepsilon_2}{\Pi \vdash \forall \rho \notin \Delta_1.\, \tau_1{}^{\varepsilon_1} \leq \forall \rho \notin \Delta_2.\, \tau_2{}^{\varepsilon_2}}\ (\mathrm{FRV}(\varepsilon_2) \subseteq \Pi, \rho \text{ and } \mathrm{wf}(\forall \rho \notin \Delta_i.\, \tau_i{}^{\varepsilon_i}))$$

Fig. 2. Subtyping rules.

is derivable in the effect system. The last rule applied is the T-Masking rule, which enables hiding the local effect on ρ. Specifically, we apply the masking rule like this:

$$\frac{\{\sigma, \rho\} \mid \emptyset \vdash e : 1 \to^{wr_\sigma} 1, \{al_\rho, al_\sigma\}}{\{\sigma\} \mid \emptyset \vdash e : 1 \to^{wr_\sigma} 1, \{al_\sigma\}}$$

Thus we hide the allocation and use of the local reference bound to x.

Effect masking makes it possible to do more optimizations: consider the familiar example of an efficient implementation *fib* of the Fibonacci function using two local references. We can use the masking rule to give it type and effect $\mathsf{int} \to^{\emptyset} \mathsf{int}, \emptyset$. This allows us to view the imperative implementation as pure, and thus, e.g., by Theorem 30 we find that it is sound to optimize two identical calls to *fib* into one call. This sounds like a simple optimization, but the point is that a compiler can perform it automatically, just based on the effect types — and our model justifies that it is sound to do so. See [14,15] for more examples.

Note, though, that these earlier works either have ground store (only integers in memory) or no dynamic allocation, so many non-trivial uses of memory cannot be expressed. By contrast, we can write, say, a function that sorts a (functional) list of integers by doing an internal (imperative) heap sort, i.e., by building a heap in fresh memory, and returning a sorted list. Using the masking rule, we could type such a function as a pure function with no effects, and would know, e.g., that sorting the same list twice serves no purpose.

We end this section by recording some simple facts about the type system (all proved by induction).

Lemma 1. *If* $\mathrm{wf}(\tau)$ *then* $\mathrm{wf}(\tau[\sigma/\rho])$.

Lemma 2. *If* $\Pi \vdash \tau_1 \leq \tau_2$ *then* $\mathrm{wf}(\tau_1)$ *iff* $\mathrm{wf}(\tau_2)$.

Proposition 3. *If* $\Pi \mid \Gamma \vdash e : \tau, \varepsilon$ *then* $\mathrm{wf}(\tau)$, $\mathrm{wf}(\Gamma)$ *and* $\mathrm{FRV}(\tau, \varepsilon, \Gamma) \subseteq \Pi$.

3. Overview of the technical development

Our starting point is the Kripke logical relation reading of types for an ML-like programming language that has been explored extensively by the authors and others [18,19,17,20]. The common approach is to augment the semantics — be it operational or denotational — with *worlds* that keep track of the layout of the heap: it knows the types of values stored at the allocated locations of the heap.[5] We then index type interpretations by worlds; this makes it possible to interpret $\mathtt{ref}\,\tau$ as the set of allocated locations that hold values of type τ. A computation has the "precondition" that all locations hold well-typed values according to the world, and this property must be re-established after running the computation, i.e., it also serves as a "postcondition". As more locations are allocated, the world grows and the interpretation of types is set up to grow as well: it is a crucial property that interpreting types in future worlds yields more (or at least no fewer) values, this we refer to as *type monotonicity*.

To extend this approach to the present region-based type-and-effect system, the worlds have to be partitioned into regions and interpreting $\mathsf{ref}_\rho\,\tau$ only considers region ρ of the world for locations that hold values of type τ. Also, computations now have effects and hence their behavior is more restricted: as "precondition" they only assume well-typedness of values at locations in the regions with read effects; as "postcondition" they are required to have performed only well-typed writes and allocations and, importantly, only in such regions as permitted, respectively, by the write and allocation effects.

The masking rule, however, introduces a new dimension to the development of worlds: in addition to adding new locations with types to existing regions as sketched above, we may now introduce new regions as well as *mask out* existing ones. One can loosely think of masking out region ρ as deallocating the locations of that region. A slightly revised intuition

[5] In general, worlds may contain complex invariants of the heap; they may even vary over time and may thus be represented by state transition systems [20]. Here, we only need the invariant that values stored at locations belong to certain types.

is that we just stop caring about region ρ: the environment has no way of telling the memory region is there, and so we may safely relinquish control. On the level of worlds, we discard the region, i.e., lose the locations and the associated types. Whether the locations are actually deallocated or just float around in the surroundings does not matter: from our point of view they are gone. Thus, semantically, region masking is related to region deallocation. Ahmed developed a step-indexeded model of region deallocation in her thesis [16], which turns out to be closely related to our treatment of region masking. We now explain our approach, which was developed independently of Ahmed's, and then detail the relationship to Ahmed's approach.

As argued by example in the Introduction, region masking leaves us with the issue of dangling pointers. Phrased more concretely: how should the interpretation of types cope with region masking? What is, say, the interpretation of $\mathsf{ref}_\rho\tau$ in a world where region ρ has been masked out? A natural choice is the empty set — after all, we know nothing about region ρ so what locations could we possibly choose. We, however, take a different approach: We generally interpret a type as all values with the properties you expect from that type. The more properties, the fewer values and vice versa. A value v in $\mathsf{ref}_\rho\tau$ is a reference to a location that we used to — but no longer — control. The value at the location may have changed or it may even have been deallocated: v is conceptually *dangling*. What can we reasonably do with a dangling pointer? We can neither read nor write it, indeed, we can do only things that go for all values, e.g., put it into a pair, project it out, etc. In other words, we expect no properties of dangling pointers and correspondingly interpret $\mathsf{ref}_\rho\tau$ as the set of all values. We take a similar stand on functions: interpreting a function type that has effects on masked out regions gives the set of all values; such functions are somehow dangling too. Running such a function relies on preconditions outside our control, so we make no promises about the behavior.

This approach goes well with type monotonicity: the interpretation of types should not shrink under masking and, indeed, masking out region ρ enlarges $\mathsf{ref}_\rho\tau$ to hold all values. There is a catch, though, since future worlds may introduce new regions as well, and reintroducing region ρ would not do. This is obvious from type monotonicity; a more conceptual explanation, however, is this: assume that some function reads region ρ and that region ρ holds location l that stores values of some type. To run the function, we need to establish the precondition that a value of the proper type is stored at location l, because the function could very well read l. We then mask out region ρ, losing all information about l in the process, and afterwards reintroduce region ρ. To run the function now, we verify that all locations in region ρ hold values of appropriate type — this is, after all, the precondition of the function — but as l is no longer in ρ, we cannot expect proper behavior of the function.

We solve this issue by tracking, in the worlds, the regions that have been masked out and prohibit their reintroduction. One viewpoint is that, as the world develops over time, a region goes through a life-cycle: initially, it is unknown to the world, but at some point it gets initialized, joining the set of *live* regions. With further development of the world, locations with associated types are added to the region. This proceeds until the region is masked out; it loses all content and is moved to the set of *dead* regions. And this is a one-way street: once you are gone, you can't come back.

In Ahmed's unary model of a region calculus with region deallocation [16], she uses the same idea of keeping track of dead regions. One technical difference is that in Ahmed's approach, the dead regions still contain the old locations and their types, whereas here we only retain the region name. This difference seems to be unimportant. In Ahmed's case, regions are part of the operational semantics, whereas in our case, they are a purely logical construct used for reasoning about the standard operational semantics, following Benton et al.'s earlier work [14,15].

Local reasoning Conceptually, a world does not describe the entire heap, just the part of it that the program sees or controls. On the level of definitions, this comes down to a frame property of our computations: a computation in a world runs in a world-adhering heap extended with a frame, and the latter remains unchanged. In addition, a computation may allocate locations that are not tracked by the ensuing world; these locations are conceptually transferred to the surroundings, they become part of the frame of the following computation. This is the fate of locations in regions that are masked out. This way, we achieve a form of local reasoning by quantifying over frames, similarly to models of separation logic for higher-order languages [21].

4. Metric spaces and the type-world circularity

As argued above, we intend to augment our semantics with worlds that track the layout of the heap; hence we are faced with the construction of such worlds. Defining them directly will not do since, loosely, a world holds semantic types whilst semantic types are parameterized over worlds; this is the *type-world circularity* observed already by Ahmed in her thesis [16].

In recent work, Reus, Schwinghammer, Støvring, Yang and two of the authors [17] have proposed a general solution to such circularities, applying metric-space theory. The notion of worlds we require here is sufficiently simple that this solution is applicable off-the-shelf, so to speak. So we omit the machinery of the construction from the main text and only present the resulting definition of worlds, since this is our main object of interest. Details of the construction are given in Appendix A. In addition, we will largely ignore the fact that we actually deal in metric spaces and not just plain sets. We emphasize that this is just for presentation purposes, worlds and types *are* metric spaces with certain properties; this is necessary to solve the circularity and must be taken into account, e.g., when interpreting types.

$$
\begin{aligned}
\mathrm{ParBij}(X, Y, \mathbf{Z}) = \; & \{P \subseteq_{fin} X \times Y \times \mathbf{Z} \mid \forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in P. \\
& (x_1 = x_2 \Rightarrow y_1 = y_2 \wedge z_1 = z_2) \\
& \wedge (y_1 = y_2 \Rightarrow x_1 = x_2 \wedge z_1 = z_2)\} \\
\mathbf{W} = \; & \{(\varphi, \psi) \in (\mathcal{RN} \rightharpoonup_{fin} \mathrm{ParBij}(\mathcal{L}, \mathcal{L}, \widehat{\mathbf{T}})) \times \mathcal{P}_{fin}(\mathcal{RN}) \mid \\
& \mathrm{dom}(\varphi) \cap \psi = \emptyset \\
& \wedge \forall r, s \in \mathrm{dom}(\varphi). r \neq s \Rightarrow \mathrm{dom}_1(\varphi(r)) \cap \mathrm{dom}_1(\varphi(s)) = \emptyset \\
& \wedge \mathrm{dom}_2(\varphi(r)) \cap \mathrm{dom}_2(\varphi(s)) = \emptyset\} \\
w \sqsubseteq w' \iff \; & \exists n \in \mathbb{N}. \exists w_0, w_1, \ldots, w_n \in \mathbf{W}. w = w_0 \wedge w' = w_n \wedge \\
& \forall i \in \{0, \ldots, n-1\}. (\exists r \in \mathcal{RN}. w_i \rightarrow_{reg(r)} w_{i+1}) \\
& \vee (\exists r \in \mathcal{RN}. \exists l_1, l_2 \in \mathcal{L}. \exists \mu \in \widehat{\mathbf{T}}. w_i \rightarrow_{al(r,l_1,l_2,\mu)} w_{i+1}) \\
& \vee (\exists r \in \mathcal{RN}. w_i \rightarrow_{mask(r)} w_{i+1}) \\
\mathbf{T} = \; & \mathbf{W} \rightarrow_{mon} \mathit{URel}(\mathcal{V}), \quad \iota : \widehat{\mathbf{T}} \cong \frac{1}{2}\mathbf{T}
\end{aligned}
$$

Fig. 3. Our semantic footing: worlds and types. The former comes equipped with the preorder that is the reflexive, transitive closure of the transitions in Fig. 4.

$$
\begin{aligned}
(\varphi, \psi) \rightarrow_{al(r,l_1,l_2,\mu)} (\varphi', \psi) \iff \; & r \in \mathrm{dom}(\varphi) \wedge l_1 \notin \mathrm{dom}_1(\varphi(r)) \\
& \wedge l_2 \notin \mathrm{dom}_2(\varphi(r)) \wedge \mathrm{dom}(\varphi') = \mathrm{dom}(\varphi) \\
& \wedge \varphi'(r) = \varphi(r) \cup \{(l_1, l_2, \mu)\} \\
& \wedge \forall s \in \mathrm{dom}(\varphi) \setminus \{r\}. \varphi'(s) = \varphi(s). \\
(\varphi, \psi) \rightarrow_{reg(r)} (\varphi', \psi) \iff \; & r \in \mathcal{RN} \setminus (\mathrm{dom}(\varphi) \cup \psi) \\
& \wedge \mathrm{dom}(\varphi') = \mathrm{dom}(\varphi) \cup \{r\} \\
& \wedge \varphi'(r) = \emptyset \wedge \forall s \in \mathrm{dom}(\varphi). \varphi'(s) = \varphi(s). \\
(\varphi, \psi) \rightarrow_{mask(r)} (\varphi', \psi') \iff \; & r \in \mathrm{dom}(\varphi) \wedge \mathrm{dom}(\varphi') = \mathrm{dom}(\varphi) \setminus \{r\} \\
& \wedge \psi' = \psi \cup \{r\} \\
& \wedge \forall s \in \mathrm{dom}(\varphi) \setminus \{r\}. \varphi'(s) = \varphi(s).
\end{aligned}
$$

Fig. 4. The three worlds transitions. We have $r \in \mathcal{RN}$ in all of them and additionally $l_1, l_2 \in \mathcal{L}$ and $\mu \in \widehat{\mathbf{T}}$ in the first. Transitions are, once parameters are given, partial maps from $\mathbf{W}$ to $\mathbf{W}$.

We face an additional challenge here: as described above, we intend to track dead regions to avoid recycling them. But if an expression is well-typed by the masking rule, then in terms of the development of worlds, it initializes and eventually masks out the very same region on each evaluation, and we may want to run it more than once. Our solution to this is to introduce a layer of indirection, following the ideas of earlier work [4,6]: on the level of syntactic types, we have the region variables $\mathcal{RV}$ introduced above. In the worlds, however, we work with a countably infinite set of *region names* $\mathcal{RN}$. To interpret types we have *region environments*, i.e., maps $\mathcal{RV} \rightharpoonup_{fin} \mathcal{RN}$ with adequate domains; thus, on each evaluation of an expression that performs masking, we can map the same region variable to a fresh region name. In the textual explanations below, however, we purposely blur the distinction between region variables, region names and the regions themselves.

The definition of worlds and types is given in Fig. 3; the ordering on worlds relies on the world transitions given in Fig. 4. Both warrant a few comments. $\mathrm{ParBij}(X, Y, \mathbf{Z})$ are finite partial bijections between X and Y decorated with elements from $\mathbf{Z}$; we write $\mathrm{dom}_1(P)$ for the set of first coordinates and $\mathrm{dom}_2(P)$ for the set of second coordinates. Worlds $\mathbf{W}$ have two components: the first is the *live* regions, these are partial bijections; the second the *dead* regions. No region can be both live and dead, nor can any location belong to more than one region.

Worlds may develop over time according to the transitions in Fig. 4. The first transition adds a location pair (l_1, l_2) with associated type (μ) to a live region (r); this corresponds to an actual allocation in the operational semantics and is a standard notion of world extension. The second and third transitions are orthogonal to the first. They give the region dynamics and have no counterpart in the operational semantics; they are, however, intimately connected to the masking rule. The first initializes a new empty region, the second masks out a live one, losing the partial bijection in the process. After being masked out, a region is considered dead and cannot be initialized once more.

Since worlds consist of partial bijections on locations and allocation of new locations takes place in "lock-step" on both sides, one might be concerned about whether it is possible to relate computations that use different numbers of locations.

$$[\![1]\!]^R\, w = \{(k, (), ()) \mid k \in \mathbb{N}\}$$

$$[\![\mathsf{int}]\!]^R\, w = \{(k, n, n) \mid k \in \mathbb{N} \wedge n \in \mathbb{Z}\}$$

$$[\![\tau_1 \times \tau_2]\!]^R\, w = \left\{ (k, \langle v_1^1, v_1^2\rangle, \langle v_2^1, v_2^2\rangle) \mid \begin{array}{l} (k, v_1^1, v_2^1) \in [\![\tau_1]\!]^R\, w \\ \wedge\ (k, v_1^2, v_2^2) \in [\![\tau_2]\!]^R\, w \end{array} \right\}$$

$$[\![\mathsf{ref}_\rho\, \tau]\!]^R\, w = \begin{cases} \left\{ (k, l_1, l_2) \mid \begin{array}{l} \exists \mu \in \widehat{\mathbf{T}}.\, (l_1, l_2, \mu) \in w(R(\rho)) \wedge \\ \forall w' \sqsupseteq w.\ [\![\tau]\!]^R\, w' \overset{k}{=} (\iota\, \mu)(w') \end{array} \right\} \\ \quad \text{when } R(\rho) \in \mathrm{dom}(w) \\ \left\{ (k, v_1, v_2) \mid k \in \mathbb{N} \wedge v_1, v_2 \in \mathcal{V} \right\} \text{ when } R(\rho) \notin \mathrm{dom}(w) \end{cases}$$

$$[\![\tau_1 \to^\varepsilon \tau_2]\!]^R\, w = \left\{ (k, \mathsf{fix}\, f(x).e_1, \mathsf{fix}\, f(x).e_2) \mid \begin{array}{l} \forall j \le k.\, \forall w' \sqsupseteq w. \\ R : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w') \Rightarrow \\ \forall v_1, v_2 \in \mathcal{V}.(j, v_1, v_2) \in [\![\tau_1]\!]^R\, w' \Rightarrow \\ (j, (\mathsf{fix}\, f(x).e_1)\, v_1, (\mathsf{fix}\, f(x).e_2)\, v_2) \\ \in \mathcal{E}\, [\![\tau_2^\varepsilon]\!]^R\, w' \end{array} \right\}$$

$$[\![\forall \rho \notin \Pi.\, \tau^\varepsilon]\!]^R\, w = \left\{ (k, \mathsf{susp}\, e_1, \mathsf{susp}\, e_2) \mid \begin{array}{l} \forall j \le k.\, \forall w' \sqsupseteq w.\, \forall r \in |w'|. \\ R[\rho \mapsto r] : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w') \Rightarrow \\ (j, e_1, e_2) \in \mathcal{E}\, [\![\tau^\varepsilon]\!]^{R[\rho \mapsto r]}\, w' \end{array} \right\}$$

Fig. 5. Interpretation of types. For $[\![\tau]\!]^R$ to be defined we require $\mathrm{wf}(\tau)$ and $R : \mathcal{RV} \rightharpoonup_{fin} \mathcal{RN}$ with $\mathrm{FRV}(\tau) \subseteq \mathrm{dom}(R)$. We have $w \in \mathbf{W}$ and assume $R(\mathrm{FRV}(\tau)) \subseteq |w|$; otherwise we define $[\![\tau]\!]^R\, w$ to be the empty set.

That is indeed possible, because the relation on computations allows computations to allocate values outside of the part of the heap tracked by the world, see Fig. 6 (in particular, f_1' and f_2' in the computation relation).

The reflexive, transitive closure of the transitions is a preorder on worlds. We require of our types $\mathbf{T}$ that they are monotone with respect to that preorder and the standard set-theoretic inclusion on $URel(\mathcal{V})$; this is the type monotonicity (hence the subscript *mon* on the function space in the definition of $\mathbf{T}$). For any set X, $URel(X)$ is the set of indexed, downwards closed relations on X, i.e.,

$$URel(X) = \{R \subseteq \mathbb{N} \times X \times X \mid \forall (k, x_1, x_2) \in R.\, \forall j < k.\, (j, x_1, x_2) \in R\}.$$

The downwards closure is essential, it prevents values from fleeing types as the operational semantics progresses.[6] One minor issue remains: the types that decorate the partial bijections that are the regions belong to $\widehat{\mathbf{T}}$ and must be coerced into $\mathbf{T}$ by the isomorphism ι before they can be applied to a world. This isomorphism is what we get from the solution to the type-world circularity (see Appendix A).

A bit of world-related notation will come in handy: for a world $w = (\varphi, \psi)$ we set $\mathrm{dom}(w) = \mathrm{dom}(\varphi)$ and $|w| = \mathrm{dom}(\varphi) \cup \psi$, i.e., $\mathrm{dom}(w)$ is the set of live regions and $|w|$ is the combined set of live and dead regions. We write $\mathrm{dom}_1(w)$ for the union of all first coordinates of all live regions; $\mathrm{dom}_2(w)$ is defined similarly. For $r \in \mathrm{dom}(w)$ we abuse notation and write $w(r)$ for $\varphi(r)$.

Finally, let us remark that our worlds are simple in the sense that the only invariants we allow are those described by semantic types. That is similar to the approach taken to model references in [22]. It suffices for showing the soundness of effect-based program transformations, which only depend on types and effects rather than on the syntax of the program. One could also consider richer notions of invariants, as in the models of references by Ahmed et al. [18] and by Dreyer et al. [20], but that would complicate the model and is orthogonal to the main question of this paper.

5. Types and the logical relation

The relational, world-indexed interpretation of types is given in Fig. 5; it relies on the interpretation of computations given in Fig. 6. It is worthwhile to comment a bit on this. Note that both the interpretation of types and computations take a number of parameters; the requirements on those are given in the captions of the figures. Here we just emphasize that we interpret well-formed types only. Note also that the definitions given here are asymmetric. We use them to define an *approximation* relation, and take its symmetric closure as the logical equivalence relation, see Fig. 7 for the details.

Overall, we follow the intuition laid out in Section 3. First, whenever we interpret a type τ in a world w where $R(\mathrm{FRV}(\tau))$ is outside the support, $|w|$, we get the empty set. This we never do; it is just a dummy clause that is neutral with respect to type monotonicity, but we need it since, for technical convenience, we want interpreted types to be applicable to all worlds. Integer, unit and product types are standard. Looking at the reference type there are two cases: The first case is the proper one, here $R(\rho)$ is a live region and we go through it in search for location pairs that hold values of a type semantically identical to τ. The quantification over future worlds ensure type monotonicity and the k-equality is necessary for the step-indexed setup; both are quite standard. The latter means that the sets we compare are equal if we

[6] The set $URel(X)$ has a natural metric [17] and the functions in $\mathbf{T}$ are not only monotone but also non-expansive; see Appendix A for details.

$$\mathcal{E}[\![\tau^{\varepsilon}]\!]^R\, w = \Big\{(k, e_1, e_2) \mid \forall j \le k.\, \forall e_1' \in \mathcal{E}.\, \forall h_1, h_2, f_1, f_2, h_1^\dagger \in \mathcal{H}.$$
$$\big[(k, h_1, h_2) \in \mathbf{P}_\varepsilon^R\, w \wedge \langle e_1 \mid h_1 \cdot f_1\rangle \xrightarrow{j} \langle e_1' \mid h_1^\dagger\rangle \wedge \mathrm{irr}\langle e_1' \mid h_1^\dagger\rangle\big]$$
$$\Rightarrow \exists w' \sqsupseteq w.\, \exists e_2' \in \mathcal{E}.\, \exists h_1', h_2', f_1', f_2' \in \mathcal{H}.\, \big[\mathrm{dom}(w) = \mathrm{dom}(w') \wedge h_1^\dagger = h_1' \cdot f_1' \cdot f_1$$
$$\wedge \langle e_2 \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle e_2' \mid h_2' \cdot f_2' \cdot f_2\rangle \wedge (k-j, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R\, w, w'$$
$$\wedge\, (k-j, e_1', e_2') \in [\![\tau]\!]^R\, w'\big]\Big\}$$

$$(k, h_1, h_2) \in \mathbf{P}_\varepsilon^R\, w \iff$$
$$\mathrm{dom}(h_i) = \mathrm{dom}_{\mathrm{i}}(w)$$
$$\wedge \big(\forall \rho \in \mathrm{rds}\,\varepsilon.\, \forall (l_1, l_2, \mu) \in w(R(\rho)).\, k > 0 \Rightarrow (k-1, h_1(l_1), h_2(l_2)) \in (\iota\,\mu)(w)\big)$$

$$(k, h_{11}, h_{21}, h_{12}, h_{22}) \in \mathbf{Q}_\varepsilon^R\, w_1, w_2 \iff$$
$$\mathrm{dom}(h_{ij}) = \mathrm{dom}_{\mathrm{i}}(w_j) \wedge \big(\forall l_1 \in \mathrm{dom}(h_{11}).\, h_{11}(l_1) \neq h_{12}(l_1) \Rightarrow \exists \rho \in \mathrm{wrs}\,\varepsilon.$$
$$\exists (m_1, l_2, \mu) \in w_1(R(\rho)).\, l_1 = m_1 \wedge k > 0 \Rightarrow (k-1, h_{12}(l_1), h_{22}(l_2)) \in (\iota\,\mu)(w_2)\big)$$
$$\wedge \big(\forall l_2 \in \mathrm{dom}(h_{21}).\, h_{21}(l_2) \neq h_{22}(l_2) \Rightarrow \exists \rho \in \mathrm{wrs}\,\varepsilon.\, \exists (l_1, m_2, \mu) \in w_1(R(\rho)).\, l_2 = m_2 \wedge$$
$$k > 0 \Rightarrow (k-1, h_{12}(l_1), h_{22}(l_2)) \in (\iota\,\mu)(w_2)\big)$$
$$\wedge \big(\forall r \in \mathrm{dom}(w_1).\, \forall (l_1, l_2, \mu) \in w_2(r) \setminus w_1(r).\, r \in R(\mathrm{als}\,\varepsilon) \wedge$$
$$k > 0 \Rightarrow (k-1, h_{12}(l_1), h_{22}(l_2)) \in (\iota\,\mu)(w_2)\big)$$

Fig. 6. Interpretation of computations with pre- and postconditions. There are implicit disjointness requirements on heaps: the heap compositions must all be well-defined. In all three cases we have $R : \mathcal{RV} \rightharpoonup_{fin} \mathcal{RN}$ and $w \in \mathbf{W}$. For $\mathcal{E}[\![\varepsilon^\tau]\!]^R\, w$ we require $\mathrm{wf}(\tau)$, $\mathrm{FRV}(\varepsilon, \tau) \subseteq \mathrm{dom}(R)$ and $R(\mathrm{FRV}(\varepsilon)) \subseteq \mathrm{dom}(w)$. $\mathbf{P}_\varepsilon^R\, w$ is defined for $\mathrm{FRV}(\varepsilon) \subseteq \mathrm{dom}(R)$ and $R(\mathrm{FRV}(\varepsilon)) \subseteq \mathrm{dom}(w)$, $\mathbf{Q}_\varepsilon^R\, w_1, w_2$ additionally requires $w_2 \in \mathbf{W}$ and $w_1 \sqsubseteq w_2$ with $\mathrm{dom}(w_1) = \mathrm{dom}(w_2)$.

restrict to elements with index strictly less than k. In the last case we look for references to a masked region; these are conceptually dangling pointers that we make no assumption about, hence we return all (pairs of) values.

Interpreting functions and region polymorphism proceeds along similar lines. To obtain type monotonicity we quantify over future worlds, but we make promises only for proper worlds, i.e., worlds where all effect-relevant regions are live. We could, similarly, have considered only future worlds with $R(\rho)$ intact for the reference types, but for simplicity we leave it out.

The interpretation of computations is crucial. Note, first, that $\mathrm{rds}\,\varepsilon$ are all region variables with read effects in ε; $\mathrm{wrs}\,\varepsilon$ and $\mathrm{als}\,\varepsilon$ are defined similarly. $\mathbf{P}_\varepsilon^R\, w$ denotes the precondition on heaps that computations running in world w with effects ε should satisfy: that all location pairs in all regions with read effects do, in fact, hold related values of the appropriate type. If the precondition holds and the left hand side terminates, then so does the right hand side, and there is a future world w' such that the results are related at the desired type in w', and the resulting heaps satisfy the postcondition $\mathbf{Q}_\varepsilon^R\, w, w'$. Note that not all locations allocated by the computations need to be in $dom_i w'$: f_i' contain precisely these locations; they contain, for instance, locations masked out throughout the computations. Also, w' is picked only when the computations finish. Thus, until that point the newly allocated locations that *should* be tracked by world do not have to be in any way related: we only need to exhibit that when the computations finish, newly allocated parts of the world are indeed related. This is enforced by the postcondition relation that we mentioned earlier, $\mathbf{Q}_\varepsilon^R\, w, w'$. It states that any writes to existing locations are of the correct type and are permitted by a write effect; also any newly allocated locations tracked by w' hold well-typed, related values and are in regions with an allocation effect. It is not unusual that the postcondition speaks of the initial heaps as well as the final ones: this is also the case in, for example, Hoare Type Theory [23]. Since we do local reasoning, there may be parts of the heap outside our control; these are the frames f_1 and f_2, which must remain unmodified.

Finally, it is worthwhile to remark that the region-dynamics of computations is quite restricted: the future world w' must have the same live regions as w. In other words, computations cannot mask out existing regions and if they initialize any new regions, they are obliged to mask them out before they terminate. Here we take inspiration from work on region-based memory management [4,6] where regions are allocated and deallocated following a stack discipline.

We prove that the interpretation is well-defined in a series of lemmas and propositions; we assume throughout that all parameters satisfy the requirements of the operators, see the figures for details.

Lemma 4. *For any $n \in \mathbb{N}$ and $w_1, w_2 \in W$ with $w_1 \overset{n}{=} w_2$ we have $\mathbf{P}_\varepsilon^R\, w_1 \overset{n}{=} \mathbf{P}_\varepsilon^R\, w_2$, with the metric of $URel(\mathcal{H})$.*

$$(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w \iff \forall x \in \mathrm{dom}(\Gamma).\, (k, \gamma_1(x), \gamma_2(x)) \in [\![\Gamma(x)]\!]^R\, w$$

$$\Pi \mid \Gamma \models e_1 \preceq e_2 : \tau, \varepsilon \iff$$
$$\forall k \in \mathbb{N}. \forall w \in \mathbf{W}. \forall R : \Pi \to |w|. \forall \gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma).$$
$$\left[R : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w) \wedge (k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w\right] \Rightarrow (k, \gamma_1(e_1), \gamma_2(e_2)) \in \mathcal{E}\, [\![\tau^\varepsilon]\!]^R\, w$$

$$\Pi \mid \Gamma \models e_1 \simeq e_2 : \tau, \varepsilon \iff \Pi \mid \Gamma \models e_1 \preceq e_2 : \tau, \varepsilon \wedge \Pi \mid \Gamma \models e_2 \preceq e_1 : \tau, \varepsilon$$

Fig. 7. Definition of the logical approximation and equivalence. We require all types to be well-formed, $\mathrm{FRV}(\Gamma, \tau, \varepsilon) \subseteq \Pi$ and, as always, $\mathrm{FV}(e_1, e_2) \in \Gamma$. $\mathrm{Subst}(\Gamma)$ denotes the set of substitutions mapping the variables of Γ to closed values. R has to be an injection from the free region variables of ε to the live regions of w.

Lemma 5. *For any $n \in \mathbb{N}$ and $w_1, w_1', w_2, w_2' \in W$ with $w_1 \overset{n}{=} w_2$, $w_1' \overset{n}{=} w_2'$, $w_1 \sqsubseteq w_1'$ and $w_2 \sqsubseteq w_2'$ we have $\mathbf{Q}_\varepsilon^R\, w_1, w_1' \overset{n}{=} \mathbf{Q}_\varepsilon^R\, w_2, w_2'$, with the metric of $\mathit{URel}(\mathcal{H})$.*

Remember that the postcondition operator has nontrivial side-conditions: for $\mathbf{Q}_\varepsilon^R\, w, w'$ to be defined we require $w \sqsubseteq w'$ and $\mathrm{dom}(w) = \mathrm{dom}(w')$ as well as the standard requirement, i.e., that R be injective from $\mathrm{FRV}(\varepsilon)$ to $\mathrm{dom}(w)$.

Proposition 6. *We have $[\![\tau]\!]^R$ is a well-defined type, i.e., it is a non-expansive and monotone map from $\mathbf{W}$ to $\mathit{URel}(\mathcal{V})$.*

Lemma 7. *For $w_1, w_2 \in \mathbf{W}$ we have that $w_1 \overset{n}{=} w_2$ implies $\mathcal{E}\, [\![\varepsilon^\tau]\!]^R\, w_1 \overset{n}{=} \mathcal{E}\, [\![\varepsilon^\tau]\!]^R\, w_2$ for any $n \in \mathbb{N}$.*

These two are proved by simultaneous induction. Notice that we make no monotonicity requirement on the interpretation of computations, indeed, we do not even claim that it maps $\mathbf{W}$ to $\mathit{URel}(\mathcal{E})$. As such, the use of n-equality is abuse of notation since we may have strayed outside our metric spaces; the meaning is the usual, though: the two sets agree if we restrict to elements with indices strictly less than n.

When interpreting a type, we use region environments with possibly excessive domains. Clearly, the value of the region environments outside the region variables of the type should not matter; this is captured in the following lemmas, crucial for proving compatibility; the proof goes by mutual induction:

Lemma 8 *(Environment extension). For any $R : \mathcal{RV} \rightharpoonup_{fin} \mathcal{RN}$ and τ such that $\mathrm{FRV}(\tau) \subseteq \mathrm{dom}(R)$ we have $[\![\tau]\!]^R = [\![\tau]\!]^{R|_{\mathrm{FRV}(\tau)}}$.*

Lemma 9. *For any $R : \mathcal{RV} \rightharpoonup_{fin} \mathcal{RN}$, τ and ε, such that $\mathrm{FRV}(\tau, \varepsilon) \subseteq \mathrm{dom}(R)$, we have $\mathcal{E}\, [\![\tau^\varepsilon]\!]^R = \mathcal{E}\, [\![\tau^\varepsilon]\!]^{R|_{\mathrm{FRV}(\tau,\varepsilon)}}$.*

The logical relation on expressions is defined in Fig. 7. Π is a syntactic over-approximation of all region variables; together with the condition $R : \Pi \to |w|$ it ensures that we deal in live and dead regions only, not unknown ones. As for functions and computations, we require that the regions of the effects are live.

The logical relation is asymmetrical: the left hand side *approximates* the right hand side. We write $\Pi \mid \Gamma \models e_1 \simeq e_2 : \tau, \varepsilon$ if the approximation goes both ways and consider the computations *equivalent* in that case. Our logical relation is sound in the following, standard, sense, with subtyping interpreted through set inclusion:

Theorem 10 *(Compatibility). The logical relation in Fig. 7 is compatible with the typing rules of Fig. 1. That is, the formation of expressions according to the typing rules respects the logical relation.*

Proposition 11. *For any region environment $R : \Pi \to \mathcal{RN}$ and any world $w \in \mathbf{W}$, if $\Pi \vdash \tau_1 \leq \tau_2$ then $[\![\tau_1]\!]^R\, w \subseteq [\![\tau_2]\!]^R\, w$.*

The proof of the theorem relies naturally on the proposition. The proofs are deferred to the next section.

We have the Fundamental Lemma as corollary:

Lemma 12. *(Fundamental)*

$$\Pi \mid \Gamma \vdash e : \tau, \varepsilon \Longrightarrow \Pi \mid \Gamma \models e \preceq e : \tau, \varepsilon.$$

Definition 1 *(Contextual equivalence).* Two well-typed computations are *contextually equivalent* if for any closing, integer contexts, they co-terminate with the same value, when run in any two heaps. Formally:

$$\Pi \mid \Gamma \vdash e_1 \equiv e_2 : \tau,\ \varepsilon \iff \Pi \mid \Gamma \vdash e_i : \tau,\ \varepsilon \wedge$$
$$\forall C \in \mathcal{C}.\ \forall h_1, h_2 \in \mathcal{H}.\ \forall n \in \mathbb{N}.\ \cdot \mid \cdot \vdash C[e_i] : \mathsf{int},\ \emptyset \wedge$$
$$(\langle C[e_1] \mid h_1 \rangle \xrightarrow{*} \langle n \mid \_ \rangle \iff \langle C[e_2] \mid h_2 \rangle \xrightarrow{*} \langle n \mid \_ \rangle)$$

Also as a corollary of compatibility, we get the following theorem:

Theorem 13. *The logical relation defined in Fig. 7 is sound with respect to contextual equivalence, i.e.,*

$$\Pi \mid \Gamma \models e_1 \simeq e_2 : \tau, \varepsilon \Longrightarrow \Pi \mid \Gamma \vdash e_1 \equiv e_2 : \tau,\ \varepsilon.$$

5.1. On interpretation of the write effects

Before we proceed to the soundness proof, we pause to discuss briefly the definitions of effect interpretations in Fig. 6. Particularly interesting is the fact that we require the values stored *outside* the write effects to be the same between the initial and final heaps. Since our relation is built over the operational semantics, the equality used there is syntactic, which could be considered unsatisfactory. Indeed, since we are building a relational model, one could think that we should use the semantic type of this location, and only require the values to be related *at that semantic type*. This, however, would lead to significant problems.

Conceptually, the first hurdle is that the type stored in the world is supposed to relate values of the first, "left-hand-side", computation to values of the "right-hand-side" one. Hence, in the presence of open types, where the types of the left- and right-values need not agree, one cannot simply use the relation itself, because we want to express an equivalence among two values on the *same* side. Moreover, this approach would not scale to the unary case, where the relation being defined is not even binary. These problems suggest that one would at the least need to generalize the construction of the worlds.

A second, more pressing issue that comes up in our setup is that we need to sequence the interpretation of the effects in a computation: for instance in a let-binding rule. Currently, this ability is expressed by Lemma 17 in the following. Thus, any relation used to relate the initial and final states of the values outside the write effects of a computation would need to be transitive. However, transitivity is a property that is normally lost in the step-indexed models: we only regain it at the contextual equivalence level. Thus, even disregarding the conceptual issues, we simply cannot use the semantic type directly, to allow more leeway in the behavior of values that are outside the write effects.

One question remains: how does this possible deficiency compare with other work, in particular domain-theoretic models, e.g., [14]. In the domain-theoretic models, the relational interpretation of types is constructed over a universal domain, which is used to give an interpretation of the untyped language. We emphasize that such models also require a *base* equality as a building block of the relational interpretation (the equality on the universal domain). In particular the values in the heap outside the write effects are related by this base equality. Of course, in the denotational models the base equality is semantic and so less fine-grained than our syntactic equality, which can be considered better, but the core problem still persists there.

6. Soundness

In this section, we present most of the cases of the proof of the Compatibility Theorem stated just above. We begin with lemmas about pre- and postconditions.

Lemma 14 *(Precondition strengthening). The effect annotation of the precondition relation can be strengthened, i.e.,* $\mathbf{P}^R_{\varepsilon\cup\varepsilon'}\, w \subseteq \mathbf{P}^R_{\varepsilon}\, w$.

Proof. Take arbitrary $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon\cup\varepsilon'}\, w$. In the definition of $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon}\, w$, the equalities $\mathrm{dom}(h_i) = \mathrm{dom_i}(w)$ are trivially enforced by the hypothesis. The property

$$\forall \rho \in \mathrm{rds}\,\varepsilon.\ \forall (l_1, l_2, \mu) \in w(R(\rho)).\ k > 0 \Rightarrow (k-1, h_1(l_1), h_2(l_2)) \in (\iota\,\mu)(w)$$

also follows from the hypothesis $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon\cup\varepsilon'}\, w$ since $\mathrm{rds}\,\varepsilon \subseteq \mathrm{rds}\,(\varepsilon \cup \varepsilon')$. □

Lemma 15 *(Postcondition weakening). The effect annotation of the postcondition relation can be weakened, respecting the domains of the region map and the world, i.e., if* $\mathrm{FRV}(\varepsilon') \subseteq \mathrm{dom}(R)$ *and* $R(\mathrm{FRV}(\varepsilon')) \subseteq \mathrm{dom}(w)$*, then* $\mathbf{Q}^R_{\varepsilon}\, w, w' \subseteq \mathbf{Q}^R_{\varepsilon\cup\varepsilon'}\, w, w'$.

Proof. In the definition of $\mathbf{Q}^R_{\varepsilon\cup\varepsilon'}\, w, w'$ only existential quantifications on $\mathrm{wrs}\,(\varepsilon \cup \varepsilon')$ and $\mathrm{als}\,(\varepsilon \cup \varepsilon')$ are used, and so the inclusions $\mathrm{wrs}\,\varepsilon \subseteq \mathrm{wrs}\,(\varepsilon \cup \varepsilon')$ and $\mathrm{als}\,\varepsilon \subseteq \mathrm{als}\,(\varepsilon \cup \varepsilon')$ suffice for the proof. □

Lemma 16 *(Precondition composition). The postcondition relation maps pairs of heaps that are related by precondition at the initial world to pairs of heaps that are related by precondition at the terminal world, i.e., if we have* $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon}\, w$ *and* $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}^R_{\varepsilon'}\, w, w'$*, then* $(k, h_1', h_2') \in \mathbf{P}^R_{\varepsilon}\, w'$ *holds too.*

Proof. Unwinding the definition of $(k, h_1', h_2') \in \mathbf{P}_\varepsilon^R\, w'$, the equalities $\mathrm{dom}(h_i') = \mathrm{dom_i}(w')$ come directly from $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_{\varepsilon'}^R\, w, w'$. To prove

$$\forall \rho \in \mathrm{rds}\,\varepsilon.\, \forall (l_1, l_2, \mu) \in w'(R(\rho)).\, k > 0 \Rightarrow (k-1, h_1'(l_1), h_2'(l_2)) \in (\iota\,\mu)(w')$$

there are multiple possibilities:

- If $l_1 \notin \mathrm{dom}(h_1)$ then $(l_1, l_2, \mu) \in w'(R(\rho)) \setminus w(R(\rho))$ and so from $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_{\varepsilon'}^R\, w, w'$ we get that $\rho \in \mathrm{als}\,\varepsilon'$ and $k > 0 \Rightarrow (k-1, h_1'(l_1), h_2'(l_2)) \in (\iota\,\mu)(w')$. Similarly if $l_2 \notin \mathrm{dom}(h_2)$.
- If $h_1(l_1) = h_1'(l_1)$ and $h_2(l_2) = h_2'(l_2)$ it comes down to $(k, h_1, h_2) \in \mathbf{P}_\varepsilon^R\, w$ and type monotonicity.
- If $h_1(l_1) \neq h_1'(l_1)$ then $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_{\varepsilon'}^R\, w, w'$ enforces that there is $\rho' \in \mathrm{wrs}\,\varepsilon'$ such that

$$\exists (m_1, l_2', \mu) \in w(R(\rho')).\, l_1 = m_1 \wedge k > 0 \Rightarrow (k-1, h_1'(l_1), h_2'(l_2')) \in (\iota\,\mu)(w')$$

By the construction of worlds, $R(\rho') = R(\rho)$ since both contain l_1. We proceed similarly if $h_2(l_2) \neq h_2'(l_2)$. □

It is worth emphasizing that we require no relationship between ε and ε'; we simply rely on the fact that the postcondition ensures well-typed updates and allocations, no matter the region.

Lemma 17 *(Postcondition composition). If we have* $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R\, w, w'$ *and* $(k, h_1', h_2', h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w', w''$*, then* $(k, h_1, h_2, h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w, w''$ *holds.*

Proof. To prove $(k, h_1, h_2, h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w, w''$, we have to show a range of things. Notice first that $\mathrm{dom}(h_i) = \mathrm{dom_i}(w)$ and $\mathrm{dom}(h_i'') = \mathrm{dom_i}(w'')$, which come respectively from $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R\, w, w'$ and $(k, h_1', h_2', h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w', w''$.

We now look into changes to the initial heap h_1; changes to h_2 are handled similarly. Take any $l_1 \in \mathrm{dom}(h_1)$ and let us consider the case $h_1'(l_1) \neq h_1''(l_1)$. From $(k, h_1', h_2', h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w', w''$ there exists $\rho \in \mathrm{wrs}\,\varepsilon$ and $(m_1, l_2, \mu) \in w'(R(\rho))$ with $m_1 = l_1$ such that $k > 0 \Rightarrow (k-1, h_1''(l_1), h_2''(l_2)) \in (\iota\,\mu)(w'')$. Since $l_1 \in \mathrm{dom}(h_1) = \mathrm{dom_1}(w)$ we have $(m_1, l_2, \mu) \in w(R(\rho))$ too.

Otherwise, we are left to consider the case $h_1'(l_1) = h_1''(l_1)$ and $h_1(l_1) \neq h_1'(l_1)$ and from the fact $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R\, w, w'$ there exists $\rho \in \mathrm{wrs}\,\varepsilon$ and $(m_1, l_2, \mu) \in w(R(\rho))$ with $m_1 = l_1$ such that $k > 0$ implies $(k-1, h_1'(l_1), h_2'(l_2)) \in (\iota\,\mu)(w')$. By an argument as above, with can assume $h_2'(l_2) = h_2''(l_2)$ without loss of generality and we are done by type monotonicity.

Finally we need to consider allocation, i.e., we take $r \in \mathrm{dom}(w)$ and $(l_1, l_2, \mu) \in w''(r) \setminus w(r)$ and must prove $r \in R(\mathrm{als}\,\varepsilon)$ and $k > 0 \Rightarrow (k-1, h_1''(l_1), h_2''(l_2)) \in (\iota\,\mu)(w'')$. If $(l_1, l_2, \mu) \notin w'(r)$, then it follows directly from $(k, h_1', h_2', h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w', w''$. Otherwise, $(l_1, l_2, \mu) \in w'(r) \setminus w(r)$, so $(k, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R\, w, w'$ implies $r \in R(\mathrm{als}\,\varepsilon)$ and $k > 0 \Rightarrow (k-1, h_1'(l_1), h_2'(l_2)) \in (\iota\,\mu)(w')$. Then, if $h_1'(l_1) = h_1''(l_1)$ and $h_2'(l_2) = h_2''(l_2)$, we conclude using type monotonicity. But if $h_1'(l_1) \neq h_1''(l_1)$ or $h_2'(l_2) \neq h_2''(l_2)$, we use the hypothesis $(k, h_1', h_2', h_1'', h_2'') \in \mathbf{Q}_\varepsilon^R\, w', w''$. □

We begin with the compatibility lemmas for the rules App and Fix.

Lemma 18 *(App).* $\Pi \mid \Gamma \models e_1 \preceq e_2 : \tau_1 \to^\varepsilon \tau_2, \varepsilon_1$ *and* $\Pi \mid \Gamma \models e_1' \preceq e_2' : \tau_1, \varepsilon_2$ *implies* $\Pi \mid \Gamma \models e_1\, e_1' \preceq e_2\, e_2' : \tau_2, \varepsilon \cup \varepsilon_1 \cup \varepsilon_2$.

Proof. We unroll the definition of the logical relation: let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ — be arbitrary. Assume $R : \mathrm{FRV}(\varepsilon \cup \varepsilon_1 \cup \varepsilon_2) \hookrightarrow \mathrm{dom}(w)$, and $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We now must show that

$$(k, \gamma_1(e_1)\,\gamma_1(e_1'), \gamma_2(e_2)\,\gamma_2(e_2')) \in \mathcal{E}\,[\![\tau_2{}^{\varepsilon \cup \varepsilon_1 \cup \varepsilon_2}]\!]^R\, w.$$

We proceed to unroll the definition of computations: let $j \leq k$, $h_1, h_2, f_1, f_2 \in \mathcal{H}$, $e_1''' \in \mathcal{E}$ and $h_1^\perp \in \mathcal{H}$ be arbitrary. Assume $(k, h_1, h_2) \in \mathbf{P}_{\varepsilon \cup \varepsilon_1 \cup \varepsilon_2}^R\, w$, that

$$\langle \gamma_1(e_1)\,\gamma_1(e_1') \mid h_1 \cdot f_1 \rangle \xrightarrow{j} \langle e_1''' \mid h_1^\perp \rangle$$

and that $\mathrm{irr}\langle e_1''' \mid h_1^\perp \rangle$.

By the operational semantics, there must be $0 \leq i \leq j$, $e_1' \in \mathcal{E}$ and $h_1^\dagger \in \mathcal{H}$, such that

$$\langle \gamma_1(e_1) \mid h_1 \cdot f_1 \rangle \xrightarrow{i} \langle e_1' \mid h_1^\dagger \rangle$$

and

$$\langle e_1'\,\gamma_1(e_1') \mid h_1^\dagger \rangle \xrightarrow{j-i} \langle e_1''' \mid h_1^\perp \rangle$$

and $\mathrm{irr}\langle e_1' \mid h_1^\dagger\rangle$ holds. The first assumption of the lemma gives us that $(k, \gamma_1(e_1), \gamma_2(e_2)) \in \mathcal{E}\,[\![\tau_1 \to^\varepsilon \tau_2{}^{\varepsilon_1}]\!]^R\, w$ and Precondition Strengthening gives $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon_1} w$. Hence, there are $w' \sqsupseteq w$ with $\mathrm{dom}(w') = \mathrm{dom}(w)$ as well as $e_2' \in \mathcal{E}$ and $h_1', h_2', f_1', f_2' \in \mathcal{H}$ such that $h_1^\dagger = h_1' \cdot f_1' \cdot f_1$ and

$$\langle \gamma_2(e_2) \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle e_2' \mid h_2' \cdot f_2' \cdot f_2\rangle,$$

with $(k-i, e_1', e_2') \in [\![\tau_1 \to^\varepsilon \tau_2]\!]^R\, w'$ as well as the postcondition $(k-i, h_1, h_2, h_1', h_2') \in \mathbf{Q}^R_{\varepsilon_1} w, w'$.

We observe $e_1' \in \mathcal{V}$ and turn the crank once more: there must be $0 \le i' \le j - i$, $e_1'' \in \mathcal{E}$ and $h_1^\ddagger \in \mathcal{H}$, such that

$$\langle \gamma_1(e_1^\dagger) \mid h_1' \cdot f_1' \cdot f_1\rangle \xrightarrow{i'} \langle e_1'' \mid h_1^\ddagger\rangle$$

and

$$\langle e_1'\, e_1'' \mid h_1^\ddagger\rangle \xrightarrow{j-i-i'} \langle e_1''' \mid h_1^\perp\rangle$$

and $\mathrm{irr}\langle e_1'' \mid h_1^\ddagger\rangle$ holds. The second assumption gives us $(k-i, \gamma_1(e_1^\dagger), \gamma_2(e_2^\dagger)) \in \mathcal{E}\,[\![\tau_1{}^{\varepsilon_2}]\!]^R\, w'$. Precondition Composition gives us $(k-i, h_1', h_2') \in \mathbf{P}^R_{\varepsilon\cup\varepsilon_1\cup\varepsilon_2} w'$ and using Precondition Strengthening then yields $(k-i, h_1', h_2') \in \mathbf{P}^R_{\varepsilon_2} w'$. Then there is $w'' \sqsupseteq w'$ with $\mathrm{dom}(w'') = \mathrm{dom}(w')$ as well as $e_2'' \in \mathcal{E}$ and $h_1'', h_2'', f_1'', f_2'' \in \mathcal{H}$ such that $h_1^\ddagger = h_1'' \cdot f_1'' \cdot f_1' \cdot f_1$ and

$$\langle \gamma_2(e_2^\dagger) \mid h_2' \cdot f_2' \cdot f_2\rangle \xrightarrow{*} \langle e_2'' \mid h_2'' \cdot f_2'' \cdot f_2' \cdot f_2\rangle,$$

with $(k-i-i', e_1'', e_2'') \in [\![\tau_1]\!]^R\, w''$ as well as the postcondition $(k-i-i', h_1', h_2', h_1'', h_2'') \in \mathbf{Q}^R_{\varepsilon_2} w', w''$. The latter, by Postcondition Weakening and Postcondition Composition, yields $(k-i-i', h_1, h_2, h_1'', h_2'') \in \mathbf{Q}^R_{\varepsilon_1\cup\varepsilon_2} w, w''$.

The key to the remaining, and proper, part of the proof is that we have $(k-i-i', e_1', e_2') \in [\![\tau_1 \to^\varepsilon \tau_2]\!]^R\, w''$. As $R : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w) = \mathrm{dom}(w'')$ we get that, since $(k-i-i', e_1'', e_2'') \in [\![\tau_1]\!]^R\, w''$, we have

$$\big(k-i-i', e_1'\, e_1'', e_2'\, e_2''\big) \in \mathcal{E}\,[\![\tau_2{}^\varepsilon]\!]^R\, w''.$$

Precondition Composition gives $(k-i-i', h_1'', h_2'') \in \mathbf{P}^R_{\varepsilon\cup\varepsilon_1\cup\varepsilon_2} w''$ and Precondition Strengthening then yields $(k-i-i', h_1'', h_2'') \in \mathbf{P}^R_{\varepsilon} w''$. Then there is $w''' \sqsupseteq w''$ with $\mathrm{dom}(w''') = \mathrm{dom}(w'')$ as well as $e_2''' \in \mathcal{E}$ and $h_1''', h_2''', f_1''', f_2''' \in \mathcal{H}$ such that $h_1^\perp = h_1''' \cdot f_1''' \cdot f_1'' \cdot f_1' \cdot f_1$ and

$$\langle e_2'\, e_2'' \mid h_2'' \cdot f_2'' \cdot f_2' \cdot f_2\rangle \xrightarrow{*} \langle e_2''' \mid h_2''' \cdot f_2''' \cdot f_2'' \cdot f_2' \cdot f_2\rangle,$$

with $(k-j, e_1''', e_2''') \in [\![\tau_2]\!]^R\, w'''$ as well as the postcondition $(k-j, h_1'', h_2'', h_1''', h_2''') \in \mathbf{Q}^R_{\varepsilon} w'', w'''$. The latter, by Postcondition Weakening and Postcondition Composition, yields $(k-j, h_1, h_2, h_1''', h_2''') \in \mathbf{Q}^R_{\varepsilon\cup\varepsilon_1\cup\varepsilon_2} w, w'''$. □

Lemma 19 *(Fix).* $\Pi, \Gamma, f : \tau_1 \to^\varepsilon \tau_2, x : \tau_1 \models e_1 \preceq e_2 : \tau_2, \varepsilon$ *implies* $\Pi, \Gamma \models \mathsf{fix}\, f(x).e_1 \preceq \mathsf{fix}\, f(x).e_2 : \tau_1 \to^\varepsilon \tau_2, \emptyset$.

Proof. We unroll the definition of the logical relation: let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. The assumption $R : \mathrm{FRV}(\emptyset) \hookrightarrow \mathrm{dom}(w)$ gives us nothing, but we do get that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. Write $e_1' = \mathsf{fix}\, f(x).\gamma_1(e_1)$ and $e_2' = \mathsf{fix}\, f(x).\gamma_2(e_2)$; we must show that

$$(k, e_1', e_2') \in \mathcal{E}\left[\!\!\left[\tau_1 \to^\varepsilon \tau_2{}^\emptyset\right]\!\!\right]^R w.$$

As both expressions are, in fact, values it will suffice to show

$$(k, e_1', e_2') \in [\![\tau_1 \to^\varepsilon \tau_2]\!]^R\, w.$$

We aim to prove by induction that for all $0 \le j \le k$ we have

$$(j, e_1', e_2') \in [\![\tau_1 \to^\varepsilon \tau_2]\!]^R\, w.$$

The base case is easy, since there is no termination in 0 steps here. So assume the above for $0 \le j < k$; we must prove that

$$(j+1, e_1', e_2') \in [\![\tau_1 \to^\varepsilon \tau_2]\!]^R\, w$$

is good too. Let $i \le j+1$ and pick $w' \sqsupseteq w$ with $R : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w')$. Let $v_1, v_2 \in \mathcal{V}$ be arbitrary with $(i, v_1, v_2) \in [\![\tau_1]\!]^R\, w'$. We must show

$$(i, e_1'\, v_1, e_2'\, v_2) \in \mathcal{E}\,[\![\tau_2{}^\varepsilon]\!]^R\, w'.$$

So, let $i' \le i$, $e_1' \in \mathcal{E}$ and $h_1, h_2, f_1, f_2, h_1^\dagger \in \mathcal{H}$ be arbitrary and assume $(i, h_1, h_2) \in \mathbf{P}^R_\varepsilon\, w'$ and that

$$\langle e_1'\ v_1 \mid h_1 \cdot f_1\rangle \xrightarrow{i'} \langle e_1' \mid h_1^\dagger\rangle$$

with $\mathrm{irr}\langle e_1' \mid h_1^\dagger\rangle$. Clearly, $i' > 0$, so the reduction must go like

$$\begin{aligned}\langle e_1'\ v_1 \mid h_1 \cdot f_1\rangle &\to \langle \gamma_1(e_1)\{e_1'/f, v_1/x\} \mid h_1 \cdot f_1\rangle \\ &\xrightarrow{i'-1} \langle e_1' \mid h_1^\dagger\rangle\end{aligned}$$

It is now time to make use of the lemma's assumption, which we can instantiate with $i-1$, w', R and the *extended* substitutions: γ_1, e_1'/f, v_1/x, and γ_2, e_2'/f, v_2/x. These substitutions are related at $i-1$ due to downwards closure, assumptions and, in the case of f, induction hypothesis, since $i-1 \le j$. This gets us

$$(i-1, \gamma_1(e_1)\{e_1'/f, v_1/x\}, \gamma_2(e_2)\{e_2'/f, v_2/x\}) \in \mathcal{E}\,[\![\tau_2{}^\varepsilon]\!]^R\, w',$$

which, after unfolding the definitions, concludes the proof. □

Lemma 20 *(Susp). If $\rho, \Pi \mid \Gamma \models e_1 \preceq e_2 : \tau, \varepsilon$ with $\rho \notin \mathrm{FRV}(\Gamma)$, then $\Pi \mid \Gamma \models \mathsf{susp}\, e_1 \preceq \mathsf{susp}\, e_2 : \forall \rho \notin \Delta.\, \tau^\varepsilon, \emptyset$ where $\Delta = \mathrm{FRV}(\tau, \varepsilon) \backslash \rho$.*

Proof. Take any $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$, $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ such that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We need to show that $(k, \mathsf{susp}\,\hat{e}_1, \mathsf{susp}\,\hat{e}_2) \in \mathcal{E}\,\left[\!\left[\forall \rho \notin \Delta.\, \tau^{\varepsilon\emptyset}\right]\!\right]^R\, w$, where $\hat{e}_i = \gamma_i(e_i)$. However, since $\mathsf{susp}\, e$ is always a value, it suffices to show that, $(k, \mathsf{susp}\,\hat{e}_1, \mathsf{susp}\,\hat{e}_2) \in [\![\forall \rho \notin \Delta.\, \tau^\varepsilon]\!]^R\, w$. This in turn requires us to show that $(j, \hat{e}_1, \hat{e}_2) \in \mathcal{E}\,[\![\tau^\varepsilon]\!]^{R[\rho \mapsto r]}\, w'$ for some $j \le k$, $w' \sqsupseteq w$ and $r \in |w'|$ such that $R[\rho \mapsto r] : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w')$. To prove it, we instantiate the assumption with j, w', $R[\rho \mapsto r]$, γ_1 and γ_2. We already have the first two prerequisites, and for the third one, Environment Extension gives us $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^{R[\rho \mapsto r]}\, w'$, since $\rho \notin \mathrm{FRV}(\Gamma)$. □

Lemma 21 *(Mask-post). The postcondition relation is closed under simultaneous allocation and masking out of a region: if, for $r \notin |w_1|$, we have $w_1 \to_{\mathrm{reg}(r)} w_1'$, $w_2' \to_{\mathrm{mask}(r)} w_2$ and $(k, h_{11}, h_{21}, h_{12}, h_{22}) \in \mathbf{Q}^{R[\rho \mapsto r]}_\varepsilon\, w_1', w_2'$, then $(k, h_{11}, h_{21}, h'_{12}, h'_{22}) \in \mathbf{Q}^R_{\varepsilon-\rho}\, w_1, w_2$, where $h'_{i2} = h_{i2}\big|_{\mathrm{dom_i}(w_2)}$.*

Proof. We have a range of things to show here. Note first that the $\mathrm{dom}(h_{i1}) = \mathrm{dom_i}(w_1)$, since the newly allocated region has to be empty, and $\mathrm{dom}(h_{i2}) = \mathrm{dom_i}(w_1')$. The other heap domains are explicitly cut down to size.

For the writes, assume $l_1 \in \mathrm{dom}(h_{11})$ such that $h_{11}(l_1) \ne h'_{12}(l_1)$. We can use these to instantiate our assumption, and get $\sigma \in \mathrm{wrs}\,\varepsilon$ and $(m_1, l_2, \mu) \in w_1'(R[\rho \mapsto r](\sigma))$ such that $l_1 = m_1$ and $k > 0 \Longrightarrow (k-1, h_{12}(l_1), h_{22}(l_2)) \in (\iota\,\mu)(w_2')$. Since the new region r is empty in w_1', this means $R[\rho \mapsto r](\sigma) \ne r$, and so $\sigma \ne \rho$ – which gets us $\sigma \in \mathrm{wrs}\,\varepsilon - \rho$ and $(m_1, l_2, \mu) \in w_1(R(\sigma))$. Since both l_i are outside the region r, they have to belong to the respective $\mathrm{dom_i}(w_2)$, and so also to $\mathrm{dom}(h'_{i2})$. Now, by type monotonicity we can replace w_2' with w_2 in our assumption, and get $k > 0 \Longrightarrow (k-1, h'_{12}(l_1), h'_{22}(l_2)) \in (\iota\,\mu)(w_2)$, which was our last obligation in this part of the proof. The other obligation on the writes proceeds symmetrically.

The last obligation we have left are the allocations. Take any $r_1 \in \mathrm{dom}(w_2)$ and $(l_1, l_2, \mu) \in w_2(r_1) \setminus w_1(r_1)$ (this means, in particular, that $r \ne r_1$). We can use these to instantiate the assumption about postcondition, since $w_i(r_1) = w_i'(r_1)$, and get $r_1 \in R[\rho \mapsto r](\mathrm{als}\,\varepsilon)$ and $k > 0 \Longrightarrow (k-1, h_{12}(l_1), h_{22}(l_2)) \in (\iota\,\mu)(w_2')$. Since $r_1 \ne r$, we get $r_1 \in R(\mathrm{als}\,\varepsilon - \rho)$, and by type monotonicity and the fact that $l_i \in \mathrm{dom_i}(w_2)$ we get $k > 0 \Longrightarrow (k-1, h'_{12}(l_1), h'_{22}(l_2)) \in (\iota\,\mu)(w_2)$, which ends the proof. □

Lemma 22 *(Mask). If $\Pi, \rho \mid \Gamma \models e_1 \preceq e_2 : \tau, \varepsilon$ and $\rho \notin \mathrm{FRV}(\Gamma, \tau)$, then $\Pi \mid \Gamma \models e_1 \preceq e_2 : \tau, \varepsilon - \rho$.*

Proof. Take any $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$, and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$. Assume that $R : \mathrm{FRV}(\varepsilon - \rho) \hookrightarrow \mathrm{dom}(w)$ and $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We need to show that $(k, \hat{e}_1, \hat{e}_2) \in \mathcal{E}\,\left[\!\left[\tau^{\varepsilon-\rho}\right]\!\right]^R\, w$, where $\hat{e}_i = \gamma_i(e_i)$. Take $r \notin |w|$, w_1 defined by $w \to_{\mathrm{reg}(r)} w_1$, $R' = R[\rho \mapsto r] : \Pi, \rho \to |w_1|$. By Environment Extension we get $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^{R'}\, w_1$. Since $R' : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w_1)$, the assumption provides us with $(k, \hat{e}_1, \hat{e}_2) \in \mathcal{E}\,[\![\tau^\varepsilon]\!]^{R'}\, w_1$. We proceed by unrolling the definition of relatedness of closed expressions, and take any $j \le k$, e_1', h_1, h_2, f_1, f_2 and $h_1^\dagger$, such that $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon-\rho}\, w$, $\langle \hat{e}_1 \mid h_1 \cdot f_1\rangle \mapsto^j \langle e_1' \mid h_1^\dagger\rangle$ and $\mathrm{irr}\langle e_1' \mid h_1^\dagger\rangle$. We use these variables to instantiate the relatedness assumption: note that $(k, h_1, h_2) \in \mathbf{P}^R_\varepsilon\, w_1$, since r is empty in w_1. This gives us a new world, $w_2 \sqsupseteq w_1$, e_2', h_1', h_2', f_1' and f_2' such that

- $\mathrm{dom}(w_2) = \mathrm{dom}(w_1)$
- $h_1^\dagger = h_1' \cdot f_1' \cdot f_1$
- $\langle \hat{e}_2 \mid h_2 \cdot f_2\rangle \mapsto^* \langle e_2' \mid h_2' \cdot f_2' \cdot f_2\rangle$

- $(k-j, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^{R'}\, w_1, w_2$
- $(k-j, e_1', e_2') \in [\![\tau]\!]^{R'}\, w_2$

Take w_3 as defined by $w_2 \to_{\mathrm{mask}(r)} w_3$, write $h_1' = h_1'' \cdot f_1''$ with $\mathrm{dom}(h_1'') = \mathrm{dom}_1(w_3)$ and similarly for h_2', h_2'', f_2''. We get $(k-j, h_1, h_2, h_1'', h_2'') \in \mathbf{Q}_{\varepsilon-\rho}^R\, w, w_3$ by Lemma 21, and by Lemma 8 we get $(k-j, e_1', e_2') \in [\![\tau]\!]^R\, w_3$. □

Lemma 23 *(Lookup).* $\Pi \mid \Gamma \models e_1 \preceq e_2 : \mathsf{ref}_\rho \tau, \varepsilon$ *implies* $\Pi \mid \Gamma \models !e_1 \preceq !e_2 : \tau, \varepsilon \cup \{rd_\rho\}$.

Proof. We unroll the definition of the logical relation: let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume $R : \mathrm{FRV}(\varepsilon \cup \{rd_\rho\}) \hookrightarrow \mathrm{dom}(w)$ and that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We now must show that

$$(k, !\,\gamma_1(e_1), !\,\gamma_2(e_2)) \in \mathcal{E}\left[\!\left[\tau^{\varepsilon \cup \{rd_\rho\}}\right]\!\right]^R w.$$

We proceed to unroll the definition of the relation on computations: let $j \le k$, $e_1'' \in \mathcal{E}$ and $h_1, h_2, f_1, f_2, g_1'' \in \mathcal{H}$ be arbitrary. Assume $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon \cup \{rd_\rho\}}\, w$, that

$$\langle !\,\gamma_1(e_1) \mid h_1 \cdot f_1\rangle \xrightarrow{j} \langle e_1'' \mid g_1''\rangle$$

and $\mathrm{irr}\langle e_1'' \mid g_1''\rangle$.

By the operational semantics, there must be $0 \le i \le j$, $e_1' \in \mathcal{E}$ and $g_1' \in \mathcal{H}$, such that

$$\langle \gamma_1(e_1) \mid h_1 \cdot f_1\rangle \xrightarrow{i} \langle e_1' \mid g_1'\rangle, \quad \langle !\,e_1' \mid g_1'\rangle \xrightarrow{j-i} \langle e_1'' \mid g_1''\rangle$$

and $\mathrm{irr}\langle e_1' \mid g_1'\rangle$ holds. The assumption of the lemma and Precondition Strengthening gives us $w' \sqsupseteq w$ with $\mathrm{dom}(w') = \mathrm{dom}(w)$ as well as $e_2' \in \mathcal{E}$ and g_2', h_1', h_2', f_1', f_2' such that

$$\langle \gamma_2(e_2) \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle e_2' \mid g_2'\rangle,$$

with $g_1' = h_1' \cdot f_1' \cdot f_1$, $g_2' = h_2' \cdot f_2' \cdot f_2$ and $(k-i, e_1', e_2') \in [\![\mathsf{ref}_\rho \tau]\!]^R\, w'$ as well as $(k-i, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R\, w, w'$.

Let us consider the fact $(k-i, e_1', e_2') \in [\![\mathsf{ref}_\rho \tau]\!]^R\, w'$. By assumption, $R(\rho) \in \mathrm{dom}(w) = \mathrm{dom}(w')$ and so we must have $l_1, l_2 \in \mathcal{L}$ and $\mu \in \widehat{\mathbf{T}}$ such that $e_1' = l_1$, $e_2' = l_2$, $(l_1, l_2, \mu) \in w'(R(\rho))$ and $[\![\tau]\!]^R\, w' \overset{k-i}{=} (\iota\, \mu)(w')$.

We must have $i = j-1$ and the entire left hand side reduction must look like

$$\langle !\,\gamma_1(e_1) \mid h_1 \cdot f_1\rangle \xrightarrow{j-1} \langle !\,l_1 \mid g_1'\rangle \to \langle h_1'(l_1) \mid g_1'\rangle,$$

in particular $e_1'' = h_1'(l_1)$ and $g_1'' = g_1'$. Moreover, since

$$\langle \gamma_2(e_2) \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle l_2 \mid g_2'\rangle,$$

and $l_2 \in \mathrm{dom}_2(w') = \mathrm{dom}(h_2') \subseteq \mathrm{dom}(g_2')$ we get a similar reduction on the right hand side

$$\langle !\,\gamma_2(e_2) \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle !\,l_2 \mid g_2'\rangle \to \langle h_2'(l_2) \mid g_2'\rangle.$$

We are left to verify only that $(k-j, h_1'(l_1), h_2'(l_2)) \in [\![\tau]\!]^R\, w'$. Since $[\![\tau]\!]^R\, w' \overset{k-j+1}{=} (\iota\, \mu)(w')$, it suffices to prove $(k-j, h_1'(l_1), h_2'(l_2)) \in (\iota\, \mu)(w')$. But this is immediate as Precondition Composition and Precondition Strengthening together gives $(k-i, h_1', h_2') \in \mathbf{P}^R_{\{rd_\rho\}}\, w'$. □

Lemma 24 *(Assign).* $\Pi \mid \Gamma \models e_1 \preceq e_2 : \mathsf{ref}_\rho \tau, \varepsilon$ *and* $\Pi \mid \Gamma \models e_1^\dagger \preceq e_2^\dagger : \tau, \varepsilon^\dagger$ *implies* $\Pi \mid \Gamma \models e_1 := e_1^\dagger \preceq e_2 := e_2^\dagger : 1, \varepsilon \cup \varepsilon^\dagger \cup \{wr_\rho\}$.

Proof. We unroll the definition of the logical relation: let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume $R : \mathrm{FRV}(\varepsilon \cup \varepsilon^\dagger \cup \{wr_\rho\})) \hookrightarrow \mathrm{dom}(w)$ and that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We now must show that

$$(k,\ \gamma_1(e_1) := \gamma_1(e_1^\dagger),\ \gamma_2(e_2) := \gamma_2(e_2^\dagger)) \in \mathcal{E}\left[\!\left[1^{\varepsilon \cup \varepsilon^\dagger \cup \{wr_\rho\}}\right]\!\right]^R w.$$

We proceed to unroll the definition of computations: let $j \le k$, $h_1, h_2, f_1, f_2 \in \mathcal{H}$, $e_1''' \in \mathcal{E}$ and $h_1^\perp \in \mathcal{H}$ be arbitrary. Assume $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon \cup \varepsilon^\dagger \cup \{wr_\rho\}}\, w$, that

$$\langle \gamma_1(e_1) := \gamma_1(e_1^\dagger) \mid h_1 \cdot f_1 \rangle \xrightarrow{j} \langle e_1''' \mid h_1^\perp \rangle$$

and $\mathrm{irr}\langle e_1''' \mid h_1^\perp \rangle$.

By the operational semantics, there must be $0 \le i \le j$, $e_1' \in \mathcal{E}$ and $h_1^\dagger \in \mathcal{H}$, such that

$$\langle \gamma_1(e_1) \mid h_1 \cdot f_1 \rangle \xrightarrow{i} \langle e_1' \mid h_1^\dagger \rangle$$

and

$$\langle e_1' := \gamma_1(e_1^\dagger) \mid h_1^\dagger \rangle \xrightarrow{j-i} \langle e_1''' \mid h_1^\perp \rangle$$

and $\mathrm{irr}\langle e_1' \mid h_1' \rangle$ holds. The first assumption of the lemma gives us that $(k, \gamma_1(e_1), \gamma_2(e_2)) \in \mathcal{E}\left[\!\left[\mathsf{ref}_\rho \tau^\varepsilon\right]\!\right]^R w$ and Precondition Strengthening gives $(k, h_1, h_2) \in \mathbf{P}_\varepsilon^R w$. Hence, there is $w' \sqsupseteq w$ with $\mathrm{dom}(w') = \mathrm{dom}(w)$ as well as $e_2' \in \mathcal{E}$ and $h_1', h_2', f_1', f_2' \in \mathcal{H}$ such that $h_1^\dagger = h_1' \cdot f_1' \cdot f_1$ and

$$\langle \gamma_2(e_2) \mid h_2 \rangle \xrightarrow{*} \langle e_2' \mid h_2' \cdot f_2' \cdot f_2 \rangle,$$

with $(k-i, e_1', e_2') \in \left[\!\left[\mathsf{ref}_\rho \tau\right]\!\right]^R w'$ as well as $(k-i, h_1, h_2, h_1', h_2') \in \mathbf{Q}_\varepsilon^R w, w'$.

An easy consequence is that $e_1' \in \mathcal{V}$ and so we can continue: there must be $0 \le i' \le j - i$, $e_1'' \in \mathcal{E}$ and $h_1^\ddagger \in \mathcal{H}$, such that

$$\langle \gamma_1(e_1^\dagger) \mid h_1' \cdot f_1' \cdot f_1' \rangle \xrightarrow{i'} \langle e_1'' \mid h_1^\ddagger \rangle$$

and

$$\langle e_1' := e_1'' \mid h_1^\ddagger \rangle \xrightarrow{j-i-i'} \langle e_1''' \mid h_1^\perp \rangle$$

and $\langle \mathrm{irr} \mid e_1'' \rangle h_1''$ holds. The second assumption gives us $(k-i, \gamma_1(e_1^\dagger), \gamma_2(e_2^\dagger)) \in \mathcal{E}\left[\!\left[\tau^{\varepsilon^\dagger}\right]\!\right]^R w'$. Precondition Composition gives $(k-i, h_1', h_2') \in \mathbf{P}^R_{\varepsilon \cup \varepsilon' \cup \{wr_\rho\}} w'$ and using Precondition Strengthening then yields $(k-i, h_1', h_2') \in \mathbf{P}^R_{\varepsilon^\dagger} w'$. Then there is $w'' \sqsupseteq w'$ with $\mathrm{dom}(w'') = \mathrm{dom}(w')$ as well as $e_2'' \in \mathcal{E}$ and $h_1'', h_2'', f_1'', f_2'' \in \mathcal{H}$ such that $h_1^\ddagger = h_1'' \cdot f_1'' \cdot f_1' \cdot f_1$ and

$$\langle \gamma_2(e_2^\dagger) \mid h_2' \rangle \xrightarrow{*} \langle e_2'' \mid h_2'' \cdot f_2'' \cdot f_2' \cdot f_2 \rangle,$$

with $(k-i-i', e_1'', e_2'') \in [\![\tau]\!]^R w''$ as well as the postcondition $(k-i-i', h_1', h_2', h_1'', h_2'') \in \mathbf{Q}^R_{\varepsilon^\dagger} w', w''$. The latter, by Postcondition Composition, yields $(k-i-i', h_1, h_2, h_1'', h_2'') \in \mathbf{Q}^R_{\varepsilon \cup \varepsilon^\dagger} w, w''$.

Now, finally, for the computation proper. Since we have that $R(\rho) \in \mathrm{dom}(w) = \mathrm{dom}(w'')$, there are $l_1, l_2 \in \mathcal{L}$ with $e_1' = l_1$ and $e_2' = l_2$ and such that there is $\mu \in \widehat{\mathbf{T}}$ with $(l_1, l_2, \mu) \in w''(R(\rho))$ and $[\![\tau]\!]^R w'' \overset{k-i-i'}{=} (\iota\, \mu)(w'')$. In particular, $l_1 \in \mathrm{dom}_1(w'') \subseteq h_1''$ and the final step of the left hand side reduction must go

$$\begin{aligned}\langle e_1' := e_1'' \mid h_1'' \cdot f_1'' \cdot f_1' \cdot f_1 \rangle &= \langle l_1 := e_1'' \mid h_1'' \cdot f_1'' \cdot f_1' \cdot f_1 \rangle \\ &\to \langle () \mid h_1''[l_1 \mapsto e_1''] \cdot f_1'' \cdot f_1' \cdot f_1 \rangle = \langle e_1''' \mid h_1''' \rangle.\end{aligned}$$

On the right hand side, we have a complete reduction as $l_2 \in \mathrm{dom}(h_2'')$:

$$\begin{aligned}\langle \gamma_2(e_2) := \gamma_2(e_2^\dagger) \mid h_2 \cdot f_2 \rangle &\xrightarrow{*} \langle l_2 := \gamma_2(e_2^\dagger) \mid h_2' \cdot f_2' \cdot f_2 \rangle \\ &\xrightarrow{*} \langle l_2 := e_2'' \mid h_2'' \cdot f_2'' \cdot f_2' \cdot f_2 \rangle \\ &\to \langle () \mid h_2''[l_2 \mapsto e_2''] \cdot f_2'' \cdot f_2' \cdot f_2 \rangle.\end{aligned}$$

The return values obviously have the correct type. All that remains is to verify that we do indeed have

$$(k-i-i'-1, h_1, h_2, h_1''[l_1 \mapsto e_1''], h_2''[l_2 \mapsto e_2'']) \in \mathbf{Q}^R_{\varepsilon \cup \varepsilon^\dagger \cup \{wr_\rho\}} w, w'',$$

but that is an easy consequence of the fact that we have $(k-i-i'-1, e_1'', e_2'') \in [\![\tau]\!]^R w''$ and that $[\![\tau]\!]^R w''$ and $(\iota\, \mu)(w'')$ are sufficiently close. □

Lemma 25 *(Alloc).* $\Pi \mid \Gamma \models e_1 \preceq e_2 : \tau, \varepsilon$ *implies*

$$\Pi \mid \Gamma \models \mathtt{ref}\, e_1 \preceq \mathtt{ref}\, e_2 : \mathsf{ref}_\rho \tau, \varepsilon \cup \{al_\rho\}$$

Proof. We unroll the definition of the logical relation: let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume $R : \mathrm{FRV}(\varepsilon \cup \{al_\rho\}) \hookrightarrow \mathrm{dom}(w)$ and that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We now must show that

$$(k, \mathtt{ref}\,\gamma_1(e_1), \mathtt{ref}\,\gamma_2(e_2)) \in \mathcal{E}\left[\!\left[\mathsf{ref}_\rho \tau^{\varepsilon \cup \{al_\rho\}}\right]\!\right]^R w.$$

We proceed to unroll the definition of computations: let $j \le k$, $h_1, h_2, f_1, f_2 \in \mathcal{H}$, $e_1'' \in \mathcal{E}$ and $h_1^\ddagger \in \mathcal{H}$ be arbitrary. Assume $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon \cup \{al_\rho\}}\, w$, that

$$\langle \mathtt{ref}\,\gamma_1(e_1) \,|\, h_1 \cdot f_1\rangle \xrightarrow{j} \langle e_1'' \,|\, h_1^\ddagger\rangle$$

and that $\mathrm{irr}\langle e_1'' \,|\, h_1^\ddagger\rangle$.

By the operational semantics, there must be $0 \le i \le j$, $e_1' \in \mathcal{E}$ and $h_1^\dagger \in \mathcal{H}$, such that

$$\langle \gamma_1(e_1) \,|\, h_1 \cdot f_1\rangle \xrightarrow{i} \langle e_1' \,|\, h_1^\dagger\rangle$$

and

$$\langle \mathtt{ref}\, e_1' \,|\, h_1^\dagger\rangle \xrightarrow{j-i} \langle e_1'' \,|\, h_1^\ddagger\rangle$$

and $\mathrm{irr}\langle e_1' \,|\, h_1^\dagger\rangle$ holds. The assumption gives us that we have $(k, \gamma_1(e_1), \gamma_2(e_2)) \in \mathcal{E}\,[\![\tau^\varepsilon]\!]^R\, w$ and Precondition Strengthening gives $(k, h_1, h_2) \in \mathbf{P}^R_\varepsilon\, w$. Then, there is $w' \sqsupseteq w$ with $\mathrm{dom}(w') = \mathrm{dom}(w)$ as well as $e_2' \in \mathcal{E}$ and $h_1', h_2', f_1', f_2' \in \mathcal{H}$ such that $h_1^\dagger = h_1' \cdot f_1' \cdot f_1$ and

$$\langle \gamma_2(e_2) \,|\, h_2 \cdot f_2\rangle \xrightarrow{*} \langle e_2' \,|\, h_2' \cdot f_2' \cdot f_2\rangle,$$

with $(k - i, e_1', e_2') \in [\![\tau]\!]^R\, w'$ as well as the postcondition $(k - i, h_1, h_2, h_1', h_2') \in \mathbf{Q}^R_\varepsilon\, w, w'$.

As before, we now exploit the fact that we must have $e_1' \in \mathcal{V}$. This means that the entire left hand side reduction must go like

$$\begin{aligned}\langle \mathtt{ref}\,\gamma_1(e_1) \,|\, h_1 \cdot f_1\rangle &\xrightarrow{i} \langle \mathtt{ref}\, e_1' \,|\, h_1' \cdot f_1' \cdot f_1\rangle \\ &\to \langle l_1 \,|\, h_1'[l_1 \mapsto e_1'] \cdot f_1' \cdot f_1\rangle = \langle e_1'' \,|\, h_1^\ddagger\rangle\end{aligned}$$

for $l_1 \notin \mathrm{dom}(h_1' \cdot f_1' \cdot f_1)$, i.e., the heap was expanded. On the right hand side, we can give the following reduction for $l_2 \notin \mathrm{dom}(h_2')$:

$$\langle \mathtt{ref}\,\gamma_2(e_2) \,|\, h_1 \cdot f_1\rangle \xrightarrow{*} \langle \mathtt{ref}\, e_2' \,|\, h_2' \cdot f_2' \cdot f_2\rangle \to \langle l_2 \,|\, h_2'[l_2 \mapsto e_2'] \cdot f_2' \cdot f_2\rangle.$$

Note that $l_1 \notin \mathrm{dom}(h_1') = \mathrm{dom}_1(w')$ so $l_1 \notin \mathrm{dom}_1(w')$ and similarly $l_2 \notin \mathrm{dom}_2(w')$. Writing $r = R(\rho)$ we have $r \in \mathrm{dom}(w) = \mathrm{dom}(w')$. This justifies the building of a new world $w'' \in \mathbf{W}$ by the transition $w' \to_{al(r, l_1, l_2, \mu)} w''$ where we naturally set $\mu = \iota^{-1}([\![\tau]\!]^R)$. We do, in other words, extend the world to match the allocation of the operational semantics.

Since $w'' \sqsupseteq w$ and $\mathrm{dom}(w'') = \mathrm{dom}(w)$ it remains to prove that we return appropriately typed values, i.e., that $(k - i - 1, l_1, l_2) \in \left[\!\left[\mathsf{ref}_\rho \tau\right]\!\right]^R\, w''$ holds, and that the postcondition of the entire computation holds, i.e., that

$$(k - i - 1, h_1, h_2, h_1'[l_1 \mapsto e_1'], h_2'[l_2 \mapsto e_2']) \in \mathbf{Q}^R_{\varepsilon \cup \{al_\rho\}}\, w, w''.$$

The former is immediate by the fact that $\iota\, \mu = \iota(\iota^{-1}([\![\tau]\!]^R)) = [\![\tau]\!]^R$. And the latter comes down to showing

$$k - i - 1 > 0 \Rightarrow (k - i - 1 - 1, e_1', e_2') \in [\![\tau]\!]^R\, w''$$

which is a consequence of having $(k - i, e_1', e_2') \in [\![\tau]\!]^R\, w'$. □

Finally, the compatibility lemma for the rule T-RegInst is a bit more tedious. It will necessitate few auxiliary lemmas.

Lemma 26. *Let* $R : \mathcal{RV} \to \mathcal{RN}$ *with* $\sigma \in \mathrm{dom}(R)$. *If* $\mathrm{wf}(\tau)$, $\sigma \notin \mathrm{FRV}(\tau, \varepsilon)$, *and* $[\![\tau]\!]^{R[\rho \mapsto R(\sigma)]} = [\![\tau[\sigma/\rho]]\!]^R$ *then* $\mathcal{E}\,[\![\tau^\varepsilon]\!]^{R[\rho \mapsto R(\sigma)]} = \mathcal{E}\left[\!\left[\tau[\sigma/\rho]^{\varepsilon[\sigma/\rho]}\right]\!\right]^R$.

Proof. We just have to prove that $\mathbf{P}^{R[\rho \mapsto R(\sigma)]}_\varepsilon\, w = \mathbf{P}^R_{\varepsilon[\sigma/\rho]}\, w$ and $\mathbf{Q}^{R[\rho \mapsto R(\sigma)]}_\varepsilon\, w = \mathbf{Q}^R_{\varepsilon[\sigma/\rho]}\, w$. This follows from the equalities:

- $R[\rho \mapsto R(\sigma)](\mathrm{rds}\,\varepsilon) = R(\mathrm{rds}\,\varepsilon[\sigma/\rho])$
- $R[\rho \mapsto R(\sigma)](\mathrm{wrs}\,\varepsilon) = R(\mathrm{wrs}\,\varepsilon[\sigma/\rho])$
- $R[\rho \mapsto R(\sigma)](\mathrm{als}\,\varepsilon) = R(\mathrm{als}\,\varepsilon[\sigma/\rho])$ □

Lemma 27. *Let $R : \mathcal{RV} \to \mathcal{RN}$ with $\sigma \in \mathrm{dom}(R)$. If $\mathrm{wf}(\tau)$ and $\sigma \notin \mathrm{FRV}(\tau)$ then $[\![\tau]\!]^{R[\rho \mapsto R(\sigma)]} = [\![\tau[\sigma/\rho]]\!]^R$.*

Proof. First notice that $R[\rho \mapsto R(\sigma)](\mathrm{FRV}(\tau)) \not\subseteq |w|$ iff $R(\mathrm{FRV}(\tau[\sigma/\rho])) \not\subseteq |w|$, in which case both sets are indeed empty by the definition of $[\![\tau]\!]$, since R does not map free region variables of τ to regions of w.

So suppose that both $R[\rho \mapsto R(\sigma)](\mathrm{FRV}(\tau)) \subseteq |w|$ and $R(\mathrm{FRV}(\tau[\sigma/\rho])) \subseteq |w|$, and let us continue the proof by induction on τ:

- For types 1, int and $\tau_1 \times \tau_2$ the proof is straightforward.
- Assume $[\![\tau]\!]^{R[\rho \mapsto R(\sigma)]} = [\![\tau[\sigma/\rho]]\!]^R$ we prove $[\![\mathsf{ref}_\alpha \tau]\!]^{R[\rho \mapsto R(\sigma)]} = [\![(\mathsf{ref}_\alpha \tau)[\sigma/\rho]]\!]^R$:
  - If $\alpha \neq \rho$ then $(\mathsf{ref}_\alpha \tau)[\sigma/\rho] = \mathsf{ref}_\alpha(\tau[\sigma/\rho])$ and since $R[\rho \mapsto R(\sigma)](\alpha) = R(\alpha)$ the equality follows directly from the induction hypothesis.
  - If $\alpha = \rho$, then $(\mathsf{ref}_\alpha \tau)[\sigma/\rho] = \mathsf{ref}_\sigma(\tau[\sigma/\rho])$ and we show that $[\![\mathsf{ref}_\rho \tau]\!]^{R[\rho \mapsto R(\sigma)]} = [\![(\mathsf{ref}_\sigma(\tau[\sigma/\rho]))]\!]^R$. To do so, we use the equality $R(\sigma) = R[\rho \mapsto R(\sigma)](\rho)$ and the induction hypothesis.
- Assume $[\![\tau_1]\!]^{R[\rho \mapsto R(\sigma)]} = [\![\tau_1[\sigma/\rho]]\!]^R$ and $[\![\tau_2]\!]^{R[\rho \mapsto R(\sigma)]} = [\![\tau_2[\sigma/\rho]]\!]^R$, we prove $[\![\tau_1 \to^\varepsilon \tau_2]\!]^{R[\rho \mapsto R(\sigma)]} = [\![(\tau_1 \to^\varepsilon \tau_2)[\sigma/\rho]]\!]^R$.
  The equality is simply proved using the three following facts:
  - The logical equivalence between the injectivity of $R[\rho \mapsto R(\sigma)]$ on $\mathrm{FRV}(\varepsilon)$ and the injectivity of R on $\mathrm{FRV}(\varepsilon[\sigma/\rho])$, from Lemma 28.
  - The first induction hypothesis $[\![\tau_1]\!]^{R[\rho \mapsto R(\sigma)]} = [\![\tau_1[\sigma/\rho]]\!]^R$.
  - The equality $\mathcal{E}\,[\![\tau_2{}^\varepsilon]\!]^{R[\rho \mapsto R(\sigma)]} = \mathcal{E}\,[\![\tau_2[\sigma/\rho]^{\varepsilon[\sigma/\rho]}]\!]^R$, from Lemma 26 with the induction hypothesis.
- Assume $[\![\tau]\!]^{R[\alpha \mapsto r][\rho \mapsto R(\sigma)]} = [\![\tau[\sigma/\rho]]\!]^{R[\alpha \mapsto r]}$ we prove $[\![\forall \alpha \notin \Delta.\, \tau^\varepsilon]\!]^{R[\rho \mapsto R(\sigma)]} = [\![(\forall \alpha \notin \Delta.\, \tau^\varepsilon)[\sigma/\rho]]\!]^R$. Again, the equality follows from the two following facts:
  - The logical equivalence between the injectivity of $R[\alpha \mapsto r][\rho \mapsto R(\sigma)]$ on $\mathrm{FRV}(\varepsilon)$ and the injectivity of $R[\alpha \mapsto r]$ on $\mathrm{FRV}(\varepsilon[\sigma/\rho])$, from Lemma 28.
  - The equality $\mathcal{E}\,[\![\tau^\varepsilon]\!]^{R[\alpha \mapsto r][\rho \mapsto R(\sigma)]} = \mathcal{E}\,[\![\tau[\sigma/\rho]^{\varepsilon[\sigma/\rho]}]\!]^{R[\alpha \mapsto r]}$, from Lemma 26 with the induction hypothesis. □

Lemma 28 *(Injectivity conservation). Let $R : \Pi \to F$, $\mathrm{FRV}(\varepsilon) \subseteq \Pi$ and $\alpha, \beta \in \mathcal{RV}$, with $\beta \notin \mathrm{FRV}(\varepsilon)$ then $R[\alpha \mapsto R(\beta)] : \mathrm{FRV}(\varepsilon) \hookrightarrow F$ iff $R : \mathrm{FRV}(\varepsilon[\beta/\alpha]) \hookrightarrow F$*

Proof. If $\alpha \notin \mathrm{FRV}(\varepsilon)$, then $\mathrm{FRV}(\varepsilon[\beta/\alpha]) = \mathrm{FRV}(\varepsilon)$ and $R[\alpha \mapsto R(\beta)]_{|\,\mathrm{FRV}(\varepsilon)} = R_{|\,\mathrm{FRV}(\varepsilon)}$, so the equivalence holds. Otherwise, suppose $\alpha \in \mathrm{FRV}(\varepsilon)$.

- If $R[\alpha \mapsto R(\beta)] : \mathrm{FRV}(\varepsilon) \hookrightarrow F$, then for any $\gamma \in \mathrm{FRV}(\varepsilon) - \alpha$, $R[\alpha \mapsto R(\beta)](\alpha) \neq R[\alpha \mapsto R(\beta)](\gamma)$. So for any $\gamma \in \mathrm{FRV}(\varepsilon[\beta/\alpha]) \setminus \{\beta\}$, $R(\beta) \neq R(\gamma)$ since $\beta \notin \mathrm{FRV}(\varepsilon)$, i.e. $R : \mathrm{FRV}(\varepsilon[\beta/\alpha]) \hookrightarrow F$.
- Reciprocally, suppose $R : \mathrm{FRV}(\varepsilon[\beta/\alpha]) \hookrightarrow F$, then for any $\gamma \in \mathrm{FRV}(\varepsilon) \setminus \{\alpha\}$, $R(\beta) \neq R(\gamma)$. So for any $\gamma \in \mathrm{FRV}(\varepsilon)$, $R[\alpha \mapsto R(\beta)](\alpha) \neq R[\alpha \mapsto R(\beta)](\gamma)$ since $\beta \notin \mathrm{FRV}(\varepsilon)$, i.e. $R[\alpha \mapsto R(\beta)] : \mathrm{FRV}(\varepsilon) \hookrightarrow F$. □

Lemma 29 *(Force). If $\Pi \mid \Gamma \models e_1 \preceq e_2 : \forall \rho \notin \Delta.\, \tau^\varepsilon, \varepsilon'$ and $\sigma \in \Pi \backslash \Delta$, then $\Pi \mid \Gamma \models \mathsf{force}\, e_1 \preceq \mathsf{force}\, e_2 : \tau[\sigma/\rho], \varepsilon[\sigma/\rho] \cup \varepsilon'$.*

Proof. Take any $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$, and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume that $R : \mathrm{FRV}(\varepsilon[\sigma/\rho] \cup \varepsilon') \hookrightarrow \mathrm{dom}(w)$, and $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We need to show that $(k, \mathsf{force}\,\hat{e}_1, \mathsf{force}\,\hat{e}_2) \in \mathcal{E}\left[\!\left[\tau[\sigma/\rho]^{\varepsilon[\sigma/\rho] \cup \varepsilon'}\right]\!\right]^R w$, where $\hat{e}_i = \gamma_i(e_i)$. Instantiating the assumption with k, w, R, γ_1 and γ_2, we obtain $(k, \hat{e}_1, \hat{e}_2) \in \mathcal{E}\left[\!\left[\forall \rho \notin \Delta.\, \tau^{\varepsilon\,\varepsilon'}\right]\!\right]^R w$. We continue by unrolling the interpretation of expressions: we take any $j \leq k$, e_1', h_1, h_2, f_1, f_2 and $h_1^\dagger$, such that $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon_1 \cup \varepsilon'}\, w$, where $\varepsilon_1 = \varepsilon[\sigma/\rho]$, $\langle \mathsf{force}\,\hat{e}_1 \mid h_1 \cdot f_1 \rangle \to^j \langle e_1' \mid h_1^\dagger \rangle$ and $\mathrm{irr}\langle e_1' \mid h_1^\dagger \rangle$. This means that there exists $i \leq j$, $e_1^\ddagger$ and $h_1^\ddagger$, such that $\langle \hat{e}_1 \mid h_1 \cdot f_1 \rangle \to^i \langle e_1^\ddagger \mid h_1^\ddagger \rangle$, $\mathrm{irr}\langle e_1^\ddagger \mid h_1^\ddagger \rangle$ and $\langle \mathsf{force}\, e_1^\ddagger \mid h_1^\ddagger \rangle \to^{j-i} \langle e_1' \mid h_1^\dagger \rangle$. Moreover, Precondition Strengthening gives $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon'}\, w$.

Now we can instantiate the assumption about $\hat{e}_1$ and $\hat{e}_2$ being related with i, $e_1^\ddagger$, h_1, h_2, f_1, f_2 and $h_1^\ddagger$. We have already provided all the prerequisites, so we obtain a world $w' \sqsupseteq w$, $e_2^\ddagger$, h_2', g_1', g_2', such that

- $\mathrm{dom}(w') = \mathrm{dom}(w)$,
- $\langle \hat{e}_2 \mid h_2 \cdot f_2 \rangle \to^* \langle e_2^\ddagger \mid h_2' \cdot g_2' \cdot f_2 \rangle$,
- $h_1^\ddagger = h_1' \cdot g_1' \cdot f_1$,
- $(k - i, h_1, h_2, h_1', h_2') \in \mathbf{Q}^R_{\varepsilon'}\, w, w'$
- $(k - i, e_1^\ddagger, e_2^\ddagger) \in [\![\forall \rho \notin \Delta.\, \tau^\varepsilon]\!]^R\, w'$.

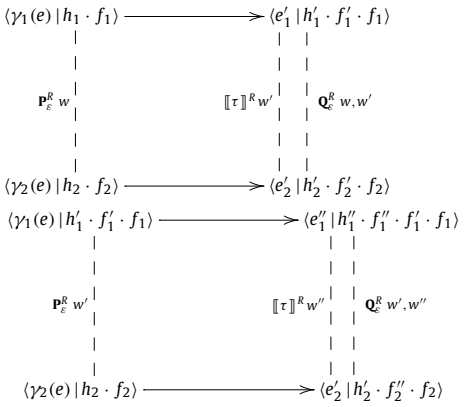


Fig. 8. Illustrated proof of the Idempotent Computation Theorem.

Recall now, that $\Delta = \mathrm{FRV}(\tau, \varepsilon) \setminus \{\rho\}$, since $\mathrm{wf}(\forall \rho \notin \Delta.\, \tau^\varepsilon, \varepsilon')$ holds, and so we know that $\sigma \notin \mathrm{FRV}(\tau, \varepsilon)$. From the definition of $[\![\forall \rho \notin \Delta.\, \tau^\varepsilon]\!]^R$ we know that for any future world w'' and interpretation of $\rho \in |w''|$ we pick, the expressions will be related. Pick w' as the world and $R(\sigma)$ as r, which is indeed in $|w'|$, then we have to prove that $R[\rho \mapsto R(\sigma)] : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w')$, which comes from Lemma 28 since $R : \mathrm{FRV}(\varepsilon[\sigma/\rho] \cup \varepsilon') \hookrightarrow \mathrm{dom}(w)$ and $\mathrm{dom}(w) = \mathrm{dom}(w')$. So we get that $(k - i, \tilde{e}_1, \tilde{e}_2) \in \mathcal{E}\, [\![\tau^\varepsilon]\!]^{R[\rho \mapsto R(\sigma)]}\, w'$, where $e_i^\ddagger = \mathrm{susp}\, \tilde{e}_i$. From this, by Lemmas 27 and 26, we get that $(k - i, \tilde{e}_1, \tilde{e}_2) \in \mathcal{E}\, [\![\tau[\sigma/\rho]^{\varepsilon[\sigma/\rho]}]\!]^R\, w'$. Finally, instantiate this assumption with $j - i - 1$, e'_1, h'_1, h'_2, $g'_1 \cdot f_1$, $g'_2 \cdot f_2$ and $h^\dagger_1$. Using Precondition Composition, we have $(k - i, h'_1, h'_2) \in \mathbf{P}^R_{\varepsilon_1 \cup \varepsilon'}\, w'$ and, since the other preconditions are trivial (the reduction takes one less step due to force-susp redex being already reduced), we get $w'' \sqsupseteq w'$, e'_2, h''_1, h''_2, g''_1 and g''_2 such that

- $\mathrm{dom}(w') = \mathrm{dom}(w'')$,
- $\langle \tilde{e}_2 \mid h'_2 \cdot g'_2 \cdot f_2 \rangle \to^* \langle e'_2 \mid h''_2 \cdot g''_2 \cdot g'_2 \cdot f_2 \rangle$,
- $h^\dagger_1 = h''_1 \cdot g''_1 \cdot g'_1 \cdot f_1$,
- $(k - j + 1, h'_1, h'_2, h''_1, h''_2) \in \mathbf{Q}^R_{\varepsilon_1}\, w', w''$,
- $(k - j + 1, e'_1, e'_2) \in [\![\tau[\sigma/\rho]]\!]^R\, w''$.

Now we finally are able to give the required witnesses: $w'' \sqsupseteq w$, e'_2, h''_1, h''_2, $g''_1 \cdot g'_1$ and $g''_2 \cdot g'_2$. The first three obligations are trivial, and the final one holds by downwards-closure of type interpretations. What is left to show is that $(k - j, h_1, h_2, h''_1, h''_2) \in \mathbf{Q}^R_{\varepsilon_1 \cup \varepsilon'}\, w, w''$, which holds by Postcondition Weakening and Postcondition Composition. □

7. Applications

We now show applications of our logical relations model: we verify four effect-based program transformations. These transformations are also considered in [14], but only for a language with ground store. To the best of our knowledge, the soundness of these effect-based transformations have not been proved before for a general ML-like language with higher-order store.

Theorem 30 *(Idempotent computation). A computation with disjoint read and write effects and no allocation effects is idempotent. More precisely, assume that we have*

$$\Pi \mid \Gamma \vdash e : \tau, \varepsilon$$

with $\mathrm{rds}\, \varepsilon \cap \mathrm{wrs}\, \varepsilon = \emptyset = \mathrm{als}\, \varepsilon$. *Then it holds that*

$$\Pi \mid \Gamma \models \texttt{let x} = e \texttt{ in let y} = e \texttt{ in (x, y)} \simeq \texttt{let x} = e \texttt{ in (x, x)} : \tau \times \tau, \varepsilon.$$

Proof. We just prove that the left hand side approximates the right hand side, the other way round proceeds similarly (see Fig. 8). Let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume that $R : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w)$ and that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We set

$$e_1 = \texttt{let x} = \gamma_1(e) \texttt{ in let y} = \gamma_1(e) \texttt{ in (x, y)}$$

and

$$e_2 = \texttt{let x} = \gamma_2(e) \texttt{ in } (\texttt{x}, \texttt{x})$$

and have to prove $(k, e_1, e_2) \in \mathcal{E}\,[\![\tau \times \tau^{\varepsilon}]\!]^R\, w$.

We proceed to unroll the definition of the relation on computations: let $j \le k$, $e_1''' \in \mathcal{E}$ and $h_1, h_2, f_1, f_2, g_1''' \in \mathcal{H}$ be arbitrary. Assume that $(k, h_1, h_2) \in \mathbf{P}_{\varepsilon}^R\, w$, that

$$\langle e_1 \,|\, h_1 \cdot f_1\rangle \xrightarrow{j} \langle e_1''' \,|\, g_1'''\rangle$$

and that $\mathrm{irr}\langle e_1''' \,|\, g_1'''\rangle$.

By the definition of the operational semantics there must be $0 \le i \le j$, $e_1' \in \mathcal{E}$ and $g_1' \in \mathcal{H}$ such that the above reduction can be split into

$$\begin{aligned}\langle e_1 \,|\, h_1 \cdot f_1\rangle &\xrightarrow{i} \langle\, \texttt{let x} = e_1' \texttt{ in let y} = \gamma_1(e) \texttt{ in } (\texttt{x}, \texttt{y}) \,|\, g_1'\rangle \\ &\xrightarrow{j-i} \langle e_1''' \,|\, g_1'''\rangle\end{aligned}$$

with

$$\langle \gamma_1(e) \,|\, h_1 \cdot f_1\rangle \xrightarrow{i} \langle e_1' \,|\, g_1'\rangle$$

and $\mathrm{irr}\langle e_1' \,|\, g_1'\rangle$.

From the Fundamental Lemma we get $(k, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\,[\![\tau^{\varepsilon}]\!]^R\, w$. Hence, we get $w' \sqsupseteq w$ with $\mathrm{dom}(w) = \mathrm{dom}(w')$ as well as $e_2' \in \mathcal{E}$ and $g_2', h_1', h_2', f_1', f_2' \in \mathcal{H}$ such that

$$\langle \gamma_2(e) \,|\, h_2 \cdot f_2\rangle \xrightarrow{*} \langle e_2' \,|\, g_2'\rangle,$$

$g_1' = h_1' \cdot f_1' \cdot f_1$, $g_2' = h_2' \cdot f_2' \cdot f_2$, $(k-i, h_1, h_2, h_1', h_2') \in \mathbf{Q}_{\varepsilon}^R\, w, w'$ and $(k-i, e_1', e_2') \in [\![\tau]\!]^R\, w'$. The latter implies $e_1' \in \mathcal{V}$ and so we must have $i < j$ as the alternative would invalidate $\mathrm{irr}\langle e_1''' \,|\, g_1'''\rangle$. Indeed, there must be $0 \le i' \le j - i - 1$, $e_1'' \in \mathcal{E}$ and $g_1'' \in \mathcal{H}$ such that we can split the last $j - i$ steps further

$$\begin{aligned}&\langle \texttt{let x} = e_1' \texttt{ in let y} = \gamma_1(e) \texttt{ in } (\texttt{x}, \texttt{y}) \,|\, g_1'\rangle \\ &\quad\to \langle \texttt{let y} = \gamma_1(e) \texttt{ in } (e_1', \texttt{y}) \,|\, g_1'\rangle \\ &\quad\xrightarrow{i'} \langle \texttt{let y} = e_1'' \texttt{ in } (e_1', \texttt{y}) \,|\, g_1''\rangle \\ &\quad\xrightarrow{j-i-1-i'} \langle e_1''' \,|\, g_1'''\rangle\end{aligned}$$

with

$$\langle \gamma_1(e) \,|\, g_1'\rangle \xrightarrow{i'} \langle e_1'' \,|\, g_1''\rangle$$

and $\mathrm{irr}\langle e_1'' \,|\, g_1''\rangle$.

Now for something odd: we reset the right hand side to the initial state. More precisely, we argue that $(k-i-1, h_1', h_2) \in \mathbf{P}_{\varepsilon}^R\, w'$; this is the crux of the entire proof. Notice initially, that not only is it the case that $\mathrm{dom}(w) = \mathrm{dom}(w')$, we also have $\forall r \in \mathrm{dom}(w).\, w(r) = w'(r)$ since $\mathrm{als}\,\varepsilon = \emptyset$; in particular we get $\mathrm{dom}_2(w) = \mathrm{dom}_2(w')$. Combining now the facts

$$\begin{aligned}&\forall \rho \in \mathrm{rds}\,\varepsilon.\, \forall (l_1, l_2, \mu) \in w(R(\rho)).\\ &\quad k > 0 \Rightarrow (k-1, h_1(l_1), h_2(l_2)) \in (\iota\,\mu)(w)\end{aligned}$$

and

$$\begin{aligned}&\forall l_1 \in \mathrm{dom}(h_1).\, h_1(l_1) \neq h_1'(l_1) \Rightarrow\\ &\quad \exists \rho \in \mathrm{wrs}\,\varepsilon.\, \exists (m_1, l_2, \mu) \in w(R(\rho)).\, l_1 = m_1,\end{aligned}$$

with $\mathrm{rds}\,\varepsilon \cap \mathrm{wrs}\,\varepsilon = \emptyset$ and the injectivity of R on $\mathrm{FRV}(\varepsilon)$ buys us

$$\begin{aligned}&\forall \rho \in \mathrm{rds}\,\varepsilon.\, \forall (l_1, l_2, \mu) \in w'(R(\rho)).\\ &\quad k - i - 1 > 0 \Rightarrow (k-i-2, h_1'(l_1), h_2(l_2)) \in (\iota\,\mu)(w').\end{aligned}$$

Loosely speaking, the locations we are permitted to read hold values of the correct type from the beginning and were not changed by the computation seen so far.

We proceed to use the fact that $(k-i-1, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\,[\![\tau^{\varepsilon}]\!]^R\, w'$ by a second application of the Fundamental Lemma instantiated with $k-i-1$ and w'. Using the fact that $(k-i-1, h_1', h_2) \in \mathbf{P}_{\varepsilon}^R\, w'$, this yields $w'' \sqsupseteq w'$ with $\mathrm{dom}(w') = \mathrm{dom}(w'')$ as well as $e_2'' \in \mathcal{E}$ and $g_2'', h_1'', h_2'', f_1'', f_2'' \in \mathcal{H}$ such that

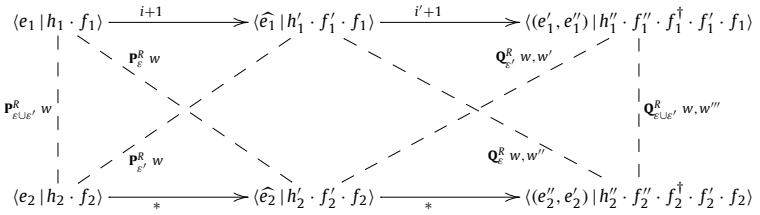


Fig. 9. Illustrated proof of the Commuting Computations Theorem. In the diagram, we use $\widehat{e_1} = \texttt{let y} = \gamma_1(e')\ \texttt{in}\ (e_1', \texttt{y})$ and $\widehat{e_2} = \texttt{let x} = \gamma_2(e)\ \texttt{in}\ (\texttt{x}, e_2')$ as abbreviations.

$$\langle \gamma_2(e) \mid h_2 \cdot f_2 \rangle \xrightarrow{*} \langle e_2'' \mid g_2'' \rangle,$$

$g_1'' = h_1'' \cdot f_1'' \cdot f_1' \cdot f_1$, $g_2'' = h_2'' \cdot f_2'' \cdot f_2$, $(k-i-1-i', h_1', h_2, h_1'', h_2'') \in \mathbf{Q}^R_\varepsilon\ w', w''$ and $(k-i-1-i', e_1'', e_2'') \in [\![\tau]\!]^R\ w''$. The latter implies $e_1'' \in \mathcal{V}$ and so we must have $i' < j-i-1$ since an equality would conflict with $\mathrm{irr}\langle e_1''' \mid g_1''' \rangle$. Even more precisely, we must have $j-i-1-i' = 1$ and the final step in the entire reduction $\langle e_1 \mid h_1 \cdot f_1 \rangle \xrightarrow{j} \langle e_1''' \mid g_1''' \rangle$ must be

$$\langle \texttt{let y} = e_1''\ \texttt{in}\ (e_1', \texttt{y}) \mid g_1'' \rangle \to \langle (e_1', e_1'') \mid g_1'' \rangle,$$

in particular $e_1''' = (e_1', e_1'')$ and $g_1''' = g_1''$.

We immediately get

$$\langle e_2 \mid h_2 \cdot f_2 \rangle \xrightarrow{*} \langle (e_2'', e_2'') \mid g_2'' \rangle.$$

In the proof, it now remains to prove $(k-j, h_1, h_2, h_1'', h_2'') \in \mathbf{Q}^R_\varepsilon\ w, w''$ and that $(k-j, (e_1', e_1''), (e_2'', e_2'')) \in [\![\tau \times \tau]\!]^R\ w''$. Notice initially, that by the determinism of the operational semantics and the fact that $e_2', e_2'' \in \mathcal{V}$ we have $e_2' = e_2''$ and $g_2' = g_2''$. Since $\mathrm{dom}_2(w') = \mathrm{dom}_2(w'')$ by $\mathrm{als}\,\varepsilon = \emptyset$ we furthermore get $h_2' = h_2''$ and $f_2' = f_2''$. Also recall $k-j = k-i-i'-2$.

On the first obligation, we take $l_1 \in \mathrm{dom}(h_1)$ and assume that $h_1(l_1) \neq h_1''(l_1)$. If also $h_1'(l_1) \neq h_1''(l_1)$ then the desired follows from $(k-i-1-i', h_1', h_2, h_1'', h_2'') \in \mathbf{Q}^R_\varepsilon\ w', w''$. Otherwise, we must have $h_1(l_1) \neq h_1'(l_1) = h_1''(l_1)$, and $(k-i, h_1, h_2, h_1', h_2') \in \mathbf{Q}^R_\varepsilon\ w, w'$ paves the way. And the second obligation is met by recalling $(k-i, e_1', e_2') \in [\![\tau]\!]^R\ w'$ and $(k-i-1-i', e_1'', e_2'') \in [\![\tau]\!]^R\ w''$. □

Theorem 31 *(Commuting computations). Two computations commute if neither reads a region that the other writes, and there is no region they both write. More precisely, assume that we have*

$$\Pi \mid \Gamma \vdash e : \tau, \varepsilon, \quad \Pi \mid \Gamma \vdash e' : \tau', \varepsilon'$$

with $\mathrm{rds}\,\varepsilon \cap \mathrm{wrs}\,\varepsilon' = \mathrm{rds}\,\varepsilon' \cap \mathrm{wrs}\,\varepsilon = \mathrm{wrs}\,\varepsilon \cap \mathrm{wrs}\,\varepsilon' = \emptyset$.

$$\texttt{let x} = e\ \texttt{in let y} = e'\ \texttt{in}\ (\texttt{x}, \texttt{y}) \simeq \texttt{let y} = e'\ \texttt{in let x} = e\ \texttt{in}\ (\texttt{x}, \texttt{y})$$

in context $\Pi \mid \Gamma$ *at type* $\tau \times \tau'$, $\varepsilon \cup \varepsilon'$.

Proof. Again we give the details only one way: that the left hand side approximates the right hand side (see Fig. 9). Let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume that $R : \mathrm{FRV}(\varepsilon \cup \varepsilon') \hookrightarrow \mathrm{dom}(w)$ and that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\ w$. Set

$$e_1 = \texttt{let x} = \gamma_1(e)\ \texttt{in let y} = \gamma_1(e')\ \texttt{in}\ (\texttt{x}, \texttt{y})$$

and

$$e_2 = \texttt{let y} = \gamma_2(e')\ \texttt{in let x} = \gamma_2(e)\ \texttt{in}\ (\texttt{x}, \texttt{y});$$

we have to prove $(k, e_1, e_2) \in \mathcal{E}\left[\!\left[\tau \times \tau'^{\varepsilon\cup\varepsilon}\right]\!\right]^R w$.

We proceed to unroll the definition of the relation on computations: let $j \leq k$, $e_1''' \in \mathcal{E}$ and $h_1, h_2, f_1, f_2, g_1''' \in \mathcal{H}$ be arbitrary. Assume that $(k, h_1, h_2) \in \mathbf{P}^R_{\varepsilon\cup\varepsilon'}\ w$, that

$$\langle e_1 \mid h_1 \cdot f_1 \rangle \xrightarrow{j} \langle e_1''' \mid g_1''' \rangle$$

and that $\mathrm{irr}\langle e_1''' \mid g_1''' \rangle$.

By the definition of the operational semantics there must be $0 \leq i \leq j$, $e_1' \in \mathcal{E}$ and $g_1' \in \mathcal{H}$ such that the above reduction can be split into

$$\langle e_1 \mid h_1 \cdot f_1\rangle \xrightarrow{i} \langle\, \mathtt{let\ x} = e_1' \mathtt{\ in\ let\ y} = \gamma_1(e') \mathtt{\ in\ (x, y)} \mid g_1'\rangle \xrightarrow{j-i} \langle e_1''' \mid g_1'''\rangle$$

with

$$\langle \gamma_1(e) \mid h_1 \cdot f_1\rangle \xrightarrow{i} \langle e_1' \mid g_1'\rangle$$

and $\mathrm{irr}\langle e_1' \mid g_1'\rangle$.

From the Fundamental Lemma we get $(k, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\,[\![\tau^{\varepsilon}]\!]^R\, w$. By Precondition Strengthening we get the two following facts: that $e_1' \in \mathcal{V}$ and that for $l_1 \in \mathrm{dom}(h_1)$ such that there is $r \in R(\mathrm{rds}\,\varepsilon')$ and $(m_1, l_2, \mu) \in w(r)$ with $m_1 = l_1$ we have $g_1'(l_1) = h_1(l_1)$. The latter is a consequence of $\mathrm{wrs}\,\varepsilon \cap \mathrm{rds}\,\varepsilon' = \emptyset$ and the injectivity of R on $\mathrm{FRV}(\varepsilon \cup \varepsilon')$.

We must have $i < j$ as the alternative would invalidate $\mathrm{irr}\langle e_1''' \mid g_1'''\rangle$. Indeed, there must be $0 \le i' \le j - i - 1$, $e_1'' \in \mathcal{E}$ and $g_1'' \in \mathcal{H}$ such that we can split the last $j - i$ steps further into

$$\begin{aligned}&\langle \mathtt{let\ x} = e_1' \mathtt{\ in\ let\ y} = \gamma_1(e') \mathtt{\ in\ (x, y)} \mid g_1'\rangle\\ &\quad\to \langle \mathtt{let\ y} = \gamma_1(e') \mathtt{\ in\ } (e_1', \mathtt{y}) \mid g_1'\rangle\\ &\quad\xrightarrow{i'} \langle \mathtt{let\ y} = e_1'' \mathtt{\ in\ } (e_1', \mathtt{y}) \mid g_1''\rangle\\ &\quad\xrightarrow{j-i-1-i'} \langle e_1''' \mid g_1'''\rangle\end{aligned}$$

with

$$\langle \gamma_1(e') \mid g_1'\rangle \xrightarrow{i'} \langle e_1'' \mid g_1''\rangle$$

and $\mathrm{irr}\langle e_1'' \mid g_1''\rangle$.

Much as in the previous proof, we now ditch our right hand side progress, but unlike that, we also lose our new future world. Notice first that the Fundamental Lemma gives us $(k - i - 1, \gamma_1(e'), \gamma_2(e')) \in \mathcal{E}\left[\!\left[\tau'^{\varepsilon'}\right]\!\right]^R w$; we would like to apply that. Write $g_1' = h_1^\dagger \cdot h_1^\circ$ for $h_1^\dagger, h_1^\circ \in \mathcal{H}$ with $\mathrm{dom}(h_1^\dagger) = \mathrm{dom}_1(w)$; the crucial observation now is that we have $(k - i - 1, h_1^\dagger, h_2) \in \mathbf{P}^R_{\varepsilon'} w$. Hence we get $w' \sqsupseteq w$ with $\mathrm{dom}(w) = \mathrm{dom}(w')$ as well as $e_2' \in \mathcal{E}$ and $g_2', h_1'', h_2', f_1'', f_2' \in \mathcal{H}$ such that

$$\langle \gamma_2(e') \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle e_2' \mid g_2'\rangle,$$

$g_1'' = h_1'' \cdot f_1'' \cdot h_1^\circ$, $g_2' = h_2' \cdot f_2' \cdot f_2$, $(k - i - 1 - i', h_1^\dagger, h_2, h_1'', h_2') \in \mathbf{Q}^R_{\varepsilon'} w, w'$ and $(k - i - 1 - i', e_1'', e_2') \in [\![\tau']\!]^R\, w'$.

Much as we have argued before, we now know that $e_1'' \in \mathcal{V}$, hence $j - i - 1 - i' = 1$ and the entire left hand side reduction must look like

$$\langle e_1 \mid h_1 \cdot f_1\rangle \xrightarrow{j} \langle (e_1', e_1'') \mid g_1''\rangle,$$

in particular, $e_1''' = (e_1', e_1'')$ and $g_1''' = g_1''$.

We have now, in some sense, considered the situation from the point of view of e'; it is time to turn the tables. There are $h_2^\dagger$, $f_2^\dagger$ with $\mathrm{dom}(h_2^\dagger) = \mathrm{dom}_2(w)$ such that $h_2' = h_2^\dagger \cdot f_2^\dagger$. So we have $(k, h_1, h_2^\dagger) \in \mathbf{P}^R_{\varepsilon}\, w$ by arguments as above; in particular we apply $\mathrm{wrs}\,\varepsilon' \cap \mathrm{rds}\,\varepsilon = \emptyset$. Now, we still have $(k, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\,[\![\tau^{\varepsilon}]\!]^R\, w$ and so there is $w'' \sqsupseteq w$ with $\mathrm{dom}(w) = \mathrm{dom}(w'')$ as well as $e_2'' \in \mathcal{E}$ and $g_2'', h_1', h_2'', f_1', f_2'' \in \mathcal{H}$ such that

$$\langle \gamma_2(e) \mid h_2^\dagger \cdot f_2^\dagger \cdot f_2' \cdot f_2\rangle \xrightarrow{*} \langle e_2'' \mid g_2''\rangle,$$

$g_1' = h_1' \cdot f_1' \cdot f_1$, $g_2'' = h_2'' \cdot f_2'' \cdot f_2^\dagger \cdot f_2' \cdot f_2$, $(k - i, h_1, h_2^\dagger, h_1', h_2'') \in \mathbf{Q}^R_{\varepsilon}\, w, w''$ and $(k - i, e_1', e_2'') \in [\![\tau]\!]^R\, w''$. Write $h_1' = h_1^\dagger \cdot f_1^\dagger$ for $f_1^\dagger \in \mathcal{H}$, in particular $g_1' = h_1^\dagger \cdot f_1^\dagger \cdot f_1' \cdot f_1$ and $g_1'' = h_1'' \cdot f_1'' \cdot f_1^\dagger \cdot f_1' \cdot f_1$; this is notation we need soon. Observe first, though, that we have the right hand side reduction

$$\langle e_2 \mid h_2 \cdot f_2\rangle \xrightarrow{*} \langle (e_2', e_2'') \mid g_2''\rangle.$$

We now build a world w''' with $\mathrm{dom}(w''') = \mathrm{dom}(w)$ and with both $w''' \sqsupseteq w'$ and $w''' \sqsupseteq w''$, i.e., a common future world. The natural choice is for the dead regions of w''' to be the dead regions of both w' and w''. For $r \in \mathrm{dom}(w)$ we set $w'''(r) = w'(r) \cup w''(r)$, but we must take care not to wreck the partial bijections nor their mutual disjointness. Let $r, s \in \mathrm{dom}(w)$ and take $(l_1', l_2', \mu') \in w'(r) \setminus w(r)$ and $(l_1'', l_2'', \mu'') \in w''(s) \setminus w(s)$; it will suffice to show $l_1' \neq l_1''$ and $l_2' \neq l_2''$. Now, we know that $l_1' \in \mathrm{dom}_1(w') = \mathrm{dom}(h_1'')$; in particular we have $l_1' \notin \mathrm{dom}(f_1^\dagger)$. Also $l_1' \notin \mathrm{dom}_1(w) = \mathrm{dom}(h_1^\dagger)$. But $l_1'' \in \mathrm{dom}_1(w'') = \mathrm{dom}(h_1')$ and since $h_1' = h_1^\dagger \cdot f_1^\dagger$ we must have $l_1' \neq l_1''$. Proving $l_2' \neq l_2''$ proceeds similarly.

It shall now suffice to show

$$(k-j, h_1, h_2, h_1'' \cdot f_1^\dagger, h_2'' \cdot f_2^\dagger) \in \mathbf{Q}^R_{\varepsilon \cup \varepsilon'}\, w, w'''$$

as the remaining obligations are discharged already. We have

$$\begin{aligned}\mathrm{dom}_1(w''') &= \mathrm{dom}_1(w') \cup \mathrm{dom}_1(w'') \\ &= \mathrm{dom}(h_1'') \cup \mathrm{dom}(h_1^\dagger \cdot f_1^\dagger) \\ &= \mathrm{dom}(h_1'' \cdot f_1^\dagger)\end{aligned}$$

and get $\mathrm{dom}_2(w''') = \mathrm{dom}(h_2'' \cdot f_2^\dagger)$ similarly. So take $l_1 \in \mathrm{dom}(h_1)$ and assume that $h_1(l_1) \neq h_1''(l_1)$. If $h_1^\dagger(l_1) \neq h_1''(l_1)$ we get $\rho \in \mathrm{wrs}\,\varepsilon'$ and $(m_1, l_2, \mu) \in w(R(\rho))$ with $l_1 = m_1$ and $(k-j, h_1''(l_1), h_2'(l_2)) \in (\iota\,\mu)(w')$. And since $\mathrm{wrs}\,\varepsilon \cap \mathrm{wrs}\,\varepsilon = \emptyset$ we know that $h_2''(l_2) = h_2'(l_1)$. If, on the other hand, $h_1^\dagger(l_1) = h_1''(l_1)$ then we must have $h_1(l_1) \neq h_1^\dagger(l_1)$ and we get $\rho \in \mathrm{wrs}\,\varepsilon$ and $(m_1, l_2, \mu) \in w(R(\rho))$ with $l_1 = m_1$ and $(k-i-1, h_1^\dagger(l_1), h_2''(l_2)) \in (\iota\,\mu)(w'')$.

It remains to consider allocation. So take $r \in \mathrm{dom}(w)$ and $(l_1, l_2, \mu) \in w'''(r) \setminus w(r)$. By the construction of w''' we get $(l_1, l_2, \mu) \in w'(r) \setminus w(r)$ or $(l_1, l_2, \mu) \in w'(r) \setminus w(r)$; in both cases we proceed similarly, so assume the former holds. Then we know $r \in R(\mathrm{als}\,\varepsilon')$ and that $(k-j, h_1''(l_1), h_2'(l_2)) \in (\iota\,\mu)(w')$. But as $l_2 \notin \mathrm{dom}_2(w)$ we must have $l_2 \in \mathrm{dom}(f_2^\dagger)$ and so $(h_2'' \cdot f_2^\dagger)(l_2) = f_2^\dagger(l_2) = h_2'(l_2)$ and we are done. □

In the remaining two applications we can prove only approximation, not equivalence, because of the possibility of non-termination.

Theorem 32 *(Neutral computation). A computation that performs no writes and has return type unit approximates the trivial computation. More precisely, having*

$$\Pi \mid \Gamma \vdash e : 1,\ \varepsilon$$

with $\mathrm{wrs}\,\varepsilon = \emptyset$ *implies*

$$\Pi \mid \Gamma \models e \preceq () : 1, \varepsilon.$$

Proof. Let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume $R : \mathrm{FRV}(\varepsilon) \hookrightarrow \mathrm{dom}(w)$ and that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We now must show that

$$(k, \gamma_1(e), ()) \in \mathcal{E}\,[\![1^\varepsilon]\!]^R\, w.$$

We proceed to unroll the definition of computations: let $j \leq k$, $h_1, h_2, f_1, f_2 \in \mathcal{H}$, $e_1' \in \mathcal{E}$ and $h_1' \in \mathcal{H}$ be arbitrary. Assume $(k, h_1, h_2) \in \mathbf{P}^R_\varepsilon\, w$, that

$$\langle \gamma_1(e) \mid h_1 \cdot f_1 \rangle \xrightarrow{j} \langle e_1' \mid h_1' \rangle$$

and that $\mathrm{irr}\langle e_1' \mid h_1' \rangle$.

By an application of the Fundamental Lemma, we get $(k, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\,[\![1^\varepsilon]\!]^R\, w$. This gives us that $e_1' = ()$, and, since there are no write effects, that $h_1' = h_1 \cdot f_1' \cdot f_1$. The right hand side of the computation proper terminates in zero steps, i.e., we have

$$\langle () \mid h_2 \cdot f_2 \rangle \xrightarrow{0} \langle () \mid h_2 \cdot f_2 \rangle.$$

It is straightforward that $(k, h_1, h_2, h_1, h_2) \in \mathbf{Q}^R_\varepsilon\, w, w$ and $(k, (), ()) \in [\![1]\!]^R\, w$, so the proof is done. □

Theorem 33 *(Pure lambda hoist). A pure computation evaluated as part of a function can, up to approximation, be evaluated once and the result cached. More precisely, having*

$$\Pi \mid \Gamma \vdash e : \tau_1, \emptyset, \quad \Pi \mid \Gamma, y : \tau_2, x : \tau_1 \vdash e' : \tau_3, \varepsilon$$

gives us

$$\Pi \mid \Gamma \models \mathtt{let\ x} = e\ \mathtt{in}\ \lambda y.e' \preceq \lambda y.\ \mathtt{let\ x} = e\ \mathtt{in}\ e' : \tau_2 \to^\varepsilon \tau_3, \emptyset$$

Proof. Let $k \in \mathbb{N}$, $w \in \mathbf{W}$, $R : \Pi \to |w|$ and $\gamma_1, \gamma_2 \in \mathrm{Subst}(\Gamma)$ be arbitrary. Assume that $(k, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R\, w$. We have to prove

$$(k, \mathtt{let\,x} = \gamma_1(e)\,\mathtt{in}\,\lambda\mathtt{y}.\,\gamma_1(e'),$$
$$\lambda\mathtt{y}.\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e')) \in \mathcal{E}\left[\!\left[\tau_2 \to^{\varepsilon} \tau_3{}^{\emptyset}\right]\!\right]^R w.$$

We proceed to unroll the definition of computations: let $j \le k$, $h_1, h_2, f_1, f_2 \in \mathcal{H}$, $e_1'' \in \mathcal{E}$ and $h_1^{\ddagger} \in \mathcal{H}$ be arbitrary. Assume $(k, h_1, h_2) \in \mathbf{P}_{\emptyset}^R\, w$, that

$$\langle \mathtt{let\,x} = \gamma_1(e)\,\mathtt{in}\,\lambda\mathtt{y}.\,\gamma_1(e') \mid h_1 \cdot f_1\rangle \xrightarrow{j} \langle e_1'' \mid h_1^{\ddagger}\rangle$$

and $\mathrm{irr}\langle e_1'' \mid h_1''\rangle$. By the operational semantics, there must be $0 \le i \le j$, $e_1' \in \mathcal{E}$ and $h_1^{\dagger} \in \mathcal{H}$, such that

$$\langle \gamma_1(e) \mid h_1 \cdot f_1\rangle \xrightarrow{i} \langle e_1' \mid h_1^{\dagger}\rangle$$

and

$$\langle \mathtt{let\,x} = e_1'\,\mathtt{in}\,\lambda\mathtt{y}.\,\gamma_1(e') \mid h_1^{\dagger}\rangle \xrightarrow{j-i} \langle e_1'' \mid h_1^{\ddagger}\rangle$$

and $\mathrm{irr}\langle e_1' \mid h_1'\rangle$ holds. The Fundamental Lemma gives us $(k, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\left[\!\left[\tau_1{}^{\emptyset}\right]\!\right]^R w$; it is an easy consequence that we have $e_1' \in \mathcal{V}$ and that there is $f_1' \in \mathcal{H}$ such that $h_1^{\dagger} = h_1 \cdot f_1' \cdot f_1$. In particular, only one step remains on the left hand side

$$\langle \mathtt{let\,x} = e_1'\,\mathtt{in}\,\lambda\mathtt{y}.\,\gamma_1(e') \mid h_1^{\dagger}\rangle$$
$$\to \langle \lambda\mathtt{y}.\,\gamma_1(e')[e_1'/\mathtt{x}] \mid h_1^{\dagger}\rangle = \langle e_1'' \mid h_1^{\ddagger}\rangle$$

The right hand side is a value in itself, so we get the following zero-step reduction

$$\langle \lambda\mathtt{y}.\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e') \mid h_2 \cdot f_2\rangle$$
$$\xrightarrow{0} \langle \lambda\mathtt{y}.\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e') \mid h_2 \cdot f_2\rangle.$$

Using $(k-j, h_1, h_2, h_1, h_2) \in \mathbf{Q}_{\emptyset}^R\, w, w$, we have left to prove that

$$(k-j, \lambda\mathtt{y}.\,\gamma_1(e')[e_1'/\mathtt{x}], \lambda\mathtt{y}.\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e')) \in [\![\tau_2 \to^{\varepsilon} \tau_3]\!]^R\, w.$$

Notice first, that since the lambda is just syntactic sugar for a non-recursive fixed point, we have

$$\lambda\mathtt{y}.\,\gamma_1(e')[e_1'/\mathtt{x}] = \mathtt{fix}\,f(\mathtt{y}).\,\gamma_1(e')[e_1'/\mathtt{x}]$$

and similarly

$$\lambda\mathtt{y}.\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e')$$
$$= \mathtt{fix}\,f(\mathtt{y}).\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e').$$

So we proceed, according to the interpretation of function types: pick $w' \sqsupseteq w$ with $R(\mathrm{FRV}(\varepsilon)) \hookrightarrow \mathrm{dom}(w')$, $k' \le k-j$, $v_1, v_2 \in \mathcal{V}$ with $(k', v_1, v_2) \in [\![\tau_2]\!]^R\, w'$. We now have to show

$$\big(k',\ (\mathtt{fix}\,f(\mathtt{y}).\,\gamma_1(e')[e_1'/\mathtt{x}])\,v_1,$$
$$(\mathtt{fix}\,f(\mathtt{y}).\,\mathtt{let\,x} = \gamma_2(e)\,\mathtt{in}\,\gamma_2(e'))\,v_2\big) \in \mathcal{E}\left[\!\left[\tau_3{}^{\varepsilon}\right]\!\right]^R\, w'.$$

We proceed, once more, to unroll the definition of the relation on computations: let $j' \le k'$, $h_1'', h_2'', f_1'', f_2'' \in \mathcal{H}$, $e_1^{\circ} \in \mathcal{E}$ and $h_1^{\circ} \in \mathcal{H}$ be arbitrary. Assume $(k', h_1'', h_2'') \in \mathbf{P}_{\varepsilon}^R\, w'$, that

$$\langle (\mathtt{fix}\,f(\mathtt{y}).\,\gamma_1(e')[e_1'/\mathtt{x}])\,v_1 \mid h_1'' \cdot f_1''\rangle \xrightarrow{j'} \langle e_1^{\circ} \mid h_1^{\circ}\rangle$$

and $\mathrm{irr}\langle e_1^{\circ} \mid h_1^{\circ}\rangle$.

Intuitively, we now temporarily forget about the changes made between w and w', and interpret the computations $\gamma_1(e)$ and $\gamma_2(e)$. Afterwards, we retrace the changes from w to w' in the resulting world, since e cannot interfere with them, and build the resulting world w''. More precisely, we build a brand new world $w^{\dagger} \in \mathbf{W}$ that has the same dead and live regions as w' but such that for any $r \in \mathrm{dom}(w^{\dagger})$ we have $w^{\dagger}(r) = w(r)$ if $r \in \mathrm{dom}(w)$ and $w^{\dagger}(r) = \emptyset$ if $r \notin |w|$. Note that $w^{\dagger} \sqsupseteq w$ and that $\mathrm{dom}_1(w^{\dagger}) \subseteq \mathrm{dom}_1(w)$ as well as $\mathrm{dom}_2(w^{\dagger}) \subseteq \mathrm{dom}_2(w')$. The latter two are the crux: there are $g_1 \subseteq h_1$ and $g_2'' \subseteq h_2''$ with $(k, g_1, g_2'') \in \mathbf{P}_{\emptyset}^R\, w^{\dagger}$. The Fundamental Lemma yields $(k, \gamma_1(e), \gamma_2(e)) \in \mathcal{E}\left[\!\left[\tau_1{}^{\emptyset}\right]\!\right]^R\, w^{\dagger}$ and so we get $w^{\ddagger} \sqsupseteq w^{\dagger}$ with $\mathrm{dom}(w^{\ddagger}) = \mathrm{dom}(w^{\dagger})$ and $e_2' \in \mathcal{E}$ with

$$\langle \gamma_2(e) \mid h_2'' \cdot f_2''\rangle \xrightarrow{*} \langle e_2' \mid h_2'' \cdot f_2''' \cdot f_2''\rangle$$

for some $f_2''' \in \mathcal{H}$ as well as $(k-i, e_1', e_2') \in [\![\tau_1]\!]^R w^\ddagger$. Now observe that the only difference between $w^\dagger$ and $w^\ddagger$ is a possible addition of dead regions as there are no allocation effects. And so we build the world $w'' \in \mathbf{W}$ that is just w' with the dead regions extended to hold also the dead regions of $w^\ddagger$. In particular, we get $w'' \sqsupseteq w^\ddagger$ as well as $w'' \sqsupseteq w'$.

Returning to the computation at hand, the left hand side reduction must go like

$$\langle (\mathtt{fix}\, f(\mathtt{y}).\, \gamma_1(e')[e_1'/\mathtt{x}])\, v_1 \mid h_1'' \cdot f_2'' \rangle \to$$
$$\langle \gamma_1(e')[e_1'/\mathtt{x}][v_1/\mathtt{y}] \mid h_1'' \cdot f_2'' \rangle \overset{j'-1}{\to} \langle e_1^\circ \mid h_1^\circ \rangle$$

and we can take a few steps on the right hand side as well

$$\begin{aligned}
&\langle (\mathtt{fix}\, f(\mathtt{y}).\, \mathtt{let}\, \mathtt{x} = \gamma_2(e)\, \mathtt{in}\, \gamma_2(e'))\, v_2 \mid h_2'' \cdot f_2'' \rangle \\
&\to \langle \mathtt{let}\, \mathtt{x} = \gamma_2(e)\, \mathtt{in}\, \gamma_2(e')[v_2/\mathtt{y}] \mid h_2'' \cdot f_2'' \rangle \\
&\overset{*}{\to} \langle \mathtt{let}\, \mathtt{x} = e_2'\, \mathtt{in}\, \gamma_2(e')[v_2/\mathtt{y}] \mid h_2'' \cdot f_2''' \cdot f_2'' \rangle \\
&\to \langle \gamma_2(e')[v_2/\mathtt{y}][e_2'/\mathtt{x}] \mid h_2'' \cdot f_2''' \cdot f_2'' \rangle.
\end{aligned}$$

All that remains now is, in essence, to apply the Fundamental Lemma to $\Pi \mid \Gamma, \mathtt{y} : \tau_2, \mathtt{x} : \tau_1 \vdash e' : \tau_3, \varepsilon$: we note that $R(\mathrm{FRV}(\varepsilon)) \hookrightarrow \mathrm{dom}(w') = \mathrm{dom}(w'')$, that $(k'-1, \gamma_1, \gamma_2) \in [\![\Gamma]\!]^R w''$, that $(k'-1, v_1, v_2) \in [\![\tau_2]\!]^R w''$ and that $(k'-1, e_1', e_2') \in [\![\tau_1]\!]^R w''$. This gives us that

$$(k'-1, \gamma_1(e')[v_1/\mathtt{y}][e_1'/\mathtt{x}], \gamma_2(e')[v_2/\mathtt{y}][e_2'/\mathtt{x}]) \in \mathcal{E}\, [\![{\tau_3}^\varepsilon]\!]^R\, w''.$$

Finally, we can use this knowledge together with

$$\langle \gamma_1(e')[e_1'/\mathtt{x}][v_1/\mathtt{y}] \mid h_1'' \cdot f_2'' \rangle \overset{j'-1}{\to} \langle e_1^\circ \mid h_1^\circ \rangle,$$

where the appropriate precondition holds by Precondition Composition, since the only difference between w' and w'' are the extra dead regions, to finish the proof. □

8. Discussion

8.1. Work by Benton et al.

An important point of reference for our work is the relational model by Benton et al. [14] of an effect system for a higher-order language with dynamic allocation and ground store, i.e., only integers in the heap. Indeed, apart from our extension to higher-order store and region polymorphism, the type systems and examples considered are roughly the same.

Having said so, we remark that our take on the issue of masking is novel; in particular it is different from that of Benton et al. Their approach does not scale easily, if at all, to the higher-order store setting: the pivot is the *Masking Lemma* [14, Lemma 3], stating that the interpretation of both types and computations in a world are preserved up to equality under masking, *provided* that the region masked out is not, syntactically, in the type respectively in the computation. Combined with ground store, this makes short work of soundness of the masking rule.

The Masking Lemma, however, does not scale easily to a higher-order store setting. Consider the computation from the introduction: the returned function has latent effect $\{wr_\sigma\}$ and correspondingly writes to location 1 in region σ. But the values stored at location 1 must be of type $\mathsf{ref}_\rho\mathsf{int}$, and that depends on the type int associated with location 0 in region ρ. In other words, the interpretation of a type may depend on regions that do not occur syntactically in the type; this is the antithesis to the Masking Lemma.

As described above, we take a different approach: that interpretations of types should grow (or at least not shrink) under any application of masking, and only when we actually perform reads, writes and allocations do we require that the regions in question are still live. This also means that we can get by without a *silent region* [14, Sections 5 and 6]. This is a designated region of the world that tracks inaccessible parts of the heap in an untyped way; in [14] it is necessary for the Masking Lemma to hold for computations.

Our different approach to masking is actually in some cases more restrictive than Benton et al.'s: our model permits *any* action on locations that have been masked out, including garbage collection or ownership transfer. The locations have left the world and we make no further assumptions on them whatsoever. By contrast, locations in a silent region are still in the world and computations may assume that they remain allocated, even if the stored value cannot be changed. Indeed, computations may actually access such locations in extensionally invisible ways: we could, say, read a location in the silent region as long as we make no use of the read value. Thus, more computations can be treated as pure if one uses the silent region. However, it is not clear how this approach could be made to work in the higher-order case.

As mentioned earlier, our approach to masking is similar to the approach used by Ahmed to model region deallocation [16].

Benton et al.: higher order store In more recent work, Benton et al. [15] have given a relational model for a language with higher order store. The language has no dynamic allocation though, nor is there any masking rule.

Unlike our approach, however, they work with a denotational semantics of the language; in particular they prove existence of the logical relation by solving a non-trivial mixed-variance domain equation, extending the standard technique by Pitts [24] to do so. One advantage is transitivity of their logical relation; that is something we do not get. This is a general issue with step-indexed models observed first by Ahmed [25]; there are fixes, but a more proper take is to reason in a logic suited for step-indexing as done, e.g., by Dreyer et al. [26]. We are convinced, however, that using the techniques developed by Støvring and two of the authors [19], we could also define a model based on a denotational semantics of the programming language, hence achieving transitivity.

Preservation of all store relations Throughout their work, Benton et al. interpret computations by requiring preservation of *all* relations on heaps that respect the effects of the computations, i.e., that ensure well-typed reads at locations with read effects and is closed under well-typed writes to locations with write effects. Our approach is more simple-minded; we do, in some sense, just preserve one relation. But it is unclear what the additional relations buy — we know of no equivalences that fail due to our approach; indeed the two alternatives may very well be equivalent. What is clear, however, is that losing the many relations, as well as the absence of a silent region, simplifies proofs considerably: writing out all the details, as we do, is obviously quite verbose, but in essence one may argue using simple diagrams, as illustrated in Section 7.

8.2. On expressiveness

In this paper we have focused on a relatively simple model that is still sufficiently rich to verify effect-based program transformations, where all the conditions for a transformation are expressed by types and effects rather than relying on the syntax of the transformed program. That said, there are some limitations imposed by the world model we chose: for example, the usual inductive argument cannot be used to relate a pure version of Fibonacci with the imperative version, since the partial bijection model we use is not expressive enough.[7] The expressivity of the world model, however, is an issue orthogonal to the rest of the interpretation of types, so a more expressive one — like the ones of Ahmed et al. [18] or Dreyer et al. [20] — could be used instead, if desired. In this paper, we refrained from doing so in order to keep that part of the setup as simple as possible.

In particular, consider the following variant of the so-called awkward example, discussed in detail in [20]. Let $\tau = (1 \xrightarrow{\varepsilon_0} 1) \xrightarrow{\varepsilon_1} \mathsf{int}$ with $\varepsilon_0 = \{rd_\rho, wr_\rho\}$ and $\varepsilon_1 = \{rd_\rho, wr_\rho, rd_\sigma, wr_\sigma\}$ and define expressions e_1 and e_2, which both have type τ and effect $\{al_\sigma\}$, by

$$\begin{aligned} e_1 &= \mathtt{let}\, x = \mathtt{ref}_\sigma\ 0\ \mathtt{in} \\ &\quad\ \lambda f.(x := 0; f(); x := 1; f(); !x) \\ e_2 &= \mathtt{let}\, x = \mathtt{ref}_\sigma\ 0\ \mathtt{in} \\ &\quad\ \lambda f.(f(); f(); x := 1; !x). \end{aligned}$$

We have used the subscript σ to indicate that x will have the type $\mathsf{ref}_\sigma\,\mathsf{int}$. Since ρ and σ are distinct, f cannot read or write the reference bound to x; this ensures that both the left and the right hand side functions always return 1. Indeed we can show that e_1 and e_2 are contextually equivalent. We remark that if, on the other hand, we had typed e_1 and e_2 with the same region variable instead of two distinct region variables, then our semantic model would not be expressive enough to show that e_1 and e_2 are contextually equivalent. For that, we could extend our model using ideas from the model for non-effect-annotated types in [20]. This way, the effect information can be used to restrict the applicable contexts and can thus make it easier to show two expressions equivalent (for those restricted contexts). Note that a standard unification-based algorithm for inferring effects, would infer the type τ for e_1 with ρ and σ distinct.

8.3. On region polymorphism

Until now, relational models of region-based type and effect systems for validating program equivalences have not included region polymorphism. Traditionally, region polymorphism involved only simple quantification over region variables [3]. Our new form of region polymorphism can express restrictions on which regions can be used to instantiate the quantified region variables, and thus restrict aliasing of regions. The type and effect system for Deterministic Parallel Java (DPJ) [10] includes explicit region disjointness constraints, which also makes it possible to restrict aliasing of regions. In future work, we would like to explore the connections between these two approaches further and investigate if our modeling techniques can be used to give a relational model for the rich region calculus used in DPJ.

[7] One could still imagine a proof that follows by taking related arguments, computing the results purely in the operational semantics, and showing these are related, though.

9. Conclusion and future work

We have presented a solution to the open problem of constructing a relational model for an effect system for a higher-order language with dynamically allocated higher-order store. We have demonstrated that the model can be used to rigorously justify effect-based program transformations.

In this paper, our conceptual region model has followed the stack discipline used in Tofte and Talpin's region-based memory management [4]. It would be interesting to explore also models based on more liberal region schemes, such as those in [27,28].

Future work also includes extending the language and model with concurrency, so that one can show the correctness of effect-based parallelization. A first model for such an extension has recently been presented by some of the current authors [29], using the same notion of worlds as in the present paper. Logical relations for concurrency have since also been explored with more sophisticated worlds (for simple type systems without effects), which are useful for reasoning about fine-grained concurrent data structures [30].

In the present approach we can hide local effects using the masking rule. In future work, we hope to investigate hiding of other forms of effects that are local qua data abstraction. For example, a lookup function on an abstract tree type should only have a read effect, even though it may also rewrite the structure of the tree internally (an internal effect). Here we hope to build on ideas from recent work on fictional separation logic [31].

Appendix A. Metric spaces and metric domains

We assume familiarity with the basics of metric spaces theory: the definition of a metric space, convergence and Cauchy sequences. Our interest is in metric spaces with three additional characteristics: They must be *complete*, i.e., all Cauchy sequences must be convergent. They must be *bisected*, i.e., all non-zero distances are of the form 2^{-n} for some $n \in \mathbb{N}$. And they must be *ultrametric* spaces. A metric space (X, d) is an ultrametric space if the metric satisfies the *strong triangle equality*:

$$\forall x, y, z \in X.\, d(x,z) \leq \max\{d(x,y), d(y,z)\}.$$

To ease the language, we refer to a complete, bisected ultrametric space as a *metric domain*, provided that it is non-empty. After all, empty metric spaces are not much fun.

The standard, geometric understanding of metric spaces goes badly with metric domains; the strong triangle inequality, e.g., outlaws partially overlapping balls. A better intuition, at least for our purposes, is this: metric domains are all about approximations. Let (X, d) be a metric domain; we imagine that every element $x \in X$ can be approximated to level n for any $n \in \mathbb{N}$. The zeroth approximation loses all information, the first approximation gives a very rough estimate and every successive level of approximation adds more details. Having $d(x, y) = 2^{-n}$ with $n \in \mathbb{N}$ for two elements $x, y \in M$ then conceptually means that x and y are indiscernible at approximation level n but differ at level $n+1$. Most often, we are interested only in the former property, and so we write $x \overset{n}{=} y$ to mean $d(x, y) \leq 2^{-n}$ and we say that x and y are *n-equal*; the relation $\overset{x}{=}$ we call *n-equality*.

A function $f : X \to Y$ between (the carrier sets of) two metric domains is considered *non-expansive* if, for any $n \in \mathbb{N}$, they preserve n-equality, i.e., if $x \overset{n}{=} y$ implies $f(x) \overset{n}{=} f(y)$ for any two $x, y \in X$. *Contractive* functions are those with the stronger property that $x \overset{n}{=} y$ implies $f(x) \overset{n+1}{=} f(y)$. Notation is lax already: the actual metrics on X and Y are implicit. The non-expansive functions are the natural maps on metric domains; the subset of contractive functions is interesting for the following reason:

Theorem 34 *(Banach's fixed point). A contractive endo-function on a metric domain has a unique fixed point.*

The metric domains with non-expansive functions between them form a *Cartesian closed category* **MetDom**. The product of metric domains X and Y is the set $X \times Y$ with the following defining property of the metric:

$$\forall n \in \mathbb{N}.\ \forall (x_1, y_1), (x_2, y_2) \in X \times Y.$$
$$(x_1, y_1) \overset{n}{=} (x_2, y_2) \iff x_1 \overset{n}{=} x_2 \wedge y_1 \overset{n}{=} y_2.$$

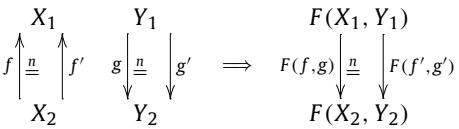
The exponential Y^X consists of the set $X \to_{nex} Y$ of non-expansive functions from X to Y and a metric defined by the following property:

$$\forall n \in \mathbb{N}.\ \forall f, g \in X \to_{nex} Y.$$
$$f \overset{n}{=} g \iff \forall x \in X.\, f(x) \overset{n}{=} g(x).$$

A functor $F : \mathbf{MetDom}^{\mathrm{op}} \times \mathbf{MetDom} \to \mathbf{MetDom}$ is *locally non-expansive* if, for any four metric domains X_1, Y_1, X_2 and Y_2, and any four morphisms $f, f' : X_2 \to_{nex} X_1$ and $g, g' : Y_1 \to_{nex} Y_2$ we have that

$$\forall n \in \mathbb{N}.\ f \overset{n}{=} f' \wedge g \overset{n}{=} g' \Longrightarrow F(f,g) \overset{n}{=} F(f',g'),$$

and it is *locally contractive* if we get $(n+1)$-equality on the right hand side. We essentially require the action on hom-sets to be non-expansive, respectively, contractive; here is the illustrated version:



The use of metric domains in the solution to the type-world circularity and similar circularities is due to the following theorem by to America and Rutten [32]. It says that one can solve recursive domain equations in **MetDom**; the proof is essentially an adaptation of the inverse-limit method from classical domain theory:

Theorem 35. *Let* $F : \mathbf{MetDom}^{\mathrm{op}} \times \mathbf{MetDom} \to \mathbf{MetDom}$ *be a locally contractive functor. Then there exists a unique (up to isomorphism) metric domain* X *such that* $X \cong F(X,X)$.

Everyday constructions often give functors that are locally non-expansive but not locally contractive; to apply the above theorem, it is a standard technique to post-compose the *shrinking functor*

$$\frac{1}{2} : \mathbf{MetDom} \to \mathbf{MetDom}.$$

This functor leaves the underlying sets untouched but halves all the distances; it is the identity on morphisms.

A few concluding remarks to this subsection are appropriate: first, if you already know metric spaces, then you may balk at the definitions of non-expansiveness and contractiveness as well as the constructions of products and exponentials. Rest assured, though, that for metric domains, our definitions are equivalent to the standard ones. Elsewhere, the category **MetDom** is known as $\mathbf{BiCBUlt}_{ne}$, a more precise if less catchy name. Benton, Birkedal, Kennedy and Varming have given a Coq formalization of all theory in this subsection and more; Støvring and the two first authors have given a generalization of the Theorem 35 [33] to metric domains that carry additional structure such as, e.g., certain orderings.

A.1. The present type-world circularity

We now proceed to build the metric domains of types **T** and worlds **W** that is our model. By comparison to the simple type-world circularity for an ML-like language given by Reus, Schwinghammer, Støvring, Yang and the two first authors [17] it has a few more quirks, but the approach is quite the same.

Above, we only defined local non-expansiveness for functors going from $\mathbf{MetDom}^{op} \times \mathbf{MetDom}$ to **MetDom**, but the concept goes for endo-functors on **MetDom** as well:

Lemma 36. *For any two sets* X *and* Y*, the construction of finite, decorated partial bijections in Fig. 3 is a functor from* **Set** *to* **Set** *by abstracting out* **Z**. *It extends to a locally non-expansive functor* $\mathrm{ParBij}(X,Y,-) : \mathbf{MetDom} \to \mathbf{MetDom}$.

The distance between two elements $P, Q \in \mathrm{ParBij}(X,Y,\mathbf{Z})$ is 1 if the partial bijections, excluding the elements of **Z**, are different. Otherwise it is the maximum of the distance between corresponding elements of **Z**.

Lemma 37. *The construction of worlds* **W** *from* $\widehat{\mathbf{T}}$ *in Fig. 3 can be seen as a functor* **Set** $\to$ **Set** *by abstracting out* $\widehat{\mathbf{T}}$. *It extends to a locally non-expansive functor* $\mathrm{Worlds} : \mathbf{MetDom} \to \mathbf{MetDom}$.

For a metric domain **X**, the distance between two elements (φ_1,ψ_1), (φ_2,ψ_2) in $\mathrm{Worlds}(\mathbf{X})$ is 1 if either $\mathrm{dom}(\varphi_1) \neq \mathrm{dom}(\varphi_2)$ or $\psi_1 \neq \psi_2$. Otherwise, it is the maximum of the distances between $\varphi_1(r)$ and $\varphi_2(r)$ as elements of $\mathrm{ParBij}(\mathcal{L},\mathcal{L},\mathbf{X})$ over all $r \in \mathrm{dom}(\varphi_1)$.

We recall from Section 4 that for any set X, $URel(X)$ is the set of indexed, downwards closed relations on X, i.e.,

$$URel(X) = \{R \subseteq \mathbb{N} \times X \times X \mid \forall (k,x_1,x_2) \in R.\, \forall j < k.\, (j,x_1,x_2) \in R\}.$$

We can restrict a relation $R \in URel(X)$ to elements with index strictly less than some $n \in \mathbb{N}$

$$R|_n = \{(k,x_1,x_2) \mid (k,x_1,x_2) \in R \wedge k < n\}$$

and this gives us a metric:

Proposition 38. *For any set* X, $URel(X)$ *is a metric domain by a metric with the following defining property:*

$$\forall n \in \mathbb{N}.\, \forall R,S \in URel(X).\, R \overset{n}{=} S \iff R|_n = S|_n.$$

Theorem 39. *We can read the entire construction of types* $\mathbf{T}$ *from* $\widehat{\mathbf{T}}$ *in Fig. 3 as a locally non-expansive functor* $\mathrm{Types} : \mathbf{MetDom}^{op} \to \mathbf{MetDom}$ *given on objects by* $\mathrm{Types}(X) = \mathrm{Worlds}(\mathrm{ParBij}(\mathcal{L}, \mathcal{L}, \mathbf{X})) \to_{nex,mon} \mathit{URel}(\mathcal{V})$.

Note that we do not take all monotone functions as carrier set, just the non-expansive ones. The ordering on the metric domain $\mathrm{Worlds}(\mathrm{ParBij}(\mathcal{L}, \mathcal{L}, \mathbf{X}))$ is the reflexive, transitive closure of the transitions in Fig. 4 and on $\mathit{URel}(\mathcal{V})$ it is set-theoretic inclusion. For once, we need a bit of metric tinkering: first we form the exponential

$$\mathit{URel}(\mathcal{V})^{\mathrm{Worlds}(\mathrm{ParBij}(\mathcal{L},\mathcal{L},X))}$$

in **MetDom**, i.e., take all the non-expansive functions, then we restrict to the monotone ones. That we can do this without losing completeness in the process must be verified; it is a consequence of the fact that inclusion is a *continuous preorder* on $\mathit{URel}(\mathcal{V})$, i.e., that for any two sequences $(R)_{n\in\mathbb{N}}$ and $(S)_{n\in\mathbb{N}}$ with limits R, S in $\mathit{URel}(\mathcal{V})$ we have

$$\left[\forall n \in \mathbb{N}.\, R_n \subseteq S_n\right] \Longrightarrow R \subseteq S.$$

All that remains now, is to post-compose the shrinking functor; this gives us the locally contractive functor

$$\frac{1}{2} \cdot \mathrm{Types} : \mathbf{MetDom}^{op} \times \mathbf{MetDom} \to \mathbf{MetDom}$$

An application of Theorem 35 yields an object $\widehat{\mathbf{T}}$ of **MetDom** and the desired isomorphism:

$$\iota : \widehat{\mathbf{T}} \cong \mathrm{Types}(\widehat{\mathbf{T}}) = \mathbf{T}.$$

References

[1] J. Thamsborg, L. Birkedal, A Kripke logical relation for effect-based program transformations, in: ICFP, ACM, 2011, pp. 445–456.
[2] D. Gifford, J. Lucassen, Integrating functional and imperative programming, in: LISP and Functional Programming, ACM, 1986, pp. 28–38.
[3] J. Lucassen, D. Gifford, Polymorphic effect systems, in: POPL, ACM, 1988, pp. 47–57.
[4] M. Tofte, J.-P. Talpin, Implementation of the typed call-by-value λ-calculus using a stack of regions, in: POPL, ACM, 1994, pp. 188–201.
[5] F. Henglein, H. Makholm, H. Niss, Effect types and region-based memory management, in: B. Pierce (Ed.), Advanced Topics in Types and Programming Languages, MIT Press, 2005.
[6] L. Birkedal, M. Tofte, M. Vejlstrup, From region inference to von Neumann machines via region representation inference, in: POPL, ACM, 1996, pp. 171–183.
[7] D. Grossman, J.G. Morrisett, T. Jim, M.W. Hicks, Y. Wang, J. Cheney, Region-based memory management in cyclone, in: PLDI, ACM, 2002, pp. 282–293.
[8] G. Boudol, Typing termination in a higher-order concurrent imperative language, Inf. Comput. 208 (6) (2010) 716–736.
[9] R. Amadio, On stratified regions, in: APLAS, Springer, 2009, pp. 210–225.
[10] R. Bocchino, V. Adve, S. Adve, S. Heumann, R. Komuravelli, J. Overbey, P. Simmons, H. Sung, M. Vakilian, A type and effect system for deterministic parallel Java, in: OOPSLA, ACM, 2009, pp. 97–116.
[11] R. Bocchino, V. Adve, Types, regions, and effects for safe programming with object-oriented parallel frameworks, in: Proceedings of ECOOP, Springer, 2011, pp. 306–332.
[12] N. Benton, A. Kenney, M. Hofmann, L. Beringer, Reading, writing and relations: towards extensional semantics for effect analyses, in: APLAS, Springer, 2006, pp. 114–130.
[13] N. Benton, P. Buchlovsky, Semantics of an effect analysis for exceptions, in: TLDI, ACM, 2007, pp. 15–26.
[14] N. Benton, L. Beringer, M. Hofmann, A. Kennedy, Relational semantics for effect-based program transformations with dynamic allocation, in: PPDP, ACM, 2007, pp. 87–96.
[15] N. Benton, L. Beringer, M. Hofmann, A. Kennedy, Relational semantics for effect-based program transformations: higher-order store, in: PPDP, ACM, 2009, pp. 301–312.
[16] A. Ahmed, Semantics of types for mutable state, PhD thesis, Princeton University, 2004.
[17] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, H. Yang, Step-indexed Kripke models over recursive worlds, in: POPL, ACM, 2011, pp. 119–132.
[18] A. Ahmed, D. Dreyer, A. Rossberg, State-dependent representation independence, in: POPL, ACM, 2009, pp. 340–353.
[19] L. Birkedal, K. Støvring, J. Thamsborg, Realizability semantics of parametric polymorphism, general references, and recursive types, in: FOSSACS, Springer, 2009, pp. 456–470.
[20] D. Dreyer, G. Neis, L. Birkedal, The impact of higher-order state and control effects on local relational reasoning, in: ICFP, ACM, 2010, pp. 143–156.
[21] L. Birkedal, N. Torp-Smith, H. Yang, Semantics of separation-logic typing and higher-order frame rules for algol-like languages, Log. Methods Comput. Sci. 2 (5) (2006) 1–33.
[22] L. Birkedal, K. Støvring, J. Thamsborg, Realisability semantics of parametric polymorphism, general references and recursive types, Math. Struct. Comput. Sci. 20 (4) (2010) 655–703.
[23] A. Nanevski, G. Morrisett, L. Birkedal, Polymorphism and separation in hoare type theory, in: J.H. Reppy, J.L. Lawall (Eds.), ICFP, ACM, 2006, pp. 62–73.
[24] A.M. Pitts, Relational properties of domains, Inf. Comput. 127 (2) (1996) 66–90.
[25] A.J. Ahmed, Step-indexed syntactic logical relations for recursive and quantified types, in: P. Sestoft (Ed.), ESOP, in: Lecture Notes in Computer Science, vol. 3924, Springer, 2006, pp. 69–83.
[26] D. Dreyer, A. Ahmed, L. Birkedal, Logical step-indexed logical relations, in: LICS, IEEE Computer Society, 2009, pp. 71–80.
[27] A. Aiken, M. Fähndrich, R. Levien, Better static memory management: improving region-based analysis of higher-order languages, in: PLDI, ACM, 1995, pp. 174–185.
[28] D. Walker, K. Watkins, On regions and linear types, in: ICFP, ACM, 2001, pp. 181–192.
[29] L. Birkedal, F. Sieczkowski, J. Thamsborg, A concurrent logical relation, in: CSL, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012, pp. 107–121.
[30] A.J. Turon, J. Thamsborg, A. Ahmed, L. Birkedal, D. Dreyer, Logical relations for fine-grained concurrency, in: POPL, ACM, 2013, pp. 343–356.
[31] J.B. Jensen, L. Birkedal, Fictional separation logic, in: ESOP, Springer, 2012, pp. 377–396.
[32] P. America, J.J.M.M. Rutten, Solving reflexive domain equations in a category of complete metric spaces, J. Comput. Syst. Sci. 39 (3) (1989) 343–375.
[33] L. Birkedal, K. Støvring, J. Thamsborg, The category-theoretic solution of recursive metric-space equations, Theor. Comput. Sci. 411 (47) (2010) 4102–4122.