

The Transitive Composability of Relation Transition Systems

Chung-Kil Hur Georg Neis Derek Dreyer Viktor Vafeiadis

Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern and Saarbrücken, Germany
E-mail: {gil, neis, dreyer, viktor}@mpi-sws.org

Abstract—Relation Transition Systems (RTSs) have recently been proposed as a foundation for reasoning effectively about program equivalence in higher-order imperative languages like ML. RTSs fruitfully synthesize the coinductive style of bisimulation-based methods with the treatment of local state in recent work on step-indexed Kripke logical relations (SKLRs). Like SKLRs, RTSs are designed to have the potential to scale to inter-language reasoning; but unlike SKLRs, RTS proofs are also transitively composable, which is of critical importance for applications such as multi-stage verified compilation.

In a POPL’12 paper [6], we presented the first RTS model for an ML-like core language, F^{μ^1} , supporting higher-order functions, recursive types, abstract types, and general mutable references, and we proved soundness of the model w.r.t. contextual equivalence. In addition, we briefly sketched the proof that RTSs are transitively composable, but our proof only covered a restricted fragment of the language/model omitting abstract types and mutable state. Here, we present the transitivity proof for the full RTS model of the full F^{μ^1} language. The proof is highly intricate, requiring a number of technical innovations. We have mechanized all our results in Coq.

I. INTRODUCTION

A longstanding problem in semantics is to find effective methods for reasoning about program equivalence in ML-like languages supporting both functional and imperative features. In recent years, considerable progress has been made on this problem, primarily by advancements to two different classes of proof methods—*bisimulations* [12, 8, 9] and *step-indexed Kripke logical relations (SKLRs)* [2, 4].

In a paper appearing in POPL’12 [6], we proposed a new method for proving equivalences in ML-like languages, which we call *Relation Transition Systems (RTSs)*. RTSs draw inspiration from—and join together some of the best features of—bisimulations and SKLRs. In particular, RTSs support reasoning about recursive features in a convenient (step-index-free) *coinductive* style, as bisimulations do; but they also provide a very flexible treatment of “local” state, closely following recent work on SKLRs, in which invariants on the evolution of a piece of local state are expressed and enforced using a *state transition system* [2, 4].

Our ulterior motivation for developing RTSs was to overcome some basic limitations of bisimulations and SKLRs with regards to their potential for *inter-language reasoning*, *i.e.*, reasoning compositionally about equivalences between programs written in different languages, such as the source

and target of a verified compiler [3, 5]. Existing bisimulation methods for higher-order stateful languages—*e.g.*, *environmental* and *normal form* bisimulations [8, 10, 9]—rely crucially on “syntactic” devices (*e.g.*, context closure) in order to deal properly with unknown higher-order values that may be passed in as function arguments. These syntactic devices are appropriate for proving “contextual” properties—including but not limited to contextual equivalence [11]—but they bake in the assumption that the programs being related share a common syntactic notion of “context”, which is clearly not a valid assumption in the inter-language setting.

In contrast, SKLRs *have* been successfully generalized to the inter-language setting—specifically, to the goal of establishing “compositional compiler correctness” [3, 5]—but they suffer from an orthogonal limitation, namely that in general they are not *transitively composable*. Ahmed grappled with this issue in her first paper on binary step-indexed logical relations [1]: her solution involved building the model over syntactically well-typed terms, an option that is not available, in general, when constructing relational models over low-level languages (*e.g.*, assembly). Moreover, her technique only helped in proving transitivity of her model for a simply-typed λ -calculus with recursive types, and has not been shown to scale to richer languages/models.

SKLRs’ lack of transitivity has not been a major point of concern in prior work on proving contextual equivalences, since the latter are (by definition) transitively composable. However, in the inter-language setting, one can no longer rely on contextual equivalence as a crutch. For example, the correctness of a realistic, multi-stage compiler is only feasible to establish by transitively composing the correctness results for its constituent stages, but those correctness results cannot generally be phrased as contextual equivalences.

RTSs avoid the aforementioned limitations by means of a new technique, which we call *global vs. local knowledge*. The idea is to distinguish one’s “local knowledge” about program equivalence (*i.e.*, the terms whose equivalence one wishes to prove) from the “global knowledge” (*i.e.*, the relation defining when unknown higher-order values passed in from the context are equivalent), and to parameterize the proof of correctness for the former over the latter. By *parameterizing over* the global knowledge instead of attempting to characterize it directly, RTSs (a) sidestep the

need for any “syntactic” devices that would preclude inter-language reasoning, and (b) become transitively composable.

In our previous paper [6], we presented an RTS model for an ML-like core language, $F^{\mu^!}$, supporting higher-order functions, recursive types, abstract types, and general mutable references. We proved soundness of the model w.r.t. contextual equivalence, and gave several interesting examples of its use. We also briefly sketched the proof that RTSs are transitively composable, but our proof only covered a restricted fragment of the language/model omitting abstract types and mutable state. Extending the proof to handle those features turns out (unsurprisingly) to be highly non-trivial.

In this paper, we generalize the proof of RTS transitivity to account for the full RTS model of the full $F^{\mu^!}$ language. This is a critical stepping stone on the path toward generalizing RTSs to support effective inter-language reasoning. We begin in Sections II and III by reviewing the details of our language/model. In Section IV, we describe the high-level structure of our transitivity proof. The proof divides into two major parts, presented in Sections V and VI, respectively. As the proof is highly intricate, requiring several tricky auxiliary constructions, we cannot provide it in its entirety here. Rather, we highlight the key technical challenges, explain carefully the central constructions and the intuitions behind them, and sketch the proofs of the main lemmas.

The full proof appears in a companion technical appendix, available at the following URL:

<http://www.mpi-sws.org/~dreyer/papers/rts-trans/>

The proof has also been fully machine-checked in Coq, and our Coq source files are available at the above URL as well. To give a rough sense of the complexity of the transitivity proof, it required approximately 3200 lines of Coq, versus 1500 lines to formalize the language, 400 lines to formalize the model, and 2000 lines to prove soundness of the model w.r.t. contextual equivalence.

II. THE LANGUAGE $F^{\mu^!}$

Figure 1 gives the syntax of $F^{\mu^!}$, along with its static and dynamic semantics judgment forms. $F^{\mu^!}$ is equipped with a standard type system, as well as a standard CBV dynamic semantics using evaluation contexts (aka continuations) K , and finite heaps h . Memory allocation is deterministic, according to an unknown strategy (see [6] for details).

Here, following Ahmed [1], we only show the syntax of *type-erased* terms (aka expressions) e , over which both the dynamic semantics and our RTS model are constructed. The typeful syntax of $F^{\mu^!}$ programs p is given in the technical appendix. We distinguish between *type variables* (α) and *type names* (\mathbf{n}). The latter appear only in the model and are like type constants with no primitive intro/elim forms. CType is the set of closed types (with no free *variables*).

III. RELATION TRANSITION SYSTEMS

Figures 2–7 display our relation transition systems (RTS)

$$\begin{aligned}
\tau_{\text{base}} &::= \text{unit} \mid \text{int} \mid \text{bool} \\
\tau \in \text{Type} &::= \alpha \mid \tau_{\text{base}} \mid \tau_1 \times \tau_2 \mid \tau_1 + \tau_2 \mid \tau_1 \rightarrow \tau_2 \mid \mu\alpha. \tau \mid \\
&\quad \forall\alpha. \tau \mid \exists\alpha. \tau \mid \text{ref } \tau \mid \mathbf{n} \quad (\mathbf{n} \in \text{TyNam}) \\
v \in \text{Val} &::= x \mid \langle \rangle \mid n \mid \text{tt} \mid \text{ff} \mid \langle v_1, v_2 \rangle \mid \text{inj}^i v \mid \text{roll } v \mid \\
&\quad \text{fix } f(x). e \mid \Lambda. e \mid \text{pack } v \mid \ell \\
e \in \text{Exp} &::= v \mid \text{if } e_0 \text{ then } e_1 \text{ else } e_2 \mid \langle e_1, e_2 \rangle \mid e.i \mid \\
&\quad \text{inj}^i e \mid (\text{case } e \text{ of } \text{inj}^1 x \Rightarrow e_1 \mid \text{inj}^2 x \Rightarrow e_2) \mid \\
&\quad \text{roll } e \mid \text{unroll } e \mid e_1 e_2 \mid e[] \mid \\
&\quad \text{pack } e \mid \text{unpack } e_1 \text{ as } x \text{ in } e_2 \mid \\
&\quad \text{ref } e \mid !e \mid e_1 := e_2 \mid e_1 == e_2 \\
K \in \text{Cont} &::= \bullet \mid \text{if } K \text{ then } e_1 \text{ else } e_2 \mid \langle K, e \rangle \mid \langle v, K \rangle \mid K.i \mid \\
&\quad \text{inj}^i K \mid (\text{case } K \text{ of } \text{inj}^1 x \Rightarrow e_1 \mid \text{inj}^2 x \Rightarrow e_2) \mid \\
&\quad \text{roll } K \mid \text{unroll } K \mid K e \mid v K \mid K[] \mid \\
&\quad \text{pack } K \mid \text{unpack } K \text{ as } x \text{ in } e \mid \text{ref } K \mid !K \mid \\
&\quad K := e \mid v := K \mid K == e \mid v == K \\
h \in \text{Heap} &::= \text{Loc} \stackrel{\text{fin}}{\text{CVal}} \quad \text{Loc} = \{\ell_1, \ell_2, \dots\} \\
\Delta &::= \cdot \mid \Delta, \alpha \quad \Gamma ::= \cdot \mid \Gamma, x : \tau \\
\textbf{Typing: } \Delta; \Gamma \vdash e : \tau &\quad \textbf{Small-step semantics: } h, e \hookrightarrow h', e'
\end{aligned}$$

Figure 1. The language $F^{\mu^!}$.

$$\begin{aligned}
\text{CTypeF} &::= \{(\tau_1 \rightarrow \tau_2) \in \text{CType}\} \cup \{(\forall\alpha. \tau) \in \text{CType}\} \cup \\
&\quad \{\mathbf{n} \in \text{TyNam}\} \cup \{\text{ref } \tau \in \text{CType}\} \\
\text{VRel} &::= \text{CType} \rightarrow \mathbb{P}(\text{CVal} \times \text{CVal}) \\
\text{VRelF} &::= \text{CTypeF} \rightarrow \mathbb{P}(\text{CVal} \times \text{CVal}) \\
\text{ERel} &::= \text{CType} \rightarrow \mathbb{P}(\text{CExp} \times \text{CExp})
\end{aligned}$$

Figure 2. Flexible types (CTypeF) and other semantic domains.

$$\begin{aligned}
\overline{R}(\tau) &::= R(\tau) \quad \text{if } \tau \in \text{CTypeF} \\
\overline{R}(\tau_{\text{base}}) &::= \{(v, v) \mid \vdash v : \tau_{\text{base}}\} \\
\overline{R}(\tau_1 \times \tau_2) &::= \{((v_1, v'_1), (v_2, v'_2)) \mid \\
&\quad (v_1, v_2) \in \overline{R}(\tau_1) \wedge (v'_1, v'_2) \in \overline{R}(\tau_2)\} \\
\overline{R}(\tau_1 + \tau_2) &::= \{(\text{inj}^1 v_1, \text{inj}^1 v_2) \mid (v_1, v_2) \in \overline{R}(\tau_1)\} \cup \\
&\quad \{(\text{inj}^2 v_1, \text{inj}^2 v_2) \mid (v_1, v_2) \in \overline{R}(\tau_2)\} \\
\overline{R}(\mu\alpha. \tau) &::= \{(\text{roll } v_1, \text{roll } v_2) \mid (v_1, v_2) \in \overline{R}(\tau[\mu\alpha. \tau/\alpha])\} \\
\overline{R}(\exists\alpha. \tau) &::= \{(\text{pack } v_1, \text{pack } v_2) \mid \exists\tau'. (v_1, v_2) \in \overline{R}(\tau[\tau'/\alpha])\}
\end{aligned}$$

Figure 3. Inductive def. of value closure (if $R \in \text{VRelF}$, then $\overline{R} \in \text{VRel}$).

model for $F^{\mu^!}$. For space reasons, we can only provide here a brief review of the model. For a fuller exposition, see [6].

Worlds. Proving that two terms are equivalent using our model $(\Delta; \Gamma \vdash e_1 \sim e_2 : \tau)$ involves constructing a “world” W , which codifies (a) the invariants on the state maintained by e_1 and e_2 , and (b) the relational interpretations of any abstract types they define. In order to model equivalence of terms whose local state *evolves* over time (e.g., “generative” classes/ADTs, whose instances/inhabitants grow dynamically when certain of their methods are invoked), the world W takes the form of a state transition system (STS). As shown in Figure 4, this consists of a preorder \sqsubseteq on some state set S , together with a set of “owned” type names N (recording which abstract types are defined semantically by the world) and two state-dependent relations: H , which says what heaps are related (i.e., obey the “current” state invariant), and L , which describes what values are known to be equivalent, at any given state $s \in S$. (We call L the “local knowledge” of W , and discuss it in more detail below.)

$$\begin{aligned}
\text{beta}(e) &:= \begin{cases} e' & \text{if } \forall h. h, e \hookrightarrow h, e' \\ \text{undef} & \text{otherwise} \end{cases} \\
\text{Know}(\mathcal{N}) &:= \{ R \in \text{VRelF} \mid \forall \mathbf{n} \notin \mathcal{N}. R(\mathbf{n}) = \emptyset \\
&\quad \wedge (\forall \tau_1, \tau_2. \forall (f_1, f_2) \in R(\tau_1 \rightarrow \tau_2). \forall i, v. \text{beta}(f_i v) \text{ defined}) \\
&\quad \wedge (\forall \alpha, \tau. \forall (f_1, f_2) \in R(\forall \alpha. \tau). \forall i. \text{beta}(f_i[]) \text{ defined}) \} \\
\text{World} &:= \{ W = (\mathbf{S}, \sqsubseteq, \mathbf{N}, \mathbf{L}, \mathbf{H}) \in \text{Set} \times \text{Preord}(\mathbf{S}) \times \mathbb{P}(\text{TyNam}) \times \\
&\quad (\mathbf{S} \rightarrow \text{VRelF} \rightarrow \text{VRelF}) \times (\mathbf{S} \rightarrow \text{VRelF} \rightarrow \text{HRel}) \mid \\
&\quad \mathbf{L} \text{ monotone in 1st, 2nd args w.r.t. } \sqsubseteq, \subseteq \wedge \\
&\quad \forall s, R. \mathbf{L}(s)(R) \in \text{Know}(\mathbf{N}) \wedge \\
&\quad \mathbf{H} \text{ monotone in 2nd arg w.r.t. } \subseteq \}
\end{aligned}$$

Figure 4. Definition of worlds.

$$\begin{aligned}
\text{dom}_{[1]}(s^{\text{rf}}) &:= \{ \ell_1 \mid \exists \tau, \ell_2. (\tau, \ell_1, \ell_2) \in s^{\text{rf}} \} \\
\text{dom}_{[2]}(s^{\text{rf}}) &:= \{ \ell_2 \mid \exists \tau, \ell_1. (\tau, \ell_1, \ell_2) \in s^{\text{rf}} \} \\
W_{\text{ref}}.\mathbf{S} &:= \{ s^{\text{rf}} \in \mathbb{P}_{\text{fin}}(\text{CType} \times \text{Loc} \times \text{Loc}) \mid \\
&\quad \forall (\tau, \ell_1, \ell_2), (\tau', \ell'_1, \ell'_2) \in s^{\text{rf}}. (\ell_1 = \ell'_1 \implies \tau = \tau' \wedge \ell_2 = \ell'_2) \wedge \\
&\quad (\ell_2 = \ell'_2 \implies \tau = \tau' \wedge \ell_1 = \ell'_1) \} \\
W_{\text{ref}}.\sqsubseteq &:= \subseteq \quad W_{\text{ref}}.\mathbf{N} := \emptyset \\
W_{\text{ref}}.\mathbf{L}(s^{\text{rf}})(R)(\text{ref } \tau) &:= \{ (\ell_1, \ell_2) \mid (\tau, \ell_1, \ell_2) \in s^{\text{rf}} \} \\
W_{\text{ref}}.\mathbf{H}(s^{\text{rf}})(R) &:= \{ (h_1, h_2) \mid \forall i. \text{dom}(h_i) = \text{dom}_{[i]}(s^{\text{rf}}) \wedge \\
&\quad \forall (\tau, \ell_1, \ell_2) \in s^{\text{rf}}. (h_1(\ell_1), h_2(\ell_2)) \in \overline{R}(\tau) \} \\
\text{LWorld} &:= \{ w = (\mathbf{S}, \sqsubseteq, \mathbf{N}, \mathbf{L}, \mathbf{H}) \in \text{Set} \times \text{Preord}(\mathbf{S}) \times \mathbb{P}(\text{TyNam}) \times \\
&\quad (W_{\text{ref}}.\mathbf{S} \rightarrow \mathbf{S} \rightarrow \text{VRelF} \rightarrow \text{VRelF}) \times (W_{\text{ref}}.\mathbf{S} \rightarrow \mathbf{S} \rightarrow \text{VRelF} \rightarrow \text{HRel}) \mid \\
&\quad \mathbf{L} \text{ monotone in 1st, 2nd, 3rd args w.r.t. } W_{\text{ref}}.\sqsubseteq, \subseteq, \subseteq \wedge \\
&\quad \forall s^{\text{rf}}, s, R. \mathbf{L}(s^{\text{rf}})(s)(R) \in \text{Know}(\mathbf{N}) \wedge \\
&\quad \mathbf{H} \text{ monotone in 3rd arg w.r.t. } \subseteq \}
\end{aligned}$$

Figure 5. Definitions of W_{ref} and local worlds.

In our original RTS model [6], worlds actually contain two preorders on the state set \mathbf{S} —one “public” and one “private”. As Dreyer *et al.* [4] have shown, this is useful for reasoning about “well-bracketed” state change. We have chosen to drop the second preorder here in this extended abstract to streamline the presentation, and because it does not introduce any challenges into the proof of RTS transitivity, our present focus. (The proof in the online appendix handles the full public/private model.)

Local Worlds and W_{ref} . To account for the ref type of $F^{\mu 1}$, we require that the world $W \in \text{World}$ be a combination of a fixed “shared world” W_{ref} and a proof-specific “local world” $w \in \text{LWorld}$ (Figure 5). The idea is that W_{ref} governs the semantics and invariants of the ref type, which are the same in all proofs; whereas w describes the invariants associated with e_1 ’s and e_2 ’s local state, as well as the relational interpretations of any abstract types they define. The local w is joined together with W_{ref} to form the full W by the “lifting” operator $w\uparrow$, which is defined in Figure 6 using a simple product construction of w ’s and W_{ref} ’s STSs.

An important point of note here is that the local world w is allowed to depend on the state s^{rf} of the shared world W_{ref} . The shared state s^{rf} is a partial bijection between (globally visible) memory locations, with each pair of related locations associated with the type τ of their contents, and the L and H components of w take $s^{\text{rf}} \in W_{\text{ref}}.\mathbf{S}$ as a parameter in addition to the local state $s \in w.\mathbf{S}$. Consequently, the definition of RTS equivalence (bottom of Figure 7) includes a side condition checking that w is *stable*—i.e., roughly,

$$\begin{aligned}
H \otimes H' &:= \{ (h_1 \uplus h'_1, h_2 \uplus h'_2) \mid (h_1, h_2) \in H \wedge (h'_1, h'_2) \in H' \} \\
w\uparrow.\mathbf{S} &:= W_{\text{ref}}.\mathbf{S} \times w.\mathbf{S} \\
w\uparrow.\sqsubseteq &:= \{ (p, p') \mid p.1 \sqsubseteq p'.1 \wedge p.2 \sqsubseteq p'.2 \} \\
w\uparrow.\mathbf{N} &:= w.\mathbf{N} \\
w\uparrow.\mathbf{L}(s^{\text{rf}}, s)(R) &:= W_{\text{ref}}.\mathbf{L}(s^{\text{rf}})(R) \cup w.\mathbf{L}(s^{\text{rf}})(s)(R) \\
w\uparrow.\mathbf{H}(s^{\text{rf}}, s)(R) &:= W_{\text{ref}}.\mathbf{H}(s^{\text{rf}})(R) \otimes w.\mathbf{H}(s^{\text{rf}})(s)(R)
\end{aligned}$$

Figure 6. Lifting ($\uparrow \in \text{LWorld} \rightarrow \text{World}$) of worlds.

that if two “local” heaps are related by $w.\mathbf{H}(s^{\text{rf}})(s)$, and the shared state s^{rf} advances to some $\hat{s}^{\text{rf}} \sqsupseteq s^{\text{rf}}$, then there must exist some $\hat{s} \sqsupseteq s$ such that the local heaps continue to be related by $w.\mathbf{H}(\hat{s}^{\text{rf}})(\hat{s})$. This stability condition is needed to ensure that, when different modules in a program update the state of the shared W_{ref} , they do not cause bad “interference” with one another’s local state invariants. Although for most practical purposes the extra parameterization of w over s^{rf} is unnecessary—in which case stability of w is trivial to show by choosing $\hat{s} := s$ —it will turn out to be critically useful in our transitivity proof (see Section VI).

Global vs. Local Knowledge. RTS proofs are much like bisimulation proofs in that they require one to declare up front all the equivalences between terms/values that one needs to know in order to establish the equivalence of the terms e_1 and e_2 in question. We call this set of putative equivalences the “local knowledge” of the proof, which constitutes the L component of the world W .

The object is then to demonstrate that this local knowledge is “consistent,” as defined in Figure 7, which implies that the equivalence is semantically justified. In particular, for any function values (f_1, f_2) related by $W.\mathbf{L}$ (at any given state $s \in W.\mathbf{S}$), we must show that f_1 and f_2 behave equivalently when applied to “related arguments”. But from what relation do we draw these “related arguments”?

Since the related arguments are passed in from somewhere in the program context, they might very well not be related by our local knowledge. Rather, we say they are drawn from the “global knowledge” about program equivalence. In the higher-order setting, characterizing this global knowledge directly is quite difficult; intuitively, it is as hard as coming up with a good model of program equivalence in the first place! Bisimulation-based approaches [8, 9] deal with this challenge by giving a *syntactic* characterization of the global knowledge, which (as we explained in the introduction) we do not want to do. Our approach instead is to avoid the problem altogether: rather than try to define the global knowledge directly, we *parameterize* our model over it.

We only require that the global knowledge G “respect” the world ($G \in \text{GK}(W)$, Figure 7), namely that: it must include the local knowledge, it must be monotone w.r.t. state changes, and it must not alter the meaning of the reference type nor of any abstract type name owned by the world. Otherwise, we place no restrictions on G . Notably, G may relate two *arbitrary* values at *any* function type, even values that are not functions! This seemingly perverse liberality is in fact essential to our transitivity proof (see Section V-A).

$$\begin{aligned}
R' \geq_{\text{ref}}^{\mathcal{N}} R & := R' \supseteq R \wedge \forall \tau. R'(\text{ref } \tau) = R(\text{ref } \tau) \wedge \forall \mathbf{n} \in \mathcal{N}. R'(\mathbf{n}) = R(\mathbf{n}) \\
\text{GK}(W) & := \{ G \in W.S \rightarrow \text{VRelF} \mid G \text{ is monotone w.r.t. } \sqsubseteq \wedge \forall s. G(s) \geq_{\text{ref}}^{W, \mathcal{N}} W.L(s)(G(s)) \} \\
\mathbf{E}_W(G)(s)(\tau) & := \{ (e_1, e_2) \mid \forall (h_1, h_2) \in W.H(s)(G(s)). \forall h_1^F, h_2^F. h_1 \uplus h_1^F \text{ defined} \wedge h_2 \uplus h_2^F \text{ defined} \implies \\
& \quad \exists h_1', h_2', v_1, v_2, K_1, K_2, e_1', e_2', s', \tau'. \\
& \quad \text{Case } \uparrow: (h_1 \uplus h_1^F, e_1) \hookrightarrow^\omega \wedge (h_2 \uplus h_2^F, e_2) \hookrightarrow^\omega \\
& \quad \vee \text{Case } \downarrow: (h_1 \uplus h_1^F, e_1) \hookrightarrow^* (h_1' \uplus h_1^F, v_1) \wedge (h_2 \uplus h_2^F, e_2) \hookrightarrow^* (h_2' \uplus h_2^F, v_2) \\
& \quad \quad \wedge s' \supseteq s \wedge (h_1', h_2') \in W.H(s')(G(s')) \wedge (v_1, v_2) \in \overline{G}(s')(\tau) \\
& \quad \vee \text{Case } \downarrow: (h_1 \uplus h_1^F, e_1) \hookrightarrow^* (h_1' \uplus h_1^F, K_1[e_1']) \wedge (h_2 \uplus h_2^F, e_2) \hookrightarrow^* (h_2' \uplus h_2^F, K_2[e_2']) \\
& \quad \quad \wedge s' \supseteq s \wedge (h_1', h_2') \in W.H(s')(G(s')) \wedge (e_1', e_2') \in \mathbf{S}(G(s'), G(s'))(\tau') \\
& \quad \quad \wedge \forall s'' \supseteq s'. \forall G' \supseteq G. \forall (v_1', v_2') \in \overline{G'}(s'')(\tau'). (K_1[v_1'], K_2[v_2']) \in \mathbf{E}_W(G')(s'')(\tau) \} \\
\mathbf{S}(R_f, R_v)(\tau) & := \{ (f_1 v_1, f_2 v_2) \mid \exists \tau'. (f_1, f_2) \in R_f(\tau' \rightarrow \tau) \wedge (v_1, v_2) \in \overline{R}_v(\tau') \} \cup \\
& \quad \{ (f_1 [], f_2 []) \mid \exists \tau', \sigma. \tau = \tau'[\sigma/\alpha] \wedge (f_1, f_2) \in R_f(\forall \alpha. \tau') \} \\
\mathbf{OE}_W(G)(s)(\Delta; \Gamma \vdash \tau) & := \{ (e_1, e_2) \mid \forall \delta \in \Delta \rightarrow \text{CType}. \forall \gamma_1, \gamma_2 \in \text{dom}(\Gamma) \rightarrow \text{CVal}. \\
& \quad (\forall x: \tau' \in \Gamma. (\gamma_1(x), \gamma_2(x)) \in \overline{G}(s)(\delta\tau')) \implies (\gamma_1 e_1, \gamma_2 e_2) \in \mathbf{E}_W(G)(s)(\delta\tau) \} \\
\text{inhabited}(W) & := \forall G \in \text{GK}(W). \exists s_0. (\emptyset, \emptyset) \in W.H(s_0)(G(s_0)) \\
\text{consistent}(W) & := \forall G \in \text{GK}(W). \forall s. \forall \tau. \forall (e_1, e_2) \in \mathbf{S}(W.L(s)(G(s)), G(s))(\tau). (\text{beta}(e_1), \text{beta}(e_2)) \in \mathbf{E}_W(G)(s)(\tau) \\
\text{stable}(w) & := \forall G \in \text{GK}(w\uparrow). \forall s^{\text{rf}}, s. \forall (h_1, h_2) \in w.H(s^{\text{rf}})(s)(G(s^{\text{rf}}), s). \\
& \quad \forall \hat{s}^{\text{rf}} \sqsupseteq s^{\text{rf}}. \forall (h_1^{\text{rf}}, h_2^{\text{rf}}) \in W_{\text{ref}}.H(\hat{s}^{\text{rf}})(G(\hat{s}^{\text{rf}}), s). h_1^{\text{rf}} \uplus h_1 \text{ defined} \wedge h_2^{\text{rf}} \uplus h_2 \text{ defined} \implies \\
& \quad \exists \hat{s} \sqsupseteq s. (h_1, h_2) \in w.H(\hat{s}^{\text{rf}})(\hat{s})(G(\hat{s}^{\text{rf}}), \hat{s}) \\
\Delta; \Gamma \vdash e_1 \sim_W e_2 : \tau & := \text{inhabited}(W) \wedge \text{consistent}(W) \wedge \forall G \in \text{GK}(W). \forall s. (e_1, e_2) \in \mathbf{OE}_W(G)(s)(\Delta; \Gamma \vdash \tau) \\
\Delta; \Gamma \vdash e_1 \sim_{e_2} : \tau & := \forall \mathcal{N} \in \mathbb{P}(\text{TyNam}). \mathcal{N} \text{ countably infinite} \implies \exists w. w.\mathcal{N} \subseteq \mathcal{N} \wedge \text{stable}(w) \wedge \Delta; \Gamma \vdash e_1 \sim_{w\uparrow} e_2 : \tau
\end{aligned}$$

Figure 7. Coinductive definition of local term equivalence, $\mathbf{E}_W \in \text{GK}(W) \rightarrow W.S \rightarrow \text{ERel}$, plus definitions of world inhabitation, world consistency, local world stability, and program equivalence.

Flexible Types and Value Closure. The local and global knowledges only declare which values are related at the so-called “flexible” types, CTypeF , which include type names, as well as function, universal, and reference types (Figure 2). Such value equivalences at flexible types, $R \in \text{VRelF}$, are extended to all (closed) types by the inductively-constructed value closure, $\overline{R} \in \text{VRel}$, in Figure 3, which defines the meaning of the remaining “rigid” type constructors. Note that while existentials ($\exists \alpha. \tau$) are rigid, the witness τ' for α can be a type *name*, which $W.L$ can define semantically via an arbitrary state-dependent value relation. In this way, RTSs support relationally parametric reasoning [7, 12].

Local Term Equivalence. We say that two closed terms e_1 and e_2 are *locally equivalent* at a given type τ w.r.t. a world W , a global knowledge G , and the current state of the world s —and denote this as $(e_1, e_2) \in \mathbf{E}_W(G)(s)(\tau)$ —if, when they are executed with related initial heaps (*i.e.*, satisfying the world’s invariants), one of these cases holds:

- (Case \uparrow) they both diverge; or
- (Case \downarrow) they both reduce to related values; or
- (Case \downarrow) they both reduce to related “stuck” configurations (S), such as function calls, where related function values are applied to related argument values inside locally equivalent evaluation contexts, *i.e.*, contexts which, when filled with related values, result (coinductively) in locally equivalent terms. In all cases, “related” values are drawn from the value closure of the global knowledge, \overline{G} , and we require that the final heaps also satisfy the world’s invariants.

Observe that, in cases \downarrow and \downarrow , we are permitted to advance to a future state $s' \supseteq s$. Correspondingly, in the \downarrow case, we must show that the continuations K_1 and K_2

are related in any future state $s'' \supseteq s'$ —as the function call may further advance the state of the world—and in any (pointwise) larger global knowledge. Also note that our definition bakes in a notion of “framing” (like in separation logic) by quantifying over frame heaps h_1^F and h_2^F .

Two open terms are locally equivalent (\mathbf{OE}_W) if, for all closing substitutions of related values at the appropriate types, the substituted (closed) terms are locally equivalent.

Program Equivalence. Finally, two *programs* are equivalent ($\Delta; \Gamma \vdash e_1 \sim_{e_2} : \tau$) iff there exists a local world w such that (a) w is parametric in the particular choice of names to represent its abstract types; (b) w is stable; and (c) $w\uparrow$ is inhabited and consistent, and relates e_1 and e_2 under any global knowledge $G \in \text{GK}(w\uparrow)$. A world is inhabited iff its heap invariant is satisfied by the empty heaps in some state; it is consistent iff any functions it relates do indeed behave locally equivalently when applied to \overline{G} -related arguments.

We conclude this section with a key lemma about \mathbf{E} . Given a consistent world, if the global knowledge extends the world’s local knowledge $W.L$ with some additional external knowledge \mathcal{R} , then the third case in the definition of \mathbf{E} can be restricted so that it applies only to external function calls (*i.e.*, calls to functions related by \mathcal{R} , not by $W.L$). This holds essentially because we can “inline” the equivalence proofs for any internal calls.

Lemma 1 (External call). For any W s.t. $\text{consistent}(W)$, $G \in \text{GK}(W)$ and $\mathcal{R} \in W.S \rightarrow \text{VRelF}$, we have

$$(\forall s. G(s) = W.L(s)(G(s)) \cup \mathcal{R}(s)) \implies \mathbf{E}_W(G) = \mathbf{E}_W^{\mathcal{R}}(G)$$

where the definition of $\mathbf{E}_W^{\mathcal{R}}$ is the same as \mathbf{E}_W except that $\mathbf{S}(G(s'), G(s'))$ is replaced by $\mathbf{S}(\mathcal{R}(s'), G(s'))$.

IV. STRUCTURE OF THE TRANSITIVITY PROOF

Given that $\Delta; \Gamma \vdash e_1 \sim e_2 : \tau$ and $\Delta; \Gamma \vdash e_2 \sim e_3 : \tau$, our goal is to show $\Delta; \Gamma \vdash e_1 \sim e_3 : \tau$. Unfolding the definition of our goal, we are given a countably infinite set of type names \mathcal{N} and must construct a stable local world w such that (a) $\Delta; \Gamma \vdash e_1 \sim_{w\uparrow} e_3 : \tau$ and (b) $w.N \subseteq \mathcal{N}$ (i.e., $w.L$ defines no names outside of \mathcal{N}). To do so, we split \mathcal{N} into three disjoint (and also countably infinite) pieces: \mathcal{N}_1 , \mathcal{N}_2 , and \mathcal{N}_3 . The first two pieces will be used to instantiate the assumptions regarding $e_1 \sim e_2$ and $e_2 \sim e_3$, respectively, thus yielding two stable local worlds w_1 and w_2 such that

$$\Delta; \Gamma \vdash e_1 \sim_{w_1\uparrow} e_2 : \tau \quad (1)$$

$$\text{and } \Delta; \Gamma \vdash e_2 \sim_{w_2\uparrow} e_3 : \tau \quad (2)$$

as well as $w_1.N \subseteq \mathcal{N}_1$ and $w_2.N \subseteq \mathcal{N}_2$. Keeping \mathcal{N}_1 and \mathcal{N}_2 disjoint is a matter of basic hygiene: it ensures that w_1 and w_2 , which we will be using in the construction of w , do not step on each other's toes by defining the same type name in incompatible ways. As for the names in \mathcal{N}_3 , we reserve them for a special purpose to be explained later.

At this point, the proof divides into two separate parts. In the first part, we use w_1 and w_2 to directly construct a full world W such that $\Delta; \Gamma \vdash e_1 \sim_W e_3 : \tau$ (and $W.N \subseteq \mathcal{N}$). While the proof of this part is quite subtle, it is essentially an extension of the transitivity proof for a restricted fragment of F^{μ} that we sketched in our previous paper [6], and which we present here in much greater detail. The main novelty over that previous proof is that we now deal with abstract types; reference types do not cause much of a problem.

However, the second part of the proof has all to do with references. Specifically, the world W that we create in the first part does not have the required shape of a lifted local world $w\uparrow$. Thus, in the second part, we (i) develop a theory of *weak isomorphisms* between worlds and prove that they preserve term equivalence, and (ii) construct a stable local world w such that $w\uparrow$ is weakly isomorphic to W .

V. FIRST PART: CONSTRUCTING THE FULL WORLD W

A. High-Level Explanation

As mentioned above, this first part of the proof is essentially agnostic as to whether the language/model supports mutable state. To ease the presentation, we therefore gloss over any state-related details at first; we will be more precise in Section V-B. We also write W_i as shorthand for $w_i\uparrow$.

We want to construct W such that $\Delta; \Gamma \vdash e_1 \sim_W e_3 : \tau$. The proofs of consistency of W and relatedness of (e_1, e_3) by **OE** turn out to be very similar, so let us focus on the latter here. We are given a global knowledge $G \in \text{GK}(W)$ and related substitutions γ_1 and γ_3 . These substitutions map each variable bound in Γ to related values $(v_1, v_3) \in \overline{G}(\tau')$. In order to make use of (1) and (2), we want to be able to: (i) “decompose” G into global knowledges $G_{(1)} \in \text{GK}(W_1)$ and $G_{(2)} \in \text{GK}(W_2)$ that would be suitable for instantiating

(1) and (2), and (ii) find a mediating substitution γ_2 s.t. for each $x \in \text{dom}(\Gamma)$, it is the case that $\overline{G_{(1)}}$ relates $(\gamma_1 x, \gamma_2 x)$ and $\overline{G_{(2)}}$ relates $(\gamma_2 x, \gamma_3 x)$. Formally, we want:

$$\overline{G}(\tau) \subseteq \overline{G_{(1)}}(\tau_{(1)}) \circ \overline{G_{(2)}}(\tau_{(2)}) \quad (3)$$

where \circ is ordinary relational composition. (Pretend *for now* that $\tau_{(i)} = \tau$. We will soon see why that's not good enough.)

If we have this, we can instantiate (1) and (2), thus obtaining $(\gamma_1 e_1, \gamma_2 e_2) \in \mathbf{E}(G_{(1)})(\tau_{(1)})$ and $(\gamma_2 e_2, \gamma_3 e_3) \in \mathbf{E}(G_{(2)})(\tau_{(2)})$. It thus remains for us to show $(\gamma_1 e_1, \gamma_3 e_3) \in \mathbf{E}(G)(\tau)$. In other words, we must (unsurprisingly) show some kind of transitivity property for **E**:

$$\mathbf{E}_{W_1}(G_{(1)})(\tau_{(1)}) \circ \mathbf{E}_{W_2}(G_{(2)})(\tau_{(2)}) \subseteq \mathbf{E}_W(G)(\tau) \quad (4)$$

Proving this will certainly require us to prove the analogous transitivity property for values, which is the inverse of (3):

$$\overline{G}(\tau) \supseteq \overline{G_{(1)}}(\tau_{(1)}) \circ \overline{G_{(2)}}(\tau_{(2)}) \quad (5)$$

Since global knowledges extend the corresponding local ones, this in turn means that $W.L$ must at least include the composition of $W_1.L$ and $W_2.L$:

$$W.L(G)(\tau) \supseteq W_1.L(G_{(1)})(\tau_{(1)}) \circ W_2.L(G_{(2)})(\tau_{(2)}) \quad (6)$$

In order to see what else we want to put into W , and how to define $G_{(i)}$ and $\tau_{(i)}$, let us consider proving (3) and (5). For a flexible type τ' the conjunction of (3) and (5) is:

$$G(\tau') = G_{(1)}(\tau'_{(1)}) \circ G_{(2)}(\tau'_{(2)}).$$

One simple (but inadequate) choice for $G_{(i)}$ would be to define it as the minimal global knowledge in $\text{GK}(W_i)$ —i.e., as the least fixed-point of $W_i.L$. But there is no reason to believe that for any $(v_1, v_3) \in G(\tau')$, there magically happens to be a “mediating” value v_2 such that $W_1.L$ relates (v_1, v_2) and $W_2.L$ relates (v_2, v_3) , as required by the \subseteq part of the above equation. To work this magic, we will explicitly *add* such mediating values to $G_{(1)}$ and $G_{(2)}$! Concretely, we define $G_{(1)}$ as the smallest global knowledge in $\text{GK}(W_1)$ that relates $(v_1, \mathbf{I}(\tau, v_1, v_3))$ at $\tau_{(1)}$ whenever G relates (v_1, v_3) at τ , and similarly $G_{(2)}$ as the smallest global knowledge in $\text{GK}(W_2)$ that relates $(\mathbf{I}(\tau, v_1, v_3), v_3)$ at $\tau_{(2)}$ whenever G relates (v_1, v_3) at τ .

Now, what is this magic **I**? For proving (3), it could be anything that maps to CVal . But for (5), it is crucial that each mediating value *uniquely* encodes the corresponding value pair (v_1, v_3) . We therefore require **I** to be *injective*. Since all involved sets are countably infinite, such an encoding function exists and we do not care about the particular choice—except that we will choose its range to be of *rigid* type, specifically int , for reasons we will explain shortly.

The proofs of (3) and (5) are by induction on the value closure (recall that it is constructed as a least fixed-point). The reason why we must add more to $W.L$ than just the composition in (6), and why $\tau_{(i)}$ cannot just be τ in general, has to do with abstract types. We illustrate the issue here for existential types, but the same problem arises for universals (although in a different place, namely in the proof of (4)).

Suppose $\tau_{(i)}$ were the identity and consider (5) at some type $\exists\alpha.\tau$: We would have to show that if $(\text{pack } v_1, \text{pack } v_2) \in \overline{G_{(1)}}(\exists\alpha.\tau)$ and $(\text{pack } v_2, \text{pack } v_3) \in \overline{G_{(2)}}(\exists\alpha.\tau)$, then $(\text{pack } v_1, \text{pack } v_3) \in \overline{G}(\exists\alpha.\tau)$. Unfolding the value closure, this means: Given some τ'_1, τ'_2 such that $(v_1, v_2) \in \overline{G_{(1)}}(\tau[\tau'_1/\alpha])$ and $(v_2, v_3) \in \overline{G_{(2)}}(\tau[\tau'_2/\alpha])$, we must come up with τ' such that $(v_1, v_3) \in \overline{G}(\tau[\tau'/\alpha])$. Now, if the two given representation types happen to be the same ($\tau'_1 = \tau'_2$), then we could proceed by just picking $\tau' := \tau'_1$. But of course in general τ'_1 and τ'_2 will be different!

The intuition behind our solution is quite simple: we pick τ' to be a fresh *type name*, which we use to represent the semantic composition of τ'_1 and τ'_2 . More concretely, we use a type name \mathbf{n} from \mathcal{N}_{\exists} (which we reserved for exactly this purpose) to uniquely encode τ'_1 and τ'_2 , then define \mathbf{n} 's meaning in W to be precisely $\overline{G_{(1)}}(\tau'_1) \circ \overline{G_{(2)}}(\tau'_2)$, and finally choose τ' to be \mathbf{n} . Since we don't know what τ'_1 and τ'_2 are, we simply have to encode all pairs of types this way. To pick the names, we use an injective function

$$\mathbf{A} \in \text{CType} \times \text{CType} \rightarrow \mathcal{N}_{\exists}$$

which, like \mathbf{I} , exists because all involved sets are countably infinite (and, as with \mathbf{I} , we do not care about its concrete definition). It should be clear by now what $\tau'_{(i)}$ does and that it is crucial for making the induction go through: it *decodes* τ' by traversing its structure and replacing each type name \mathbf{n} that equals $\mathbf{A}(\tau_1, \tau_2)$ —for some τ_1, τ_2 —with τ_i .

That's the intuition; the reality is a bit more complex. It turns out that, in order to prove (3) and (5), the decoding must in fact be *bijective*, but the one sketched above is not injective. For instance, $\mathbf{A}(\text{int}, \text{int})$ and int are obviously distinct types but both decode to int (by either projection). Fortunately, there is an easy way to obtain the desired bijectivity: we only encode a type pair (τ_1, τ_2) directly as a name $\mathbf{A}(\tau_1, \tau_2)$ if τ_1 and τ_2 are “sufficiently different”. If they share some structure, however, we keep the parts that are the same and only apply \mathbf{A} to the parts that are different. To take a simple example: to encode $(\text{int}, \text{bool})$, we would use the type name $\mathbf{A}(\text{int}, \text{bool})$, but to encode $(\text{int} \rightarrow \text{int}, \text{int} \rightarrow \text{bool})$, we would pick $\text{int} \rightarrow \mathbf{A}(\text{int}, \text{bool})$ instead of $\mathbf{A}(\text{int} \rightarrow \text{int}, \text{int} \rightarrow \text{bool})$.

Finally, property (4) is shown by coinduction. Recalling that \mathbf{E} comprises three cases (\uparrow , \downarrow , and \downarrow), the key here is that the case of \mathbf{E} by which the first and second terms—call them e'_1 and e'_2 —are related should match the case by which the second and third terms— e'_2 and e'_3 —are related. In other words, out of the 3×3 possible cases, 6 should never arise. As a representative example, consider the situation where e'_1 and e'_2 are related because they reduce to related values (case \downarrow), and e'_2 and e'_3 because they reduce to related function calls with related continuations (case \downarrow). By Lemma 1 we can assume that these calls are to *external* functions related by $G_{(2)}$. But this means that the function called in e'_2 must be of the form $\mathbf{I}(\tau, v_1, v_3)$, *i.e.*, not a function at all (recall

that we required \mathbf{I} to map to *rigid* values)! Hence, e'_2 gets stuck, contradicting the assumption that it reduced to a value.

The remaining possibilities (where the cases of relatedness for (e'_1, e'_2) and (e'_2, e'_3) match) are handled as follows:

Case \uparrow . Then both e'_1 and e'_3 diverge, so we are done.

Case \downarrow . Then e'_1, e'_2 , and e'_3 reduce to values v_1, v_2 , and v_3 , such that $\overline{G_{(1)}}$ relates (v_1, v_2) and $\overline{G_{(2)}}$ relates (v_2, v_3) . Thus, by (5), \overline{G} relates (v_1, v_3) and we are done.

Case \downarrow . We know that e'_1 and e'_2 reduce to related function calls in related continuations, and that the same applies to e'_2 and e'_3 . Using Lemma 1 as above, we know that all four function calls are actually stuck. Since, by determinacy, e'_2 cannot get stuck in two different ways, we have a unique function call and a unique continuation in the middle. So, it suffices to show a corresponding transitivity property for \mathbf{S} and for continuations. The one for continuations follows from (3) and the coinductive hypothesis. The one for \mathbf{S} follows from (5) and injectivity of \mathbf{I} ; when reasoning about type instantiations $(f_i[])$, we must also make use of our type encoding \mathbf{A} —dually to how we handle pack in proving (5).

B. The Gory Details

We first formalize the syntactic encoding of types. The previously motivated notion of two types being “sufficiently different” is defined as the negation of *similarity*.

Definition 1. Similarity, written \approx , is defined inductively:

$$\begin{array}{c} \mathbf{n} \notin \mathcal{N}_{\exists} \quad \tau \sim \sigma \quad \tau \sim \sigma \quad (\star = \mu, \forall, \exists) \\ \mathbf{n} \approx \mathbf{n} \quad \alpha \approx \alpha \quad \tau_{\text{base}} \approx \tau_{\text{base}} \quad \text{ref } \tau \approx \text{ref } \sigma \quad \star\alpha.\tau \approx \star\alpha.\sigma \\ \tau \sim \sigma \quad \tau' \sim \sigma' \quad (\star = \times, +, \rightarrow) \quad \tau, \sigma \in \text{CType} \quad \tau \approx \sigma \\ \tau \star \tau' \approx \sigma \star \sigma' \quad \tau \sim \sigma \quad \tau \sim \sigma \end{array}$$

Two closed dissimilar types are encoded as a type name from \mathcal{N}_{\exists} using a bijection

$$\mathbf{A} \in \{(\tau_1, \tau_2) \in \text{CType} \times \text{CType} \mid \tau_1 \not\approx \tau_2\} \rightarrow \mathcal{N}_{\exists}.$$

The encoding of two arbitrary closed types is a natural lifting of \mathbf{A} . Instead of defining it explicitly, we find it simpler to define the decoding and then state the existence of a corresponding encoding.

Definition 2. We recursively define *decoding projections* $(-)_{(i)} \in \text{Type} \rightarrow \text{Type}$ for $i = 1, 2$ as follows. Note that if τ is closed, then so is $\tau_{(i)}$.

$$\begin{array}{l} \mathbf{n}_{(i)} := \begin{cases} \tau_i & \text{if } \mathbf{n} = \mathbf{A}(\tau_1, \tau_2) \text{ for some } \tau_1, \tau_2 \\ \mathbf{n} & \text{otherwise, i.e., } \mathbf{n} \notin \mathcal{N}_{\exists} \end{cases} \\ \alpha_{(i)} := \alpha \quad \tau_{\text{base}(i)} := \tau_{\text{base}} \quad (\text{ref } \tau)_{(i)} := \text{ref } \tau_{(i)} \\ (\star\alpha.\tau)_{(i)} := \star\alpha.\tau_{(i)} \quad (\text{where } \star = \mu, \forall, \exists) \\ (\tau \star \tau')_{(i)} := \tau_{(i)} \star \tau'_{(i)} \quad (\text{where } \star = \times, +, \rightarrow) \end{array}$$

The encoding of two arbitrary closed types is now implicit in the surjectivity part of the next lemma.

Lemma 2. $\langle(-)_{(1)}, (-)_{(2)}\rangle \in \text{CType} \rightarrow \text{CType} \times \text{CType}$ is bijective.

Proof: Injectivity (generalized to open types) can be easily shown by induction on types using two sub-lemmas:

- (a) $\forall \tau. \tau_{(1)} \sim \tau_{(2)}$, which is shown by straightforward induction on τ .
- (b) $\forall \tau. \tau_{(1)} \approx \tau_{(2)} \iff \tau \notin \mathcal{N}_\exists$, which is shown by case analysis on τ using (a).

Surjectivity is generalized as follows:

$$\forall \tau_1, \tau_2. \tau_1 \sim \tau_2 \implies \exists \tau. \tau_{(1)} = \tau_1 \wedge \tau_{(2)} = \tau_2$$

(Note that if τ_1 and τ_2 are closed, the premise holds trivially.) This property can be proven by induction on τ_1 . In each case we ask if $\tau_1 \approx \tau_2$ holds. If it does, we use the inductive hypothesis; otherwise, the premise implies that τ_1, τ_2 are closed and thus we can pick τ to be $\mathbf{A}(\tau_1, \tau_2)$. ■

Now, let us define the *decomposition* of value relations. This will eventually be used to decompose a global knowledge that respects the yet-to-be-defined W into one that respects W_1 and one that respects W_2 .

Definition 3. Given $R \in \text{VRelF}$, we define $R_{\{i\}} \in \text{VRelF}$ and $R_{(i)}^* \in W_i.S \rightarrow \text{VRelF}$ (for $i = 1, 2$) as follows.

$$\begin{aligned} R_{\{1\}} &:= \{(\tau_{(1)}, v_1, \mathbf{I}(\tau, v_1, v_3)) \mid \tau \in \text{CTyF}_{\text{ref}}^{\setminus \mathcal{N}} \wedge (v_1, v_3) \in R(\tau)\} \\ R_{\{2\}} &:= \{(\tau_{(2)}, \mathbf{I}(\tau, v_1, v_3), v_3) \mid \tau \in \text{CTyF}_{\text{ref}}^{\setminus \mathcal{N}} \wedge (v_1, v_3) \in R(\tau)\} \\ R_{(i)}^*(s) &:= [W_i.L(s)]_{(R_{\{i\}})}^* \end{aligned}$$

Here,

- $\text{CTyF}_{\text{ref}}^{\setminus \mathcal{N}}$ means $\text{CTyF} \setminus (\mathcal{N} \cup \{\text{ref } \tau \in \text{CTyF}\})$,
- $[F]_R^*$ denotes the least fixed-point of the monotone function $F(-) \cup R$, thus making $R_{(i)}^*$ the smallest global knowledge that both respects W_i and contains $R_{\{i\}}$, and
- \mathbf{I} is an injective function in $\text{CTyF} \times \text{CVal} \times \text{CVal} \rightarrow \mathcal{N}$.

With the help of this, we can now construct the world W in Figure 8. Its well-formedness is easy to check. Note that, although W only actually defines names from \mathcal{N}_\exists , we declare that it owns the larger set \mathcal{N} in order to reduce the set of global knowledges that we have to worry about.

Next, some notation for decomposing a global knowledge.

Definition 4. Given $G \in \text{GK}(W)$, we define

$$G_{(1)}^{s_2} \in W_1.S \rightarrow \text{VRelF} \text{ for } s_2 \in W_2.S,$$

and $G_{(2)}^{s_1} \in W_2.S \rightarrow \text{VRelF}$ for $s_1 \in W_1.S$ as follows:

$$G_{(1)}^{s_2}(s) := G(s, s_2)_{(1)}^*(s) \quad G_{(2)}^{s_1}(s) := G(s_1, s)_{(2)}^*(s)$$

Note that if the global knowledge argument R in the definition of $W.L$ and $W.H$ is of the form $G(s_1, s_2)$ for $G \in \text{GK}(W)$, then the global knowledge passed to $W_1.L$ and $W_1.H$ is exactly $G_{(1)}^{s_2}(s_1)$ (and similarly for W_2). It remains to show that $G_{(1)}^{s_2}(s_1)$ and $G_{(2)}^{s_1}(s_2)$ are in fact valid global knowledges that respect W_1 and W_2 , respectively:

Lemma 3. $\forall G \in \text{GK}(W)$.

$$\forall s_2. G_{(1)}^{s_2} \in \text{GK}(W_1) \wedge \forall s_1. G_{(2)}^{s_1} \in \text{GK}(W_2)$$

Proof: We show the first conjunct (the second is analogous). Monotonicity of $G_{(1)}^{s_2}$ is proven by fixed-point induction using the fact that G , $(-)_\{1\}$ and $W_1.L$ are monotone. $G_{(1)}^{s_2}(s_1) \geq_{W_1.N}^{W_1.L} W_1.L(s_1)(G_{(1)}^{s_2}(s_1))$ follows from

$$\begin{aligned} W.N &:= \mathcal{N} \\ W.S &:= W_1.S \times W_2.S \\ W.\sqsubseteq &:= \{(p, p') \mid p.1 \sqsubseteq p'.1 \wedge p.2 \sqsubseteq p'.2\} \\ W.L(s_1, s_2)(R)(\tau) &:= \begin{cases} \overline{R_{(1)}^{s_2}(s_1)}(\tau_{(1)}) \circ \overline{R_{(2)}^{s_1}(s_2)}(\tau_{(2)}) & \text{if } \tau \in \mathcal{N}_\exists \\ W_1.L(s_1)(R_{(1)}^{s_2}(s_1))(\tau_{(1)}) & \text{if } \tau \notin \mathcal{N}_\exists \\ \quad \circ W_2.L(s_2)(R_{(2)}^{s_1}(s_2))(\tau_{(2)}) & \end{cases} \\ W.H(s_1, s_2)(R) &:= W_1.H(s_1)(R_{(1)}^{s_2}(s_1)) \circ W_2.H(s_2)(R_{(2)}^{s_1}(s_2)) \end{aligned}$$

Figure 8. Construction of $W \in \text{World}$.

$$G_{(1)}^{s_2}(s_1)(\tau) = W_1.L(s_1)(G_{(1)}^{s_2}(s_1))(\tau) \cup G(s_1, s_2)_{\{1\}}(\tau). \quad (\text{Note that } G(s_1, s_2)_{\{1\}}(\tau) = \emptyset \text{ if } \tau \text{ is a reference type or a name in } W_1.N = \mathcal{N}_1 \subseteq \mathcal{N}.) \quad \blacksquare$$

We now come to the main lemma of this part, namely the conjunction of properties (3) and (5) from Section V-A:

Lemma 4. $\forall G \in \text{GK}(W). \forall \tau \in \text{CTyF}. \forall s_1, s_2.$

$$\overline{G_{(1)}^{s_2}(s_1)}(\tau_{(1)}) \circ \overline{G_{(2)}^{s_1}(s_2)}(\tau_{(2)}) = \overline{G(s_1, s_2)}(\tau)$$

Proof: The \supseteq part is proven by fixed-point induction on $\overline{G(s_1, s_2)}$ (recall that $\overline{(-)}$ is defined as a least fixed-point). The \subseteq part is equivalent to

$$\begin{aligned} \overline{G_{(1)}^{s_2}(s_1)}(\tau) &\subseteq \{(v_1, v_2) \mid \forall \sigma, v_3. \\ &(\tau = \sigma_{(1)} \wedge (v_2, v_3) \in \overline{G_{(2)}^{s_1}(s_2)}(\sigma_{(2)})) \implies (v_1, v_3) \in \overline{G(s_1, s_2)}(\sigma)\} \end{aligned}$$

which is proven by (generalized) induction on $\overline{G_{(1)}^{s_2}(s_1)}$.

Base cases. In both parts, the base cases follow from

$$\forall G \in \text{GK}(W). \forall \tau \in \text{CTyF}. \forall s_1, s_2.$$

$$\overline{G_{(1)}^{s_2}(s_1)}(\tau_{(1)}) \circ \overline{G_{(2)}^{s_1}(s_2)}(\tau_{(2)}) = G(s_1, s_2)(\tau)$$

which we now show by case analysis on τ . If $\tau \in \mathcal{N}_\exists$, then $G(s_1, s_2)(\tau) = W.L(s_1, s_2)(G(s_1, s_2))(\tau) \quad (G \in \text{GK}(W))$
 $= \overline{G_{(1)}^{s_2}(s_1)}(\tau_{(1)}) \circ \overline{G_{(2)}^{s_1}(s_2)}(\tau_{(2)}) \quad (\text{def. of } W.L)$

and we are done. Otherwise ($\tau \in \text{CTyF} \setminus \mathcal{N}_\exists$), we have:

$$\begin{aligned} &\overline{G_{(1)}^{s_2}(s_1)}(\tau_{(1)}) \circ \overline{G_{(2)}^{s_1}(s_2)}(\tau_{(2)}) \\ &= \overline{G_{(1)}^{s_2}(s_1)}(\tau_{(1)}) \circ G_{(2)}^{s_1}(s_2)(\tau_{(2)}) \\ &= (W_1.L(s_1)(G_{(1)}^{s_2}(s_1))(\tau_{(1)}) \cup G(s_1, s_2)_{\{1\}}(\tau_{(1)})) \circ \\ &\quad (W_2.L(s_2)(G_{(2)}^{s_1}(s_2))(\tau_{(2)}) \cup G(s_1, s_2)_{\{2\}}(\tau_{(2)})) \quad (7) \end{aligned}$$

Now, if τ is a reference type or a name from $\mathcal{N}_1 \uplus \mathcal{N}_2$, we finish by rewriting (7) as follows:

$$\begin{aligned} &= (W_1.L(s_1)(G_{(1)}^{s_2}(s_1))(\tau_{(1)}) \cup \emptyset) \circ \\ &\quad (W_2.L(s_2)(G_{(2)}^{s_1}(s_2))(\tau_{(2)}) \cup \emptyset) \quad (\text{def. of } (-)_{\{i\}}) \\ &= W.L(s_1, s_2)(G(s_1, s_2))(\tau) \quad (\text{def. of } W.L) \\ &= G(s_1, s_2)(\tau) \quad (G \in \text{GK}(W)) \end{aligned}$$

Otherwise ($\tau \in \text{CTyF}_{\text{ref}}^{\setminus \mathcal{N}}$), we continue by distributing \circ over \cup in (7):

$$\begin{aligned} &(W_1.L(s_1)(G_{(1)}^{s_2}(s_1))(\tau_{(1)}) \circ W_2.L(s_2)(G_{(2)}^{s_1}(s_2))(\tau_{(2)})) \\ &\cup (G(s_1, s_2)_{\{1\}}(\tau_{(1)}) \circ G(s_1, s_2)_{\{2\}}(\tau_{(2)})) \\ &\cup (W_1.L(s_1)(G_{(1)}^{s_2}(s_1))(\tau_{(1)}) \circ G(s_1, s_2)_{\{2\}}(\tau_{(2)})) \\ &\cup (G(s_1, s_2)_{\{1\}}(\tau_{(1)}) \circ W_2.L(s_2)(G_{(2)}^{s_1}(s_2))(\tau_{(2)})) \quad (8) \end{aligned}$$

The first disjunct equals $W.L(s_1, s_2)(G(s_1, s_2))(\tau)$ by construction of W ; the second equals $G(s_1, s_2)(\tau)$ by construction of $(-)_\{i\}$, injectivity of \mathbf{I} and the injectivity part of Lemma 2; and the third and the fourth are empty by construction of $(-)_\{i\}$ and the fact that

$W_i.L(s)(R) \in \text{Know}(W_i.N)$ for any s, R . So, (8) becomes $W.L(s_1, s_2)(G(s_1, s_2))(\tau) \cup G(s_1, s_2)(\tau)$. As $G \in \text{GK}(W)$, the second disjunct contains the first, and we are done.

Inductive cases. In both parts, the inductive cases boil down to showing that for any $R_1, R_2, R \in \text{VRelF}$ and $S_1, S_2 \in \text{VRel}$ and $\tau \notin \text{CTypeF}$ the equation

$$F_{R_1}(S_1)(\tau_{(1)}) \circ F_{R_2}(S_2)(\tau_{(2)}) = F_R(S_1 \bullet S_2)(\tau)$$

holds, where $F_R \in \text{VRel} \rightarrow \text{VRel}$ denotes the monotone generating function of \bar{R} (i.e., the function of which \bar{R} is least fixed-point), and $(S_1 \bullet S_2)(\tau) := S_1(\tau_{(1)}) \circ S_2(\tau_{(2)})$. This is straightforward to show by case analysis on τ . The only really interesting case is for existential types, where (in one direction) we are given two witness types τ_1 and τ_2 and then apply Lemma 2 to find a witness type τ satisfying $\tau_{(1)} = \tau_1$ and $\tau_{(2)} = \tau_2$. ■

Finally, we can prove transitivity of \mathbf{E} and then the original goal of this first part.

Lemma 5. $\forall G \in \text{GK}(W). \forall \tau \in \text{CType}. \forall s_1, s_2.$

$$\mathbf{E}_{W_1}(G^{s_2}_{(1)})(s_1)(\tau_{(1)}) \circ \mathbf{E}_{W_2}(G^{s_1}_{(2)})(s_2)(\tau_{(2)}) \subseteq \mathbf{E}_W(G)(s_1, s_2)(\tau)$$

Proof: By coinduction, following the sketch in Section V-A (and choosing the middle frame heap, h_2^F , to be empty). ■

Lemma 6. $\Delta; \Gamma \vdash e_1 \sim_W e_3 : \tau$

Proof: Inhabitation of W follows easily from that of W_1 and W_2 and the construction of $W.H$. The proofs of consistency and relatedness of (e_1, e_3) by \mathbf{OE}_W are very similar and straightforward, using Lemmas 4 and 5. ■

VI. SECOND PART:

CONSTRUCTING THE CORRESPONDING LOCAL WORLD w

We now come to the second part of our transitivity proof. Conceptually it is quite simple, but the formal details are very subtle. Recall that we basically want to create a world that relates the same things as W from the previous section, but has the shape of a lifted world, i.e., has the form $w\uparrow$. By definition, the state space of a lifted world is of the form $W_{\text{ref}}.S \times \dots$, and thus cannot be the same as W 's state space $(W_{\text{ref}}.S \times w_1.S) \times (W_{\text{ref}}.S \times w_2.S)$. Recall further that a world's local knowledge and heap relation are state-dependent. So, in order to characterize what it means for two worlds with different state spaces to “relate the same things”, we need to introduce a notion of *world isomorphism*.

A. World Isomorphisms

Roughly, two (full) worlds W_a, W_b are isomorphic iff they declare the same type names, and each state of W_a corresponds to a state of W_b (and vice versa) such that the the same values and heaps are related at corresponding states. Different kinds of isomorphism arise depending on what counts as a correspondence. For our purpose, a one-to-one correspondence is too strong. If, say, W_b contains an “inconsistent” state (i.e., a state at which W_b 's heap relation is empty), then we should not have to worry about finding a

similarly irrelevant state in W_a . So, instead of a full one-to-one correspondence, we use a *partial* one, wherein a state s in one world is permitted to have no correspondent in the other iff s is inconsistent. This plays a crucial role in our transitivity proof, as we will see in a moment.

Definition 5. For any $W_a, W_b \in \text{World}$, a pair of functions $\phi \in W_a.S \rightarrow \mathbb{P}(W_b.S)$ and $\psi \in W_b.S \rightarrow \mathbb{P}(W_a.S)$ form a *weak isomorphism*, written $\phi : W_a \cong W_b : \psi$, if:

- (1) $W_a.N = W_b.N$
- (2a) $\forall s_a, s'_a. \forall s_b \in \phi(s_a), s'_b \in \phi(s'_a). s_a \sqsubseteq s'_a \implies s_b \sqsubseteq s'_b$
- (3a) $\forall s_a. \forall s_b \in \phi(s_a). W_a.L(s_a) = W_b.L(s_b)$
- (4a) $\forall s_a. \forall G \in \text{GK}(W_a). W_b.H(s_a)(G(s_a)) \subseteq \bigcup_{s_b \in \phi(s_a)} W_b.H(s_b)(G(s_a))$
- (5a) $\forall s_a. \forall s_b \in \phi(s_a). \forall s'_a \in \psi(s_b). s_a \sqsubseteq s'_a$
- (2b)–(5b) symmetric to (2a)–(5a)

Note that full worlds and weak morphisms—i.e., ϕ satisfying (1), (2a), (3a), and (4a)—form a category.

Theorem 7. If $\phi : W_a \cong W_b : \psi$, then: $\forall \Delta, \Gamma, \tau, e_1, e_2.$

$$\Delta; \Gamma \vdash e_1 \sim_{W_a} e_2 : \tau \iff \Delta; \Gamma \vdash e_1 \sim_{W_b} e_2 : \tau$$

Proof: See Appendix A. ■

B. Defining w

Recall that lifting a local world means linking it with the shared world W_{ref} , which provides the meaning of reference types. Accordingly, the to-be-constructed local world w 's knowledge must not relate anything at reference types, and, in order for $w\uparrow$ to be isomorphic to W , must correspond to $W.L$ at all other types. This is easy to achieve by just choosing w 's state space to be the same as W 's and then defining $w.L(s^{\text{rf}})(s)$ to be $W.L(s)$ for non-reference types. Regarding reference types, we have to satisfy (by the definition of lifting):

$$W_{\text{ref}}.L(s^{\text{rf}})(R)(\text{ref } \tau) = W.L(s)(R)(\text{ref } \tau)$$

This is problematic. Note that s really has the form $((s_1^{\text{rf}}, s_1^{\text{lc}}), (s_2^{\text{rf}}, s_2^{\text{lc}}))$ (we will later omit the inner parentheses for convenience), with $s_1^{\text{rf}}, s_2^{\text{rf}}$ being states of W_{ref} , and $s_1^{\text{lc}}, s_2^{\text{lc}}$ being states of w_1, w_2 , respectively. Unfolding the definition of $W.L$, the above equation is equivalent to

$$W_{\text{ref}}.L(s^{\text{rf}})(R)(\text{ref } \tau) = W_{\text{ref}}.L(s_1^{\text{rf}})(R_{(1)}^*(s_1^{\text{rf}}, s_1^{\text{lc}}))(\text{ref } \tau_{(1)}) \circ W_{\text{ref}}.L(s_2^{\text{rf}})(R_{(2)}^*(s_2^{\text{rf}}, s_2^{\text{lc}}))(\text{ref } \tau_{(2)})$$

which in turn reduces to $s^{\text{rf}} = s_1^{\text{rf}} \bullet s_2^{\text{rf}}$, where

$$s_1^{\text{rf}} \bullet s_2^{\text{rf}} := \{(\tau, \ell_1, \ell_3) \mid \exists \ell_2. (\tau_{(1)}, \ell_1, \ell_2) \in s_1^{\text{rf}} \wedge (\tau_{(2)}, \ell_2, \ell_3) \in s_2^{\text{rf}}\}.$$

This clearly cannot be true in general as all three states may be arbitrary. Remember, however, that we do not have to worry about inconsistent states! So the solution is easy: in the states where the equation holds—i.e., where s^{rf} and s are *coherent*—we are fine; we just need to make sure that in any other case the heap relation is empty. And since w 's heap relation may depend on the shared state s^{rf} , this can be easily done. Accordingly, the $w\uparrow$ state corresponding to

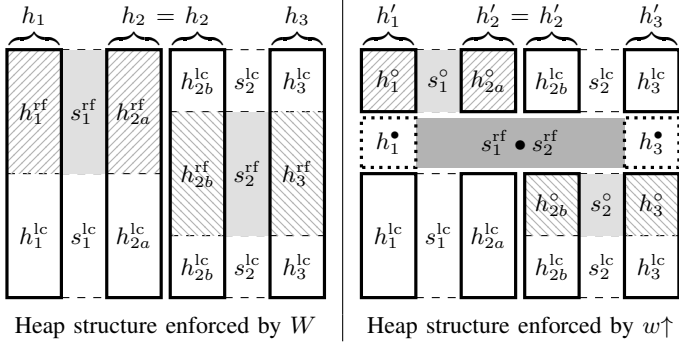


Figure 9. Construction of a stable local world $w \in \text{LWorld}$ such that $w\uparrow$ is weakly isomorphic to W .

s will be $(s_1^{\text{rf}} \bullet s_2^{\text{rf}}, s)$, and the W state corresponding to (s^{rf}, s) will be s —but only if s^{rf} happens to be $s_1^{\text{rf}} \bullet s_2^{\text{rf}}$.

What should $w.H$ relate when $s^{\text{rf}} = s_1^{\text{rf}} \bullet s_2^{\text{rf}}$ does hold? For such states, we want the following equation to be true (ignoring the global knowledge parameter to avoid clutter):

$$w\uparrow.H(s^{\text{rf}}, s) = W.H(s) \quad (9)$$

Let us look at what we know about heaps (h_1, h_3) related by $W.H(s)$. First, by construction of W , there is some h_2 mediating between $w_1\uparrow$ and $w_2\uparrow$. By definition of lifting, h_1 and h_2 can be split between $W_{\text{ref}}.H(s_1^{\text{rf}})$ and $w_1.H(s_1^{\text{rf}})(s_1^{\text{lc}})$, and similarly h_2 and h_3 can be split between $W_{\text{ref}}.H(s_2^{\text{rf}})$ and $w_2.H(s_2^{\text{rf}})(s_2^{\text{lc}})$. Of course, in general the two splits of h_2 may be arbitrarily different. This situation is depicted in the first diagram of Figure 9.

Note that $w\uparrow.H(s^{\text{rf}}, s)$ in (9) unfolds to $W_{\text{ref}}.H(s^{\text{rf}}) \otimes w.H(s^{\text{rf}})(s)$. So basically all we have to do is to define $w.H(s^{\text{rf}})(s)$ to be the “septraction” [13] of $W_{\text{ref}}.H(s^{\text{rf}})$ from $W.H(s)$. The way we do this is essentially by describing, in the definition of $w.H$, the situation from the figure but leaving out the pieces related by $W_{\text{ref}}.H(s^{\text{rf}})$. This is shown in the second diagram of Figure 9: $w.H$ relates heaps h'_1, h'_3 iff $h'_i = h_i^{\text{lc}} \uplus h_i^{\circ}$, where h_i° is the sub-heap of h_i^{rf} not covered by $s^{\text{rf}} = s_1^{\text{rf}} \bullet s_2^{\text{rf}}$. The missing pieces, h_1° and h_3° , are then going to be related by $W_{\text{ref}}.H(s^{\text{rf}})$ when w is lifted.

Formally, $w.H$ is defined as shown on the right in Figure 9, together with the other components of w . As explained, it is empty whenever s^{rf} is not compatible with s_1^{rf} and s_2^{rf} . The sub-heap h_1° (and similarly h_3°) is characterized by saying that it is related by W_{ref} to a sub-heap h_{2a}° of h_{2a}^{rf} at the state obtained by essentially subtracting those parts from s_1^{rf} that are involved in $s_1^{\text{rf}} \bullet s_2^{\text{rf}}$.

C. Showing w 's Stability

The well-formedness of w is fairly easy to check, but proving w stable is non-trivial (because $w.H$'s dependency on s^{rf} is non-trivial). Recall that stability is crucial for soundness, as it ensures that a local world's dependency on the shared state is compatible with any changes to that state.

Definition 6. For $G \in \text{GK}(w\uparrow)$, we define $\overleftarrow{G} \in W.S \rightarrow \text{VRelF}$ as follows:

$$\overleftarrow{G}(s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}}) := G(s_1^{\text{rf}} \bullet s_2^{\text{rf}}, (s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}}))$$

$w.N := W.N$ $w.S := W.S$ $w.\sqsubseteq := W.\sqsubseteq$
 $w.L(s^{\text{rf}})(s)(R)(\tau) := \{(v_1, v_3) \in W.L(s)(R)(\tau) \mid \tau \neq \text{ref}(-)\}$
 $w.H(s^{\text{rf}})(s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}})(R) := \{(h'_1, h'_3) \mid s^{\text{rf}} = s_1^{\text{rf}} \bullet s_2^{\text{rf}} \wedge \exists h_1^{\circ}, h_1^{\text{lc}}, h_{2a}^{\circ}, h_{2a}^{\text{lc}}, h_{2b}^{\circ}, h_{2b}^{\text{lc}}, h_3^{\circ}, h_3^{\text{lc}}. h'_1 = h_1^{\circ} \uplus h_1^{\text{lc}} \wedge h_{2a}^{\circ} \uplus h_{2a}^{\text{lc}} = h_{2b}^{\circ} \uplus h_{2b}^{\text{lc}} \wedge h'_3 = h_3^{\circ} \uplus h_3^{\text{lc}} \wedge \text{dom}(h_{2a}^{\text{lc}}) \cap \text{dom}_{[2]}(s_1^{\text{rf}}) = \text{dom}(h_{2b}^{\text{lc}}) \cap \text{dom}_{[1]}(s_2^{\text{rf}}) = \emptyset \wedge (h_1^{\circ}, h_{2a}^{\circ}) \in W_{\text{ref}}.H(s_1^{\circ})(R_{(1)}^*(s_1^{\text{rf}}, s_1^{\text{lc}})) \wedge (h_{2b}^{\circ}, h_3^{\circ}) \in W_{\text{ref}}.H(s_2^{\circ})(R_{(2)}^*(s_2^{\text{rf}}, s_2^{\text{lc}})) \wedge (h_1^{\text{lc}}, h_{2a}^{\text{lc}}) \in w_1.H(s_1^{\text{rf}})(s_1^{\text{lc}})(R_{(1)}^*(s_1^{\text{rf}}, s_1^{\text{lc}})) \wedge (h_{2b}^{\text{lc}}, h_3^{\text{lc}}) \in w_2.H(s_2^{\text{rf}})(s_2^{\text{lc}})(R_{(2)}^*(s_2^{\text{rf}}, s_2^{\text{lc}}))\}$
where $s_i^{\circ} = \{(\tau, \ell_1, \ell_2) \in s_i^{\text{rf}} \mid \ell_i \notin \text{dom}_{[i]}(s_1^{\text{rf}} \bullet s_2^{\text{rf}})\}$ for $i=1,2$

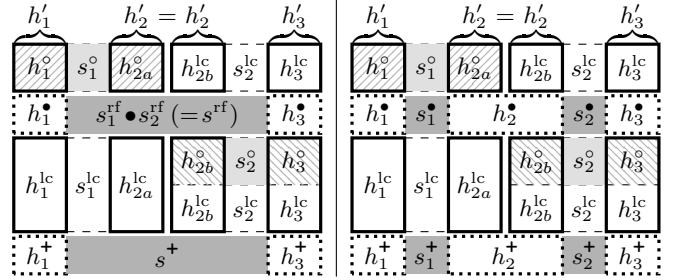


Figure 10. “Horizontal” decomposition of s^{rf} in proof of $\text{stable}(w)$.

Note that the fact that $s_1^{\text{rf}} \bullet s_2^{\text{rf}}$ is a valid W_{ref} state (i.e., a partial bijection) relies on the injectivity part of Lemma 2.

Lemma 8. $\forall G \in \text{GK}(w\uparrow). \overleftarrow{G} \in \text{GK}(W)$

Lemma 9. $\text{stable}(w)$

Proof: Suppose that $G \in \text{GK}(w\uparrow)$, $s = (s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}})$, $\hat{s}^{\text{rf}} \sqsupseteq s^{\text{rf}}$, defined $(h'_1 \uplus \hat{h}_1^{\circ})$, defined $(h'_3 \uplus \hat{h}_3^{\circ})$,

$$(h'_1, h'_3) \in w.H(s^{\text{rf}})(s)(G(s^{\text{rf}}, s)) \quad (10)$$

$$\text{and } (\hat{h}_1^{\circ}, \hat{h}_3^{\circ}) \in W_{\text{ref}}.H(\hat{s}^{\text{rf}})(G(\hat{s}^{\text{rf}}, s)). \quad (11)$$

Our goal is to find $\hat{s} = (\hat{s}_1^{\text{rf}}, \hat{s}_1^{\text{lc}}, \hat{s}_2^{\text{rf}}, \hat{s}_2^{\text{lc}}) \sqsupseteq s$ such that $(h'_1, h'_3) \in w.H(\hat{s}^{\text{rf}})(\hat{s})(G(\hat{s}^{\text{rf}}, \hat{s}))$. The main idea is to use $\text{stable}(w_i)$ to obtain \hat{s}_i^{lc} for $i = 1, 2$.

In order to do so, we must first construct states and heaps needed for instantiation. From (10) we know $s^{\text{rf}} = s_1^{\text{rf}} \bullet s_2^{\text{rf}}$ and that h'_1, h'_3 are structured as depicted in the diagram of Figure 10. From $\hat{s}^{\text{rf}} \sqsupseteq s^{\text{rf}}$ we know $\hat{s}^{\text{rf}} = s^{\text{rf}} \uplus s^+$ for some s^+ . Thus by (11), $\hat{h}_1^{\circ}, \hat{h}_3^{\circ}$ can be split as $\hat{h}_i^{\circ} = h_i^{\circ} \uplus h_i^+$ such that

$$(h_1^{\circ}, h_3^{\circ}) \in W_{\text{ref}}.H(s^{\text{rf}})(G(s^{\text{rf}}, s)) \quad (12)$$

$$\text{and } (h_1^+, h_3^+) \in W_{\text{ref}}.H(s^+)(G(s^+, s)), \quad (13)$$

as depicted in the left part of the diagram in Figure 10.

We will now “horizontally” decompose s^+ and s^{rf} , as shown in the right part of Figure 10. Since $s_i^{\text{rf}} \sqsupseteq s_i^{\circ}$, there is s_i^{\bullet} such that $s_i^{\text{rf}} = s_i^{\circ} \uplus s_i^{\bullet}$; consequently we have $\text{dom}_{[2]}(s_1^{\bullet}) = \text{dom}_{[1]}(s_2^{\bullet})$ and $s_1^{\bullet} \bullet s_2^{\bullet} = s_1^{\text{rf}} \bullet s_2^{\text{rf}} = s^{\text{rf}}$. To decompose s^+ , we choose a set of fresh locations L (of appropriate size) for the middle, i.e., we define s_i^+ such that $s_1^+ \bullet s_2^+ = s^+$ and $\text{dom}_{[2]}(s_1^+) = \text{dom}_{[1]}(s_2^+) = L$. We can then define \hat{s}_i^{rf} (used to instantiate $\text{stable}(w_i)$) as $\hat{s}_i^{\text{rf}} := s_i^{\circ} \uplus s_i^{\bullet} \uplus s_i^+$, so we have $\hat{s}_1^{\text{rf}} \bullet \hat{s}_2^{\text{rf}} = s^{\text{rf}} \uplus s^+ = \hat{s}^{\text{rf}}$.

The mediating heaps h_2^+ (with domain L) and h_2° are

constructed as follows. Since $\hat{s}_i^{\text{rf}} \supseteq s_i^{\text{rf}}$, we know

$$\overleftarrow{G}(\hat{s}_1^{\text{rf}}, s_1^{\text{lc}}, \hat{s}_2^{\text{rf}}, s_2^{\text{lc}}) = G(\hat{s}^{\text{rf}}, (s_1^{\text{rf}}, s_1^{\text{lc}}, \hat{s}_2^{\text{rf}}, s_2^{\text{lc}})) \supseteq G(\hat{s}^{\text{rf}}, s) \quad (14)$$

by monotonicity of G . From (12), (13), (14), monotonicity of $W_{\text{ref}}.H$, and Lemmas 8 and 4, we can then find mediating values to construct h_2^* and h_2^+ satisfying (for $\star \in \{\bullet, +\}$)

$$(h_1^*, h_2^*) \in W_{\text{ref}}.H(s_1^*)(\overleftarrow{G}_{(1)}^{(s_2^{\text{rf}}, s_2^{\text{lc}})}(\hat{s}_1^{\text{rf}}, s_1^{\text{lc}})) \quad (15)$$

$$\text{and } (h_2^*, h_3^*) \in W_{\text{ref}}.H(s_2^*)(\overleftarrow{G}_{(2)}^{(s_1^{\text{rf}}, s_1^{\text{lc}})}(\hat{s}_2^{\text{rf}}, s_2^{\text{lc}})). \quad (16)$$

We will now prepare to instantiate $\text{stable}(w_i)$, starting with $\text{stable}(w_1)$. First, observe that $\overleftarrow{G}_{(1)}^{(s_2^{\text{rf}}, s_2^{\text{lc}})} \in \text{GK}(w_1\uparrow)$, thanks to Lemmas 8 and 3. Next, by monotonicity of G , we have that (for $\star \in \{\hat{s}_1^{\text{rf}}, s_1^{\text{rf}}\}$)

$$\overleftarrow{G}_{(1)}^{(s_2^{\text{rf}}, s_2^{\text{lc}})}(\star, s_1^{\text{lc}}) = \overleftarrow{G}(\star, s_1^{\text{lc}}, \hat{s}_2^{\text{rf}}, s_2^{\text{lc}})_{(1)}^* \supseteq G(s^{\text{rf}}, s)_{(1)}^*$$

and thus (10), along with monotonicity of $W_{\text{ref}}.H$ and $w_1.H$ and the definition of $w.H$ in Figure 9, gives us:

$$(h_1^{\circ}, h_{2a}^{\circ}) \in W_{\text{ref}}.H(s_1^{\circ})(\overleftarrow{G}_{(1)}^{(s_2^{\text{rf}}, s_2^{\text{lc}})}(\hat{s}_1^{\text{rf}}, s_1^{\text{lc}})) \quad (17)$$

$$\text{and } (h_1^{\text{lc}}, h_{2a}^{\text{lc}}) \in w_1.H(s_1^{\text{rf}})(s_1^{\text{lc}})(\overleftarrow{G}_{(1)}^{(s_2^{\text{rf}}, s_2^{\text{lc}})}(\hat{s}_1^{\text{rf}}, s_1^{\text{lc}})). \quad (18)$$

Thus, by (15), (17), (18), and $\hat{s}_1^{\text{rf}} \supseteq s_1^{\text{rf}}$, we can instantiate $\text{stable}(w_1)$, yielding $\hat{s}_1^{\text{lc}} \supseteq s_1^{\text{lc}}$ such that

$$(h_1^{\text{lc}}, h_{2a}^{\text{lc}}) \in w_1.H(\hat{s}_1^{\text{rf}})(\hat{s}_1^{\text{lc}})(\overleftarrow{G}_{(1)}^{(s_2^{\text{rf}}, s_2^{\text{lc}})}(\hat{s}_1^{\text{rf}}, \hat{s}_1^{\text{lc}})). \quad (19)$$

In a similar manner, $\text{stable}(w_2)$ yields $\hat{s}_2^{\text{lc}} \supseteq s_2^{\text{lc}}$ such that

$$(h_{2b}^{\text{lc}}, h_3^{\text{lc}}) \in w_2.H(\hat{s}_2^{\text{rf}})(\hat{s}_2^{\text{lc}})(\overleftarrow{G}_{(2)}^{(s_1^{\text{rf}}, s_1^{\text{lc}})}(\hat{s}_2^{\text{rf}}, \hat{s}_2^{\text{lc}})). \quad (20)$$

Let $\hat{s} := (\hat{s}_1^{\text{rf}}, \hat{s}_1^{\text{lc}}, \hat{s}_2^{\text{rf}}, \hat{s}_2^{\text{lc}})$. Finally, by monotonicity of G , $W_{\text{ref}}.H$, and $w_i.H$, and by definition of $w.H$, we get $(h_1', h_3') \in w.H(\hat{s}^{\text{rf}})(\hat{s})(\overleftarrow{G}(\hat{s})) = w.H(\hat{s}^{\text{rf}})(\hat{s})(G(\hat{s}^{\text{rf}}, \hat{s}))$ from (10), (19), and (20), as desired. ■

D. Proving W and $w\uparrow$ Isomorphic

We now show that W and $w\uparrow$ are weakly isomorphic, and then put all the pieces together, arriving at our goal and thereby finishing the proof of transitivity.

Lemma 10. $\exists \phi, \psi. \phi : W \cong w\uparrow : \psi$

Proof: We define $\phi \in W.S \rightarrow \mathbb{P}(w\uparrow.S)$ and $\psi \in w\uparrow.S \rightarrow \mathbb{P}(W.S)$ as follows.

$$\begin{aligned} \phi(s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}}) &:= \{(s_1^{\text{rf}} \bullet s_2^{\text{rf}}, (s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}}))\} \\ \psi(s^{\text{rf}}, (s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}})) &:= \begin{cases} \{(s_1^{\text{rf}}, s_1^{\text{lc}}, s_2^{\text{rf}}, s_2^{\text{lc}})\} & \text{if } s^{\text{rf}} = s_1^{\text{rf}} \bullet s_2^{\text{rf}} \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

Showing that ϕ and ψ form a weak isomorphism is mostly straightforward, but for conditions (4a) and (4b) quite tedious. The key idea behind the proofs of these is the same as that behind Lemma 9: splitting the given heaps as depicted in the diagrams of Figure 9. In particular, the proof of (4b) is very similar to that of Lemma 9 in the way it uses the construction \overleftarrow{G} and Lemma 8. ■

Theorem 11 (Transitivity). $\Delta; \Gamma \vdash e_1 \sim e_3 : \tau$

Proof: We have $\Delta; \Gamma \vdash e_1 \sim_{w\uparrow} e_3 : \tau$ by Lemmas 6 and 10 and Theorem 7. The result then follows from Lemma 9. ■

VII. CONCLUSION

In this paper, we have focused on establishing transitive composability of RTSs in a traditional single-language setting—because it is already challenging enough!—but our ultimate goal is to provide a foundation for *inter-language* reasoning. We believe strongly that RTSs should be generalizable to inter-language reasoning because, like SKLRs, RTSs are inherently “semantic” (or “relational”), always quantifying over “related” terms/values without assuming that the related entities share a common syntax. As a starting point, we believe it should be possible (straightforward, even) to adapt Hur and Dreyer’s SKLR relating ML and assembly programs [5] to a formulation using RTSs instead.

As for generalizing our RTS transitivity proof to the inter-language setting, we are quite optimistic. The transitivity proof is *direct*—*i.e.*, it does not exploit contextual equivalence internally—and the first part of the proof makes few assumptions about the “middle” language (from which e_2 is drawn) except that it is capable of encoding the **I** function from Section V-A. The key challenge will be in adapting the second part of the proof to the case where the W_{ref} ’s in the proofs of $e_1 \sim e_2$ and $e_2 \sim e_3$ are of different shapes (because the languages involved have different memory models).

REFERENCES

- [1] A. Ahmed, “Step-indexed syntactic logical relations for recursive and quantified types,” in *ESOP*, 2006.
- [2] A. Ahmed, D. Dreyer, and A. Rossberg, “State-dependent representation independence,” in *POPL*, 2009.
- [3] N. Benton and C.-K. Hur, “Biorthogonality, step-indexing and compiler correctness,” in *ICFP*, 2009.
- [4] D. Dreyer, G. Neis, and L. Birkedal, “The impact of higher-order state and control effects on local relational reasoning,” in *ICFP*, 2010.
- [5] C.-K. Hur and D. Dreyer, “A Kripke logical relation between ML and assembly,” in *POPL*, 2011.
- [6] C.-K. Hur, D. Dreyer, G. Neis, and V. Vafeiadis, “The marriage of bisimulations and Kripke logical relations,” in *POPL*, 2012.
- [7] J. C. Reynolds, “Types, abstraction, and parametric polymorphism,” *Information Processing*, 1983.
- [8] D. Sangiorgi, N. Kobayashi, and E. Sumii, “Environmental bisimulations for higher-order languages,” *TOPLAS*, vol. 33, no. 1, pp. 1–69, Jan. 2011.
- [9] K. Støvring and S. Lassen, “A complete, co-inductive syntactic theory of sequential control and state,” in *POPL*, 2007.
- [10] E. Sumii, “A complete characterization of observational equivalence in polymorphic λ -calculus with general references,” in *CSL*, 2009.
- [11] E. Sumii, “A bisimulation-like proof method for contextual properties in untyped λ -calculus with references and deallocation,” *TCS*, vol. 411, no. 51–52, pp. 4358–4378, Dec. 2011.
- [12] E. Sumii and B. Pierce, “A bisimulation for type abstraction and recursion,” *JACM*, vol. 54, no. 5, pp. 1–43, Oct. 2007.
- [13] V. Vafeiadis and M. Parkinson, “A marriage of rely/guarantee and separation logic,” in *CONCUR*, 2007.

A. Proof of Isomorphism Theorem

Weak Isomorphisms. We define weak morphisms and isomorphisms.

Definition 7. Assuming $W_1, W_2 \in \text{World}$, then a function $\phi \in W_1.S \rightarrow \mathbb{P}(W_2.S)$ is a *weak morphism* from W_1 to W_2 , written $\phi : W_1 \rightarrow W_2$, if:

- 1) $W_1.N = W_2.N$
- 2) $\forall s_1, s'_1. \forall s_2 \in \phi(s_1), s'_2 \in \phi(s'_1). s_1 \sqsubseteq s'_1 \implies s_2 \sqsubseteq s'_2$
- 3) $\forall s_1. \forall s_2 \in \phi(s_1). W_1.L(s_1) = W_2.L(s_2)$
- 4) $\forall s_1. \forall G \in \text{GK}(W_1).$

$$W_1.H(s_1)(G(s_1)) \subseteq \bigcup_{s_2 \in \phi(s_1)} W_2.H(s_2)(G(s_1))$$

Definition 8. The identity weak morphism id_W on W is defined as $\text{id}_W(s) = \{s\}$. It is obvious that id_W forms a weak morphism.

Definition 9. The composition of weak morphisms $\psi \circ \phi$ is defined as $(\psi \circ \phi)(s) = \bigcup_{s' \in \phi(s)} \psi(s')$. We will show that $\psi \circ \phi$ forms a weak morphism in Lemma 18.

Definition 10. Between two weak morphisms $\phi_1, \phi_2 : W_1 \rightarrow W_2$, we define the following preorder \sqsubseteq :

$$\phi_1 \sqsupseteq \phi_2 \quad \text{iff} \quad \forall s. \forall s_1 \in \phi_1(s). \forall s_2 \in \phi_2(s). s_1 \sqsupseteq s_2$$

Definition 11. A pair of weak morphisms $\phi : W_1 \rightarrow W_2$, $\psi : W_2 \rightarrow W_1$ forms a weak isomorphism, which we write as $\phi : W_1 \cong W_2 : \psi$, if $\psi \circ \phi \sqsupseteq \text{id}$ and $\phi \circ \psi \sqsupseteq \text{id}$.

Global Knowledge Constructions. Recall that for a monotone function $F \in \text{VRelF} \rightarrow \text{VRelF}$ and $R \in \text{VRelF}$, we write $[F]_R^*$ for the least fixed-point of the monotone function $F(-) \cup R$.

Definition 12. Given $\phi : W_1 \rightarrow W_2$ and $G \in \text{GK}(W_2)$, we define $\overleftarrow{G}_\phi \in W_1.S \rightarrow \text{VRelF}$ as follows.

$$\overleftarrow{G}_\phi(s_1) = [W_1.L(s_1)]_{(\bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} G(s'_2))}^*$$

Definition 13. Given $\phi : W_1 \rightarrow W_2$ and $G \in \text{GK}(W_1)$, we define $\overrightarrow{G}_\phi^{s_1} \in W_2.S \rightarrow \text{VRelF}$ for $s_1 \in W_1.S$ as follows.

$$\overrightarrow{G}_\phi^{s_1}(s_2) = [W_2.L(s_2)]_{(\bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} G(s'_1))}^*$$

Lemma 12. Given $\phi : W_1 \rightarrow W_2$ and $G \in \text{GK}(W_2)$, we have $\forall s_2 \in \phi(s_1). \overleftarrow{G}_\phi(s_1) = G(s_2)$.

Proof:

- Suppose $s_2 \in \phi(s_1)$.
- We first show $G(s_2) = \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} G(s'_2)$:
 - $G(s_2) \subseteq \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} G(s'_2)$ is obvious.
 - To see the other inclusion, note that whenever $s'_2 \in \phi(s'_1)$ and $s'_1 \sqsubseteq s_1$, then $s'_2 \sqsubseteq s_2$ and thus $G(s'_2) \subseteq G(s_2)$.
- Consequently we know $\overleftarrow{G}_\phi(s_1) = [W_1.L(s_1)]_{G(s_2)}^* = W_1.L(s_1)([W_1.L(s_1)]_{G(s_2)}^*) \cup G(s_2) \supseteq G(s_2)$.

- To show $[W_1.L(s_1)]_{G(s_2)}^* \subseteq G(s_2)$, it suffices by induction to show $W_1.L(s_1)(G(s_2)) \cup G(s_2) \subseteq G(s_2)$.
- This follows from

$$W_1.L(s_1)(G(s_2)) = W_2.L(s_2)(G(s_2)) \subseteq G(s_2) .$$

Lemma 13. If $\phi : W_1 \rightarrow W_2$ and $G \in \text{GK}(W_2)$, then $\overleftarrow{G}_\phi \in \text{GK}(W_1)$.

Proof:

- \overleftarrow{G}_ϕ is monotone by definition because $[W_1.L(s_1)]_{(-)}^*$ is monotone.
- We have

$$\overleftarrow{G}_\phi(s_1) = W_1.L(s_1)(\overleftarrow{G}_\phi(s_1)) \cup \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} G(s'_2) \supseteq W_1.L(s_1)(\overleftarrow{G}_\phi(s_1))$$
- Moreover, for any $\tau \in \text{CType}$ we have $\overleftarrow{G}_\phi(s_1)(\text{ref } \tau) = W_1.L(s_1)(\overleftarrow{G}_\phi(s_1))(\text{ref } \tau)$ because:

$$\begin{aligned} & \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} G(s'_2)(\text{ref } \tau) \\ &= \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} W_2.L(s'_2)(G(s'_2))(\text{ref } \tau) \quad (G \in \text{GK}(W_2)) \\ &= \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} W_1.L(s'_1)(G(s'_2))(\text{ref } \tau) \quad (\phi : W_1 \rightarrow W_2) \\ &= \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} W_1.L(s'_1)(\overleftarrow{G}_\phi(s'_1))(\text{ref } \tau) \quad (\text{Lemma 12}) \\ &\subseteq \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \in \phi(s'_1)} W_1.L(s_1)(\overleftarrow{G}_\phi(s_1))(\text{ref } \tau) \\ &= W_1.L(s_1)(\overleftarrow{G}_\phi(s_1))(\text{ref } \tau) \end{aligned}$$
- Using an analogous argument, for any $\mathbf{n} \in W_1.N = W_2.N$, we have $\overleftarrow{G}_\phi(s_1)(\mathbf{n}) = W_1.L(s_1)(\overleftarrow{G}_\phi(s_1))(\mathbf{n})$.

Lemma 14. Given $\phi : W_1 \rightarrow W_2$ and $G \in \text{GK}(W_1)$, we have $\forall s_2 \in \phi(s_1). \overrightarrow{G}_\phi^{s_1}(s_2) = G(s_1)$.

Proof:

- Suppose $s_2 \in \phi(s_1)$.
- We first show $G(s_1) = \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} G(s'_1)$:
 - $G(s_1) \subseteq \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} G(s'_1)$ is obvious.
 - To see the other inclusion, note that whenever $s'_1 \sqsubseteq s_1$, then $G(s'_1) \subseteq G(s_1)$.
- Consequently we know $\overrightarrow{G}_\phi^{s_1}(s_2) = [W_2.L(s_2)]_{G(s_1)}^* = W_2.L(s_2)([W_2.L(s_2)]_{G(s_1)}^*) \cup G(s_1) \supseteq G(s_1)$.
- To show $[W_2.L(s_2)]_{G(s_1)}^* \subseteq G(s_1)$, it suffices by induction to show $W_2.L(s_2)(G(s_1)) \cup G(s_1) \subseteq G(s_1)$.
- This follows from

$$W_2.L(s_2)(G(s_1)) = W_1.L(s_1)(G(s_1)) \subseteq G(s_1) .$$

Lemma 15. If $\phi : W_1 \rightarrow W_2$ and $G \in \text{GK}(W_1)$, then $\overrightarrow{G}_\phi^{s_1} \in \text{GK}(W_2)$ for any $s_1 \in W_1.S$.

Proof:

- We have the following monotonicity by definition because

$[W_2.L(s_2)]_{(-)}^*$ is monotone:

$$\forall \widehat{s}_1 \sqsupseteq \widehat{s}_1. \forall \widehat{s}_2 \sqsupseteq \widehat{s}_2. \overrightarrow{G_\phi^{s_1}}(\widehat{s}_2) \sqsupseteq \overrightarrow{G_\phi^{s_1}}(\widehat{s}_2)$$

• We have

$$\begin{aligned} \overrightarrow{G_\phi^{s_1}}(s_2) &= W_2.L(s_2)(\overrightarrow{G_\phi^{s_1}}(s_2)) \cup \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} G(s'_1) \\ &\supseteq W_2.L(s_2)(\overrightarrow{G_\phi^{s_1}}(s_2)) \end{aligned}$$

• Moreover, for any $\tau \in \text{CType}$ we have $\overrightarrow{G_\phi^{s_1}}(s_2)(\text{ref } \tau) = W_2.L(s_2)(\overrightarrow{G_\phi^{s_1}}(s_2))(\text{ref } \tau)$ because:

$$\begin{aligned} &\bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} G(s'_1)(\text{ref } \tau) \\ &= \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} W_1.L(s'_1)(G(s'_1))(\text{ref } \tau) \\ &\quad (G \in \text{GK}(W_1)) \\ &= \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} W_2.L(s'_2)(G(s'_1))(\text{ref } \tau) \\ &\quad (\phi : W_1 \rightarrow W_2) \\ &= \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} W_2.L(s'_2)(\overrightarrow{G_\phi^{s_1}}(s'_2))(\text{ref } \tau) \\ &\quad (\text{Lemma 14}) \\ &\subseteq \bigcup_{s'_1 \sqsubseteq s_1 \wedge s'_2 \sqsubseteq s_2 \wedge s'_2 \in \phi(s'_1)} W_2.L(s_2)(\overrightarrow{G_\phi^{s_1}}(s_2))(\text{ref } \tau) \\ &\quad (\text{Monotonicity}) \\ &= W_2.L(s_2)(\overrightarrow{G_\phi^{s_1}}(s_2))(\text{ref } \tau) \end{aligned}$$

• Using an analogous argument, for any $\mathbf{n} \in W_2.N = W_1.N$, we have $\overrightarrow{G_\phi^{s_1}}(s_2)(\mathbf{n}) = W_2.L(s_2)(\overrightarrow{G_\phi^{s_1}}(s_2))(\mathbf{n})$. ■

Category of Worlds.

Lemma 16. If $\phi : W_1 \rightarrow W_2$, then $\phi \circ \text{id}_{W_1} = \phi = \text{id}_{W_2} \circ \phi$.

Proof: Obvious. ■

Lemma 17. If $\phi : W_1 \rightarrow W_2$, $\psi : W_2 \rightarrow W_3$, $\chi : W_3 \rightarrow W_4$, then $\chi \circ (\psi \circ \phi) = (\chi \circ \psi) \circ \phi$.

Proof:

$$\begin{aligned} &s_4 \in (\chi \circ (\psi \circ \phi))(s_1) \\ \iff &\exists s_3. s_4 \in \chi(s_3) \wedge s_3 \in (\psi \circ \phi)(s_1) \\ \iff &\exists s_3, s_2. s_4 \in \chi(s_3) \wedge s_3 \in \psi(s_2) \wedge s_2 \in \phi(s_1) \\ \iff &\exists s_2. s_4 \in (\chi \circ \psi)(s_2) \wedge s_2 \in \phi(s_1) \\ \iff &s_4 \in ((\chi \circ \psi) \circ \phi)(s_1) \end{aligned}$$

Lemma 18. If $\phi : W_1 \rightarrow W_2$ and $\psi : W_2 \rightarrow W_3$, then $\psi \circ \phi$ forms a weak morphism.

Proof: Conditions (1) through (3) hold vacuously. Condition (4) follows from Lemmas 14 and 15. ■

Proposition 19. Worlds with weak morphisms form a category.

Proof: It follows from Lemmas 16, 17 and 18. ■

Isomorphism Theorem. We finally show that weak isomorphisms preserve term equivalence.

Definition 14. Given $\phi : W_1 \rightarrow W_2$, we define $|-|_\phi \in$

$\mathbb{P}(W_1.S) \rightarrow \mathbb{P}(W_2.S)$ as follows:

$$|S|_\phi = \{s_1 \in S \mid \phi(s_1) \neq \emptyset\}$$

When ϕ is clear from context, we often just write $|S|$.

Lemma 20. If $\phi : W_1 \cong W_2 : \psi$ and $G \in \text{GK}(W_2)$, then:

$$\bigcap_{s_1 \in |\psi(s_2)|} \mathbf{E}_{W_1}(\overleftarrow{G_\phi})(s_1) \subseteq \mathbf{E}_{W_2}(G)(s_2)$$

Proof: We define:

$$\mathbf{E}'_{W_2}(G)(s_2) = \bigcap_{s_1 \in |\psi(s_2)|} \mathbf{E}_{W_1}(\overleftarrow{G_\phi})(s_1)$$

We prove $\mathbf{E}'_{W_2} \subseteq \mathbf{E}_{W_2}$ by coinduction. Concretely, we have to show:

$\forall G, s_2, \tau.$

$\forall (e_1, e_2) \in \mathbf{E}'_{W_2}(G)(s_2)(\tau).$

$\forall (h_1, h_2) \in W_2.H(s_2)(G(s_2)).$

$\forall h_1^F, h_2^F. h_1 \uplus h_1^F \text{ defined} \wedge h_2 \uplus h_2^F \text{ defined} \implies$

$\exists h'_1, h'_2, v_1, v_2, K_1, K_2, e'_1, e'_2, s', \tau'.$

Case \uparrow : $(h_1 \uplus h_1^F, e_1) \hookrightarrow^\omega \wedge (h_2 \uplus h_2^F, e_2) \hookrightarrow^\omega$

Case \downarrow : $(h_1 \uplus h_1^F, e_1) \hookrightarrow^* (h'_1 \uplus h_1^F, v_1)$

$\wedge (h_2 \uplus h_2^F, e_2) \hookrightarrow^* (h'_2 \uplus h_2^F, v_2)$

$\wedge s'_2 \sqsupseteq s_2 \wedge (h'_1, h'_2) \in W_1.H(s'_2)(G(s'_2))$

$\wedge (v_1, v_2) \in \overrightarrow{G}(s'_2)(\tau)$

Case \downarrow : $(h_1 \uplus h_1^F, e_1) \hookrightarrow^* (h'_1 \uplus h_1^F, K_1[e'_1])$

$\wedge (h_2 \uplus h_2^F, e_2) \hookrightarrow^* (h'_2 \uplus h_2^F, K_2[e'_2])$

$\wedge s'_2 \sqsupseteq s_2 \wedge (h'_1, h'_2) \in W_1.H(s'_2)(G(s'_2))$

$\wedge (e'_1, e'_2) \in \mathbf{S}(G(s'_2), G(s'_2))(\tau')$

$\wedge \forall s''_2 \sqsupseteq s'_2. \forall G' \supseteq G. \forall (v'_1, v'_2) \in \overrightarrow{G'}(s''_2)(\tau').$

$(K_1[v'_1], K_2[v'_2]) \in \mathbf{E}'_{W_2}(G')(s''_2)(\tau')$

• Suppose $(e_1, e_2) \in \mathbf{E}'_{W_2}(G)(s_2)(\tau)$, and thus

$$(e_1, e_2) \in \bigcap_{s_1 \in |\psi(s_2)|} \mathbf{E}_{W_1}(\overleftarrow{G_\phi})(s_1)(\tau).$$

• Further suppose $(h_1, h_2) \in W_2.H(s_2)(G(s_2))$, and thus $(h_1, h_2) \in \bigcup_{s_1 \in \psi(s_2)} W_1.H(s_1)(G(s_2))$.

• Thus there exists $s_1 \in \psi(s_2)$ such that

$$(h_1, h_2) \in W_1.H(s_1)(G(s_2)).$$

• By Lemma 14 we know $G(s_2) = \overrightarrow{G_\psi^{s_2}}(s_1)$ and thus $W_1.H(s_1)(\overrightarrow{G_\psi^{s_2}}(s_1)) \neq \emptyset$.

• Since

$$W_1.H(s_1)(\overrightarrow{G_\psi^{s_2}}(s_1)) \subseteq \bigcup_{s'_2 \in \phi(s_1)} W_2.H(s'_2)(\overrightarrow{G_\psi^{s_2}}(s_1)),$$

there also exists $s'_2 \in \phi(s_1)$.

• Since $s'_2 \in (\phi \circ \psi)(s_2)$ and $\phi \circ \psi \sqsupseteq \text{id}$, we have that $s'_2 \sqsupseteq s_2$.

• Hence $(h_1, h_2) \in W_1.H(s_1)(G(s'_2))$, which means $(h_1, h_2) \in W_1.H(s_1)(\overleftarrow{G_\phi}(s_1))$ by Lemma 12.

• For h_1^F and h_2^F with defined $(h_1 \uplus h_1^F)$ and defined $(h_2 \uplus h_2^F)$ we then get three cases from $(e_1, e_2) \in \mathbf{E}_{W_1}(\overleftarrow{G_\phi})(s_1)(\tau)$:

(a) $h_1 \uplus h_1^F, e_1 \hookrightarrow^\omega \wedge h_2 \uplus h_2^F, e_2 \hookrightarrow^\omega$:

we are done.

(b) $h_1 \uplus h_1^F, e_1 \hookrightarrow^* h'_1 \uplus h_1^F, v_1 \wedge h_2 \uplus h_2^F, e_2 \hookrightarrow^* h'_2 \uplus h_2^F, v_2$

with $s'_1 \sqsupseteq s_1 \wedge (h'_1, h'_2) \in W_1.H(s'_1)(\overleftarrow{G_\phi}(s'_1)) \wedge$

$(v_1, v_2) \in \overrightarrow{G_\phi}(s'_1)(\tau)$:

– Since

$$W_1.H(s'_1)(\overleftarrow{G}_\phi(s'_1)) \subseteq \bigcup_{s''_2 \in \phi(s'_1)} W_2.H(s''_2)(\overleftarrow{G}_\phi(s'_1)),$$

there exists $s''_2 \in \phi(s'_1)$ such that $(h'_1, h'_2) \in W_2.H(s''_2)(G(s''_2))$ by Lemma 12.

– Also by Lemma 12 we have $(v_1, v_2) \in \overline{G}(s''_2)(\tau)$.

– We get $s''_2 \supseteq s'_2 \supseteq s_2$ from $s''_2 \in \phi(s'_1)$, $s'_2 \in \phi(s_1)$.

- (c) $h_1 \uplus h_1^F, e_1 \hookrightarrow^* h'_1 \uplus h_1^F, K_1[e'_1] \wedge$
 $h_2 \uplus h_2^F, e_2 \hookrightarrow^n h'_2 \uplus h_2^F, K_2[e'_2] \wedge$
 $s'_1 \supseteq s_1 \wedge (h'_1, h'_2) \in W_1.H(s'_1)(\overleftarrow{G}_\phi(s'_1)) \wedge$
 $(e'_1, e'_2) \in \mathbf{S}(\overleftarrow{G}_\phi(s'_1), \overleftarrow{G}_\phi(s'_1))(\tau')$ and
 $\forall s''_1 \supseteq s'_1. \forall G' \supseteq \overleftarrow{G}_\phi. \forall (v'_1, v'_2) \in \overline{G'}(s''_1)(\tau').$
 $(K_1[v'_1], K_2[v'_2]) \in \mathbf{E}_{W_1}(G')(s''_1)(\tau):$

– Since

$$W_1.H(s'_1)(\overleftarrow{G}_\phi(s'_1)) \subseteq \bigcup_{s''_2 \in \phi(s'_1)} W_2.H(s''_2)(\overleftarrow{G}_\phi(s'_1)),$$

there exists $s''_2 \in \phi(s'_1)$ such that

$$(h'_1, h'_2) \in W_2.H(s''_2)(G(s''_2))$$

by Lemma 12.

– Also by Lemma 12 we have

$$(e'_1, e'_2) \in \mathbf{S}(G(s''_2), G(s''_2))(\tau').$$

– We get $s''_2 \supseteq s'_2 \supseteq s_2$ from $s''_2 \in \phi(s'_1)$, $s'_2 \in \phi(s_1)$.

– It remains to show:

$$\forall s''_2 \supseteq s'_2. \forall G' \supseteq G. \forall (v'_1, v'_2) \in \overline{G'}(s''_2)(\tau').$$

$$(K_1[v'_1], K_2[v'_2]) \in \mathbf{E}'_{W_2}(G')(s''_2)(\tau)$$

– So suppose $s''_2 \supseteq s'_2$, $G' \supseteq G$ and $(v'_1, v'_2) \in \overline{G'}(s''_2)(\tau')$.

– By definition of \mathbf{E}'_{W_2} it suffices to show

$$(K_1[v'_1], K_2[v'_2]) \in \mathbf{E}_{W_1}(\overleftarrow{G}'_\phi)(s''_2)(\tau)$$

for any $s''_2 \in |\psi(s''_2)|$.

– Since

$$(h'_1, h'_2) \in W_2.H(s''_2)(G(s''_2)) \\ \subseteq \bigcup_{s''_1 \in \psi(s''_2)} W_1.H(s''_1)(G(s''_2)),$$

there exists $s''_1 \in \psi(s''_2)$.

– First, from $s''_1 \in |\psi(s''_2)|$ and $s''_2 \supseteq s''_1$ and $s''_1 \in \psi(s''_2)$ we get $s''_1 \supseteq s'_1$, which in turn yields $s''_1 \supseteq s'_1$ because $s''_1 \in (\psi \circ \phi)(s'_1)$.

– Second, note that $\overleftarrow{G}'_\phi \supseteq \overleftarrow{G}_\phi$.

– Third, pick $s''''_2 \in \phi(s''_1)$, so $s''''_2 \in (\phi \circ \psi)(s''_2)$ and thus $s''''_2 \supseteq s''_2$.

– Hence, $(v'_1, v'_2) \in \overline{G'}(s''_2)(\tau') \subseteq \overline{G'}(s''''_2)(\tau') = \overline{\overleftarrow{G}'_\phi}(s''_1)(\tau')$ by Lemma 12.

– The claim then follows from

$$\forall s''_1 \supseteq s'_1. \forall G' \supseteq \overleftarrow{G}_\phi. \forall (v'_1, v'_2) \in \overline{G'}(s''_1)(\tau'). \\ (K_1[v'_1], K_2[v'_2]) \in \mathbf{E}_{W_1}(G')(s''_1)(\tau).$$

■

Theorem 21 (Weak isomorphisms preserve equivalence).

If $\phi : W_1 \cong W_2 : \psi$, then: $\forall \Delta, \Gamma, \tau, e_1, e_2.$

$$\Delta; \Gamma \vdash e_1 \sim_{W_1} e_2 : \tau \iff \Delta; \Gamma \vdash e_1 \sim_{W_2} e_2 : \tau$$

Proof: By symmetry, it is enough to show

$$\Delta; \Gamma \vdash e_1 \sim_{W_1} e_2 : \tau \implies \Delta; \Gamma \vdash e_1 \sim_{W_2} e_2 : \tau.$$

From the premise we know:

- (1) *inhabited*(W_1)
- (2) *consistent*(W_1)
- (3) $\forall G \in \text{GK}(W_1). \forall s_1. \forall \delta \in \Delta \rightarrow \text{CType}. \\ \forall \gamma_1, \gamma_2 \in \text{dom}(\Gamma) \rightarrow \text{CVal}. \\ (\forall x: \tau' \in \Gamma. (\gamma_1(x), \gamma_2(x)) \in \overline{G}(s_1)(\delta\tau')) \implies \\ (\gamma_1 e_1, \gamma_2 e_2) \in \mathbf{E}_{W_1}(G)(s_1)(\delta\tau)$

I. We prove *inhabited*(W_2).

- Suppose $G \in \text{GK}(W_2)$.
- From (1) we know there is s_1 such that

$$(\emptyset, \emptyset) \in W_1.H(s_1)(\overleftarrow{G}_\phi(s_1)).$$

- From $W_1.H(s_1)(\overleftarrow{G}_\phi(s_1)) \subseteq \bigcup_{s_2 \in \phi(s_1)} W_2.H(s_2)(\overleftarrow{G}_\phi(s_1))$, we get $s_2 \in \phi(s_1)$ with $(\emptyset, \emptyset) \in W_2.H(s_2)(\overleftarrow{G}_\phi(s_1))$.
- By Lemma 12 we have $(\emptyset, \emptyset) \in W_2.H(s_2)(G(s_2))$.

II. We prove *consistent*(W_2).

- Let $G \in \text{GK}(W_2)$ and $(e'_1, e'_2) \in \mathbf{S}(W_2.L(s_2)(G(s_2)), G(s_2))(\tau')$.
- We need to show

$$(beta(e'_1), beta(e'_2)) \in \mathbf{E}_{W_2}(G)(s_2)(\tau').$$

- By Lemma 20, it suffices to show $(beta(e'_1), beta(e'_2)) \in \mathbf{E}_{W_1}(\overleftarrow{G}_\phi)(s_1)(\tau')$ for any $s_1 \in |\psi(s_2)|$.
- Since $\phi(s_1) \neq \emptyset$, we have $s'_2 \in \phi(s_1)$ and thus $\overleftarrow{G}_\phi(s_1) = G(s'_2)$ by Lemma 12.
- Since $s'_2 \in (\phi \circ \psi)(s_2)$, we have $s'_2 \supseteq s_2$ and thus $\overleftarrow{G}_\phi(s_1) = G(s'_2) \supseteq G(s_2)$.
- Thus we have

$$\mathbf{S}(W_2.L(s_2)(G(s_2)), G(s_2)) \\ \subseteq \mathbf{S}(W_2.L(s_2)(\overleftarrow{G}_\phi(s_1)), \overleftarrow{G}_\phi(s_1)) \\ = \mathbf{S}(W_1.L(s_1)(\overleftarrow{G}_\phi(s_1)), \overleftarrow{G}_\phi(s_1)).$$

- Consequently, the claim follows from (2).

III. We prove $\forall G \in \text{GK}(W_2). \forall s_2. \forall \delta \in \Delta \rightarrow \text{CType}.$

$$\forall \gamma_1, \gamma_2 \in \text{dom}(\Gamma) \rightarrow \text{CVal}. \\ (\forall x: \tau' \in \Gamma. (\gamma_1(x), \gamma_2(x)) \in \overline{G}(s_2)(\delta\tau')) \implies \\ (\gamma_1 e_1, \gamma_2 e_2) \in \mathbf{E}_{W_2}(G)(s_2)(\delta\tau).$$

- Let $G \in \text{GK}(W_2)$, $s_2 \in W_2.S$, $\delta \in \Delta \rightarrow \text{CType}$, $\gamma_1, \gamma_2 \in \text{dom}(\Gamma) \rightarrow \text{CVal}$ and suppose $\forall x: \tau' \in \Gamma. (\gamma_1(x), \gamma_2(x)) \in \overline{G}(s_2)(\delta\tau')$.
- By Lemma 20, it suffices to show $(\gamma_1 e_1, \gamma_2 e_2) \in \mathbf{E}_{W_1}(\overleftarrow{G}_\phi)(s_1)(\delta\tau)$ for any $s_1 \in |\psi(s_2)|$.
- Since $\phi(s_1) \neq \emptyset$, we have $s'_2 \in \phi(s_1)$ and thus $\overleftarrow{G}_\phi(s_1) = G(s'_2)$ by Lemma 12.
- Since $s'_2 \in (\phi \circ \psi)(s_2)$, we have $s'_2 \supseteq s_2$ and thus $\overleftarrow{G}_\phi(s_1) = G(s'_2) \supseteq G(s_2)$.
- Thus we have, for any $x: \tau' \in \Gamma$,

$$(\gamma_1(x), \gamma_2(x)) \in \overline{G}(s_2)(\delta\tau') \subseteq \overline{G}_\phi(s_1)(\delta\tau').$$

- Consequently, the claim follows from (3). ■