Technical Report MPI-SWS-2014-003
January 2014

A Logical Step Forward in Parametric Bisimulations

Chung-Kil Hur Georg Neis Derek Dreyer Viktor Vafeiadis
Seoul National University Max Planck Institute for Software Systems (MPI-SWS)
gil.hur@cse.snu.ac.kr {neis,dreyer,viktor}@mpi-sws.org

Abstract

In the last several years, a number of effective methods have been developed for reasoning about program equivalence in higher-order imperative languages like ML. Most recently, we proposed *parametric bisimulations (PBs)*, which fruitfully synthesize the direct coinductive style of bisimulations with the flexible invariants on local state afforded by Kripke logical relations, and which furthermore support transitive composition of equivalence proofs. However, the PB model of our previous work suffered from two limitations. First, it failed to validate the eta law for function values, which is important for our intended application of compiler certification. Second, it was not clear how to scale the method to reason about control effects.

In this paper, we propose *stuttering parametric bisimulations (SPBs)*, a variant of PBs that addresses their aforementioned limitations. Interestingly, despite the fact that the eta law and control effects seem like unrelated issues, our solutions to both problems hinge on the same technical device, namely the use of a "logical" reduction semantics that permits finite but unbounded stuttering in between physical steps. This technique is closely related to the key idea in *well-founded* and *stuttering* bisimulations, adapted here for the first time to reasoning about open, higher-order programs. We present SPBs—along with meta-theoretic results and example applications—for a language with recursive types and first-class continuations. Following our previous account of PBs, we can easily extend SPBs to handle abstract types and general mutable references as well (see the appendix for details). All our results have been fully mechanized in Coq.

Contents

1 – 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 the last several years, considerable progress has been made on this problem, primarily by advancements to two different classes of proof methods—*bisimulations* [20, 17, 18] and *step-indexed Kripke logical relations (SKLRs)* [1, 6].

In recent work [8], we proposed a new method for proving contextual equivalences in ML-like languages, which at the time we called *relation transition systems*, but since then have been referring to as *parametric bisimulations (PBs)*. PBs marry together some of the best features of state-of-the-art bisimulations and SKLRs into a single method:

- Like bisimulations [17, 19, 18], PBs support reasoning about "recursive" language features (*e.g.,* recursive types, higher-order state) in a *direct, coinductive* style.
- Like SKLRs [1, 6], PBs provide a very flexible treatment of "local" state, in which one can define a per-module *state transition system* to express and enforce invariants on how the module's local state may change over time.

Furthermore, PBs were designed to overcome some apparent limitations of their ancestors with regards to their potential to scale to *inter-language reasoning*—*i.e.,* reasoning compositionally about equivalences between programs written in different languages, such as the source and target of a certified compiler [2, 7]:

- Like SKLRs, PBs are truly *semantic*, in the sense that they do not rely fundamentally on the assumption that the programs they relate are written in the same language. (In contrast, bisimulations for higher-order languages rely crucially on this assumption.)
- Unlike SKLRs, the relation between programs induced by PBs is *transitive*. We believe this property will prove essential for our ultimate goal of using PBs for certifying *multi-phase* compilers, as it will enable correctness proofs for different phases to be linked transitively.

Unfortunately, the PB model we developed in our previous work [8] suffered from two key limitations:

1) It failed to validate the *eta law* for function values,
$$f : \sigma \to \tau \vdash f \sim (\lambda x.\, f\ x) : \sigma \to \tau,$$
as well as more complex equivalences (*e.g.,* the *syntactic minimal invariance* property [3]) which depend on it. This law is potentially quite important for our ultimate goal of compositional compiler certification, since eta-conversion is commonly used in compile-time optimization.
2) The language it modeled did not provide any form of control effects, such as *first-class continuations*, and it was not clear how to scale it to account for a language with such effects. In particular, the model baked in the assumption that the language in question had a "uniform" reduction semantics (*i.e.,* that the reduction relation was parametric in the evaluation context), an assumption that is of course not valid in the presence of callcc.

In this paper, we propose *stuttering parametric bisimulations (SPBs)*, which build closely on PBs and enjoy all their benefits, while also circumventing these two limitations. Interestingly, although the limitations of PBs appear at first glance to be completely unrelated, our solutions to both of them hinge on the same exact technical device, namely the use of a *logical reduction semantics* that permits finite but unbounded stuttering steps in between actual "physical" steps. Such stuttering steps effectively enable proofs of equivalence of programs to engage in a game of "hot potato", whereby the burden of proof may be tossed back and forth between different parts of the programs, until *eventually* some part makes a physical step of computation. This technique is inspired by the key idea in *well-founded* [15] and *stuttering* [4] bisimulations, adapted here for the first time to reasoning about open, higher-order programs.

There is a superficial sense in which SPB proofs may seem similar to SKLR proofs, namely that they both involve a certain element of "step-counting". The key difference is *which* steps they are counting: in SKLR proofs one counts physical reduction steps (because the model is built by induction on such steps), whereas in SPB proofs one only counts the stuttering steps (to ensure the absence of infinite stuttering). In essence, SPBs are "non-step"-indexed relations!

We present SPBs in the setting of λ^{μ}_{cc}, a CBV λ-calculus with recursive types and first-class continuations. λ^{μ}_{cc} provides a rich enough setting to explore the problems with PBs and the solutions offered by SPBs, while avoiding other orthogonal issues. Following our previous work on PBs, the SPB approach can be scaled straightforwardly to account for abstract types and general references, and the full SPB model supporting those features is given in the appendix. Moreover, our meta-theoretic results (soundness and transitivity—see Section 6) for the full PB model have been mechanized in Coq. This Coq development is available online at:

`http://www.mpi-sws.org/~neis/spbs`

We begin, in Section 2, by defining the language λ^{μ}_{cc} and its pure fragment λ^{μ} (lacking first-class continuations). In Section 3, we review our previous PB model for λ^{μ}, and the essential ideas behind it. We then present our SPB model in two stages. First, in Section 4, we explain the problem with the eta law, and how the logical reduction semantics fixes it, leading to the definition of a SPB model for λ^{μ} that closely follows the structure of the PB model. Second, in Section 5, we show how the same logical reduction semantics also aids in modeling control effects, and we use this insight to develop our generalized SPB model for λ^{μ}_{cc}. In Section 6, we summarize the key meta-theoretic results of the paper. Finally, in Section 7, we discuss related work and conclude.

2 – THE LANGUAGES λ^{μ} AND λ^{μ}_{cc}

Figure 1 gives the syntax of λ^{μ}_{cc}, along with excerpts of its static and dynamic semantics. The language is equipped with a standard type system, as well as a standard deterministic CBV dynamic semantics using evaluation contexts (aka

Statics: $\boxed{\Gamma \vdash p : \tau}$

$$
\begin{array}{lcl}
\Gamma & ::= & \cdot \mid \Gamma, x{:}\tau \quad \text{where } \mathrm{fv}(\tau) = \emptyset \\
\tau, \sigma & ::= & \alpha \mid \mathsf{int} \mid \sigma_1 \times \sigma_2 \mid \sigma_1 + \sigma_2 \mid \sigma_1 \to \sigma_2 \mid \mu\alpha.\,\sigma \mid \mathsf{cont}\,\sigma \\
p & ::= & x \mid n \mid p_1 \odot p_2 \mid \mathsf{ifz}\ p\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2 \mid \langle p_1, p_2\rangle \mid p.1 \mid \\
& & p.2 \mid \mathsf{inl}_\sigma\, p \mid \mathsf{inr}_\sigma\, p \mid (\mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow p_1 \mid \mathsf{inr}\, x \Rightarrow p_2) \mid \\
& & \mathsf{fix}\, f(x{:}\sigma_1){:}\sigma_2.\, p \mid p_1\ p_2 \mid \mathsf{roll}_\sigma\ p \mid \mathsf{unroll}\ p \mid \\
& & \mathsf{callcc}_\sigma\,(x.\ p) \mid \mathsf{throw}_\sigma\ p_1\ \mathsf{to}\ p_2 \mid \mathsf{isolate}\ p
\end{array}
$$

$$
\frac{\Gamma, f{:}(\tau_1 \to \tau_2), x{:}\tau_1 \vdash p : \tau_2}{\Gamma \vdash \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.\, p : \tau_1 \to \tau_2}
\qquad
\frac{\Gamma \vdash p_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash p_2 : \tau_1}{\Gamma \vdash p_1\ p_2 : \tau_2}
$$

$$
\frac{\Gamma, x{:}\mathsf{cont}\,\tau \vdash p : \tau}{\Gamma \vdash \mathsf{callcc}_\tau\,(x.\ p) : \tau}
\qquad
\frac{\Gamma \vdash p' : \tau' \quad \Gamma \vdash p : \mathsf{cont}\,\tau'}{\Gamma \vdash \mathsf{throw}_\tau\ p'\ \mathsf{to}\ p : \tau}
$$

$$
\frac{\Gamma \vdash p : \tau \to \mathsf{int}}{\Gamma \vdash \mathsf{isolate}\ p : \mathsf{cont}\,\tau}
$$

Dynamics: $\boxed{e \hookrightarrow e}$

$$
\begin{array}{lcl}
v & ::= & x \mid n \mid \langle v_1, v_2\rangle \mid \mathsf{inl}\ v \mid \mathsf{inr}\ v \mid \mathsf{fix}\, f(x).\, e \mid \mathsf{roll}\ v \mid \mathsf{cont}\ K \\
e & ::= & v \mid e_1 \odot e_2 \mid \mathsf{ifz}\ e_0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \langle e_1, e_2\rangle \mid e.1 \mid e.2 \mid \\
& & \mathsf{inl}\ e \mid \mathsf{inr}\ e \mid (\mathsf{case}\ e\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow e_1 \mid \mathsf{inr}\, x \Rightarrow e_2) \mid e_1\ e_2 \mid \\
& & \mathsf{roll}\ e \mid \mathsf{unroll}\ e \mid \mathsf{callcc}\,(x.\ e) \mid \mathsf{throw}\ e_1\ \mathsf{to}\ e_2 \mid \mathsf{isolate}\ e \\
K & ::= & \bullet \mid K \odot e \mid v \odot K \mid \mathsf{ifz}\ K\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \langle K, e\rangle \mid \\
& & \langle v, K\rangle \mid K.1 \mid K.2 \mid \mathsf{inl}\ K \mid \mathsf{inr}\ K \mid \\
& & (\mathsf{case}\ K\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow e_1 \mid \mathsf{inr}\, x \Rightarrow e_2) \mid K\ e \mid v\ K \mid \mathsf{roll}\ K \mid \\
& & \mathsf{unroll}\ K \mid \mathsf{throw}\ K\ \mathsf{to}\ e \mid \mathsf{throw}\ v\ \mathsf{to}\ K \mid \mathsf{isolate}\ K
\end{array}
$$

$$
\begin{array}{lcl}
K[(\mathsf{fix}\, f(x).\, e)\ v] & \hookrightarrow & K[e[(\mathsf{fix}\, f(x).\, e)/f, v/x]] \\
K[\mathsf{callcc}\,(x.\ e)] & \hookrightarrow & K[e[\mathsf{cont}\ K/x]] \\
K[\mathsf{throw}\ v\ \mathsf{to}\ \mathsf{cont}\ K'] & \hookrightarrow & K'[v] \\
K[\mathsf{isolate}\ v] & \hookrightarrow & K[\mathsf{cont}\ (v\ \bullet)]
\end{array}
$$

Fig. 1. Statics and dynamics (excerpts) of λ^μ and $\lambda^\mu_{\mathrm{cc}}$.

continuations) K. While the static semantics talks about typed (annotated) programs p, the dynamic semantics is defined for erased programs $e = |p|$.

First-class continuations have type $\mathsf{cont}\,\tau$, and all associated constructs and rules are highlighted in red. At runtime, a first-class continuation is just another value form ($\mathsf{cont}\ K$). There are two ways a program can get hold of such a value: either by capturing its current continuation via callcc, or by turning a function into a continuation via isolate (thereby isolating it from the current continuation). It can then yield control to a value of type $\mathsf{cont}\,\tau$ by throw-ing it a value of type τ.

The fragment λ^μ is obtained from $\lambda^\mu_{\mathrm{cc}}$ by simply dropping the highlighted constructs and rules.

Contextual equivalence for either language is defined in the usual way with the help of contexts C (programs with a hole):

Definition 1 (Contextual Equivalence)**.**

$$
\boxed{\Gamma \vdash p_1 \sim_{\mathrm{ctx}} p_2 : \tau} \;:=\; \Gamma \vdash p_1 : \tau \ \wedge\ \Gamma \vdash p_2 : \tau \ \wedge\ \forall C.
$$
$$
\vdash C : (\Gamma; \tau) \leadsto (\cdot; \mathsf{int}) \implies \big(|C[p_1]| \hookrightarrow^\omega \iff |C[p_2]| \hookrightarrow^\omega\big)
$$

Here $\hookrightarrow^\omega$ stands for divergence (infinite reduction). The definition of contexts and their typing judgment straightforwardly follows that of programs p and is omitted for space reasons. Suffice it to say that $\vdash C : (\Gamma; \tau) \leadsto (\Gamma'; \tau')$ implies $\forall p.\ \Gamma \vdash p : \tau \implies \Gamma' \vdash C[p] : \tau'$.

In the remainder, we often write $\lambda x.\, e$ as shorthand for the non-recursive function $\mathsf{fix}\, f(x).\, e$, where $f \notin \mathrm{fv}(e)$.

3 – Parametric Bisimulations (PBs)

In this section, we briefly review the main ideas behind our previous model of *parametric bisimulations (PBs)* [8] (originally named *relation transition systems*) as well as the formal definition of the PB model for λ^μ, which is given in the left column of Figure 2.

The top-level judgment of our PB model has the form $\Gamma \vdash e_1 \sim e_2 : \tau$, stipulating that e_1 and e_2 are equivalent terms at type τ. Typically, in coinductive proofs, to establish such an equivalence, one exhibits a *coinduction hypothesis* L, which relates e_1 and e_2 but also relates other auxiliary terms that one needs to prove equivalent in the course of proving e_1 and e_2 equivalent. The soundness of this hypothesis w.r.t. contextual equivalence is then established by proving it to be *consistent* in a certain sense (a.k.a. a "bisimulation").

PB proofs work in a similar way, but with an unusual twist concerning the treatment of (higher-order) functions. If in the coinduction hypothesis L we claim that two functions $\lambda x.e_1$ and $\lambda x.e_2$ are equivalent at type $\sigma \to \tau$, the aforementioned consistency condition will require us to demonstrate that e_1 and e_2 do in fact behave equivalently (at type τ) when instantiated with "equivalent arguments" at type σ.[1] A natural question then arises: from what relation do we draw these "equivalent arguments"? This is a tough question—if we knew how to usefully characterize when values of arbitrary type σ were equivalent, we would have solved our original problem!

The essential novelty of PBs is to answer this question precisely by *not* answering it. To understand this cryptic statement, let us make a distinction between *global knowledge* and *local knowledge*. Local knowledge is another term for the coinduction hypothesis L: it specifies which values *we* "know" are equivalent *in our proof*, but it is local to our proof—it does not pretend to be a global characterization of which values are equivalent in general. In contrast, the global knowledge G represents the sum total of knowledge concerning equivalence of values in the "whole program" in which $\lambda x.e_i$ appear, and it is this G from which we draw "equivalent arguments". The key insight of PB proofs is that, in order to verify the consistency of L, we do not need to know what G is. In fact, we do not even need to know that G is sound w.r.t. contextual equivalence. Rather, we simply take G as a *parameter* of our equivalence proof.

This idea of parameterizing the proof over the global knowledge G has important implications for how we define consistency for function values. If L relates $\lambda x.e_1$ and $\lambda x.e_2$ at $\sigma \to \tau$, then we must show that for all (v_1, v_2) related by G at σ, $e_1[v_1/x]$ and $e_2[v_2/x]$ "behave equivalently" at τ. The trouble is that, in an absolute sense, they don't. Knowing that v_1 and v_2 are related by G tells us absolutely nothing, since G is a parameter that can relate any two values (at least at function type—see the discussion of "value closure" below).

[1]The functions related by L may of course be recursive (*e.g.,* $\mathsf{fix}\, f(x).\, e$). We restrict attention to λ-terms here just to simplify the presentation.

For example, suppose that $\sigma = \mathsf{int} \to \mathsf{int}$, $\tau = \mathsf{int}$, and $e_1 = e_2 = x(0)$. In this case, we should clearly be able to prove $\lambda x.e_1$ and $\lambda x.e_2$ equivalent—they are syntactically equal!—but it is possible that they are passed as arguments, say, $v_1 = \lambda x.x + 1$ and $v_2 = 5$, in which case $e_1[v_1/x] \hookrightarrow^* 1$, while $e_2[v_2/x] \hookrightarrow^* 5(0)$, a stuck term. Even if we were to restrict G to, at function type, only relate λ-expressions (instead of arbitrary junk like the integer 5), G might still relate v_1 here with, say, a divergent function, in which case $e_2[v_2/x] \hookrightarrow^\omega$.

While $e_1[v_1/x]$ and $e_2[v_2/x]$ in this example clearly do not have the same observable behavior, they *can* be understood to have the same *local* behavior. Intuitively, two terms are locally equivalent w.r.t. G if they behave equivalently *modulo* what happens during calls to functions related by G. In the above example, $e_1[v_1/x]$ and $e_2[v_2/x]$ apply values related by G (namely, v_1 and v_2) to the same integer argument (0). The fact that they behave differently is thus not their own fault, but G's fault, so we can say that in fact they are locally equivalent.

This notion is formalized by the *local term equivalence* relation $\mathbf{E}(G)$. We say that two closed terms e_1 and e_2 are *locally equivalent* at a given type τ w.r.t. a global knowledge G—denoted $(e_1, e_2) \in \mathbf{E}(G)(\tau)$—if one of these cases holds:

(**Case** $\Uparrow$) they both diverge; or

(**Case** $\Downarrow$) they both reduce to related values; or

(**Case** $\lightning$) they both reduce to related "stuck" configurations ($\mathbf{S}(G, G)$), *i.e.,* function calls where related function values are applied to related argument values inside locally equivalent continuations ($\mathbf{K}(G)$). Locally equivalent continuations, in turn, are those which, when filled with related values, result (coinductively) in locally equivalent terms. In all cases, "related" values are drawn from the global knowledge G (or rather its value closure $\overline{G}$, as we explain below).

The definition of $\mathbf{E}(G)$ is highly reminiscent of *normal form* (or *open*) bisimulations [16, 12, 18], in which one establishes the consistency of equivalence of $\lambda x.e_1$ and $\lambda x.e_2$ by showing equivalence of e_1 and e_2 as open terms (assuming x is a "fresh" variable). Correspondingly, normal form bisimulations permit the proof to "get stuck" at a point where both terms apply the same function variable x (*e.g.,* in the example above, $x(0)$). This is no accident: PBs were inspired heavily by normal form bisimulations. The key difference is that PBs' use of a global knowledge is more *semantic*: by drawing v_1 and v_2 from an unknown G instead of modeling them as the same variable x, we have the potential to scale to reasoning about equivalences between different languages (*e.g.,* involving assembly, which has no notion of variable binding [7]).

To conclude this section, we mention three important technical points in the formalization of PBs.

First, we restrict the local and global knowledge to only relate (closed) *values*, not arbitrary terms, and only at "flexible" types, CTyF, which for λ^μ means just function types. Value equivalence at flexible types, $R \in \mathrm{VRelF}$, is then extended to all (closed) types by the inductively-constructed *value closure*, $\overline{R} \in \mathrm{VRel}$. (We call the non-flexible types "rigid" since the value closure defines equivalence at those types in a fixed, canonical way.) Term equivalence is accounted for by $\mathbf{E}(R)$.

Second, when parameterizing over the global knowledge G, we impose the condition that G should at least contain the local knowledge L of our proof, since global subsumes local knowledge. This requirement is critical in enabling coinductive reasoning; for example, to show that two recursive functions f_1 and f_2 related by L are equivalent, we may wish to show that $f_1(v_1)$ and $f_2(v_2)$ evaluate to some expressions of the form $K_1[f_1(v_1')]$ and $K_2[f_2(v_2')]$ (with (v_1', v_2') related by $\overline{G}$ and (K_1, K_2) by $\mathbf{K}(G)$). In this case, if we know that G contains L, then we also know that $(f_1, f_2) \in G$, and we can appeal to the "stuck" case ($\lightning$) of $\mathbf{E}(G)$ to complete the proof.

Formally, the requirement that G subsumes L is slightly more complicated, because we allow L to be *itself* parameterized over G (so long as it is monotone in its G parameter—see the definition of LK). This additional parameterization of L over G is essential: it enables L to assert equivalences between open terms by quantifying over closing instantiations drawn from G. (We will see an example of this in Section 4.) As a result, the condition that "G subsumes L", written $G \in \mathrm{GK}(L)$, is defined to mean that $G \supseteq L(G)$.

Finally, in order to keep the kind of coinductive reasoning we just described sound, we must be quite careful in the definition of "$\mathrm{consistent}(L)$". Specifically, for each pair of functions (f_1, f_2) related by $L(G)$, and arguments (v_1, v_2) related by $\overline{G}$, we cannot simply require $(f_1(v_1), f_2(v_2))$ to be related by $\mathbf{E}(G)$ because, via the stuck case of $\mathbf{E}(G)$, this is a tautology! Instead, we demand that $f_i(v_i) \hookrightarrow e_i$, and that e_1 and e_2 are related by $\mathbf{E}(G)$. Requiring the terms to take a step of reduction at this point ensures that "progress" is made in the coinductive argument, *i.e.,* that the coinduction is *guarded*. As we will see in the next section, however, the guardedness here is more restrictive than it ought to be.

4 – Stuttering Parametric Bisimulations (First Step: Validating Eta)

4.1 The Problem with Eta

We begin by reviewing the inherent problem with eta in the PB model. The eta law for an arbitrary function type $\tau' \to \tau$ corresponds to the following equivalence:

$$f : \tau' \to \tau \vdash f \sim (\lambda x.\, f\ x) : \tau' \to \tau$$

This equivalence does *not* hold for the PB model presented above. Here we prove as much for the case of $\tau' = \tau = \mathsf{int}$.

Proof: By definition the equivalence holds iff there exists a consistent local knowledge L such that for any global knowledge $G \in \mathrm{GK}(L)$ and any related values $(v_1, v_2) \in G(\mathsf{int} \to \mathsf{int})$ we have $(v_1, \lambda x.\, v_2\ x) \in \mathbf{E}(G)(\mathsf{int} \to \mathsf{int})$. Being values, such v_1 and $\lambda x.\, v_2\ x$ are related by $\mathbf{E}(G)$ iff they are related by G. Thus, in order to disprove the eta law, it suffices to construct a "bad" global knowledge $G^{\lightning} \in \mathrm{GK}(L)$ that relates v_1 and v_2 but does not relate v_1 and $\lambda x.\, v_2\ x$.

This is an easy task. Let $G^{\lightning}$ be the least global knowledge that subsumes L and also relates $\lambda y.\, 0$ and $\lambda y.\, 1$ at $\mathsf{int} \to \mathsf{int}$:[2]

$$G^{\lightning} = L(G^{\lightning}) \cup \{(\mathsf{int} \to \mathsf{int}, \lambda y.\, 0, \lambda x.\, 1)\}$$

[2] This is a simple fixed point construction since L is a monotone function.

CTy(F), CVal, CExp, CCont = the sets of closed (flexible) types, closed values, closed expressions, and closed continuations, respectively

$$
\begin{array}{lll}
\mathrm{VRelF} & := & \mathbb{P}(\mathrm{CTyF} \times \mathrm{CVal} \times \mathrm{CVal}) \\
\mathrm{VRel} & := & \mathbb{P}(\mathrm{CTy} \times \mathrm{CVal} \times \mathrm{CVal}) \\
\mathrm{KRel} & := & \mathbb{P}(\mathrm{CTy} \times \mathrm{CTy} \times \mathrm{CCont} \times \mathrm{CCont}) \\
\mathrm{ERel} & := & \mathbb{P}(\mathrm{CTy} \times \mathrm{CExp} \times \mathrm{CExp})
\end{array}
$$

$$
\begin{array}{lll}
\overline{R}(\tau \to \tau') & := & R(\tau \to \tau') \\
\overline{R}(\mathsf{int}) & := & \{(m, m)\} \\
\overline{R}(\tau \times \tau') & := & \{(\langle v_1, v_1'\rangle, \langle v_2, v_2'\rangle) \mid (v_1, v_2) \in \overline{R}(\tau)\} \\
 & \cap & \{(\langle v_1, v_1'\rangle, \langle v_2, v_2'\rangle) \mid (v_1', v_2') \in \overline{R}(\tau')\} \\
\overline{R}(\tau + \tau') & := & \{(\mathsf{inl}\, v_1, \mathsf{inl}\, v_2) \mid (v_1, v_2) \in \overline{R}(\tau)\} \\
 & \cup & \{(\mathsf{inr}\, v_1', \mathsf{inr}\, v_2') \mid (v_1', v_2') \in \overline{R}(\tau')\} \\
\overline{R}(\mu\alpha.\,\sigma) & := & \{(\mathsf{roll}\, v_1, \mathsf{roll}\, v_2) \mid (v_1, v_2) \in \overline{R}(\sigma[\mu\alpha.\,\sigma/\alpha])\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{LK} := \{L \in \mathrm{VRelF} \to \mathrm{VRelF} \mid \forall R.\ \forall(\tau, v_1, v_2) \in L(R). \\
\qquad (\forall R' \supseteq R.\ (\tau, v_1, v_2) \in L(R')) \wedge \\
\qquad (\exists f_i, x_i, e_i.\ v_i = \mathsf{fix}\, f_i(x_i).\, e_i)\}
\end{array}
$$

$$
\mathrm{GK}(L) := \{G \in \mathrm{VRelF} \mid G \supseteq L(G)\}
$$

$$
\begin{array}{l}
\mathbf{S} \in \mathrm{VRelF} \times \mathrm{VRelF} \to \mathrm{ERel} \\
\mathbf{S}(R_c, R) := \{(\tau, v_1\ v_1', v_2\ v_2') \mid \exists \tau'. \\
\qquad (\tau' \to \tau, v_1, v_2) \in R_c \wedge (\tau', v_1, v_2) \in \overline{R}\}
\end{array}
$$

$$
\begin{array}{l}
\mathbf{K} \in \mathrm{VRelF} \to \mathrm{KRel} \\
\mathbf{K}(R) := \{(\tau', \tau, K_1, K_2) \mid \forall (v_1, v_2) \in \overline{R}(\tau'). \\
\qquad (\tau, K_1[v_1], K_2[v_2]) \in \mathbf{E}(R)\}
\end{array}
$$

$$
\begin{array}{l}
\mathbf{E} \in \mathrm{VRelF} \to \mathrm{ERel} \\
\mathbf{E}(R) := \{(\tau, e_1, e_2) \mid (e_1 \hookrightarrow^{\omega} \wedge e_2 \hookrightarrow^{\omega}) \vee \\
\qquad (\exists (v_1, v_2) \in \overline{R}(\tau). \\
\qquad\quad e_1 \hookrightarrow^{*} v_1 \wedge e_2 \hookrightarrow^{*} v_2) \vee \\
\qquad (\exists (\tau', e_1', e_2') \in \mathbf{S}(R, R). \\
\qquad\ \exists (K_1, K_2) \in \mathbf{K}(R)(\tau', \tau). \\
\qquad\quad e_1 \hookrightarrow^{*} K_1[e_1'] \wedge e_2 \hookrightarrow^{*} K_2[e_2'])\}
\end{array}
$$

$$
\begin{array}{l}
R^{\$} := \{(\tau, e_1, e_2) \mid \exists (e_1', e_2') \in R(\tau). \\
\qquad\qquad e_1 \hookrightarrow e_1' \wedge e_2 \hookrightarrow e_2'\}
\end{array}
$$

$$
\mathrm{consistent}(L) := \forall G \in \mathrm{GK}(L).\ \mathbf{S}(L(G), G) \subseteq \mathbf{E}(G)^{\$}
$$

$$
\begin{array}{lll}
R(\cdot) & := & \{(\emptyset, \emptyset)\} \\
R(x{:}\tau, \Gamma) & := & \{(\gamma_1[x \mapsto v_1], \gamma_2[x \mapsto v_2]) \mid \\
 & & (v_1, v_2) \in R(\tau) \wedge (\gamma_1, \gamma_2) \in R(\Gamma)\}
\end{array}
$$

$$
\begin{array}{l}
\Gamma \vdash e_1 \sim_L e_2 : \tau := \ \forall G \in \mathrm{GK}(L).\ \forall (\gamma_1, \gamma_2) \in \overline{G}(\Gamma). \\
\qquad\qquad (\tau, \gamma_1 e_1, \gamma_2 e_2) \in \mathbf{E}(G)\}
\end{array}
$$

$$
\Gamma \vdash e_1 \sim e_2 : \tau := \ \exists L.\ \mathrm{consistent}(L) \wedge \Gamma \vdash e_1 \sim_L e_2 : \tau
$$

$$
\begin{array}{lll}
\mathrm{VRelF} & := & \mathbb{P}(\mathrm{CTyF} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \\
\mathrm{VRel} & := & \mathbb{P}(\mathrm{CTy} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \\
\mathrm{KRel} & := & \mathbb{P}(\mathrm{CTy} \times \mathrm{CTy} \times \mathbb{N} \times \mathrm{CCont} \times \mathbb{N} \times \mathrm{CCont}) \\
\mathrm{ERel} & := & \mathbb{P}(\mathrm{CTy} \times \mathbb{N} \times \mathrm{CExp} \times \mathbb{N} \times \mathrm{CExp})
\end{array}
$$

$$
\begin{array}{lll}
\overline{R}(\tau \to \tau') & := & R(\tau \to \tau') \\
\overline{R}(\mathsf{int}) & := & \{(\_, m, \_, m)\} \\
\overline{R}(\tau \times \tau') & := & \{(\_, \langle v_1, v_1'\rangle, \_, \langle v_2, v_2'\rangle) \mid (\_, v_1, \_, v_2) \in \overline{R}(\tau)\} \\
 & \cap & \{(\_, \langle v_1, v_1'\rangle, \_, \langle v_2, v_2'\rangle) \mid (\_, v_1', \_, v_2') \in \overline{R}(\tau')\} \\
\overline{R}(\tau + \tau') & := & \{(\_, \mathsf{inl}\, v_1, \_, \mathsf{inl}\, v_2) \mid (\_, v_1, \_, v_2) \in \overline{R}(\tau)\} \\
 & \cup & \{(\_, \mathsf{inr}\, v_1', \_, \mathsf{inr}\, v_2') \mid (\_, v_1', \_, v_2') \in \overline{R}(\tau')\} \\
\overline{R}(\mu\alpha.\,\sigma) & := & \{(\_, \mathsf{roll}\, v_1, \_, \mathsf{roll}\, v_2) \mid (\_, v_1, \_, v_2) \in \overline{R}(\sigma[\mu\alpha.\,\sigma/\alpha])\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{LK} := \{L \in \mathrm{VRelF} \to \mathrm{VRelF} \mid \forall R.\ \forall(\tau, n_1, v_1, n_2, v_2) \in L(R). \\
\qquad \forall R' \supseteq R, n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', v_1, n_2', v_2) \in L(R')\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{GK}(L) := \{G \in \mathrm{VRelF} \mid G \supseteq L(G) \wedge \forall (\tau, n_1, v_1, n_2, v_2) \in G. \\
\qquad \forall n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', v_1, n_2', v_2) \in G\}
\end{array}
$$

$$
\begin{array}{l}
\mathbf{S} \in \mathrm{VRelF} \times \mathrm{VRelF} \to \mathrm{ERel} \\
\mathbf{S}(R_c, R) := \{(\tau, n_1, v_1\ v_1', n_2, v_2\ v_2') \mid \exists \tau'. \\
\qquad (\tau' \to \tau, n_1, v_1, n_2, v_2) \in R_c \wedge (\tau', \_, v_1', \_, v_2') \in \overline{R}\}
\end{array}
$$

$$
\begin{array}{l}
\mathbf{K} \in \mathrm{VRelF} \to \mathrm{KRel} \\
\mathbf{K}(R) := \{(\tau', \tau, k_1, K_1, k_2, K_2) \mid \forall (n_1, v_1, n_2, v_2) \in \overline{R}(\tau'). \\
\qquad (\tau, k_1 + n_1, K_1[v_1], k_2 + n_2, K_2[v_2]) \in \mathbf{E}(R)\}
\end{array}
$$

$$
\begin{array}{l}
\mathbf{E} \in \mathrm{VRelF} \to \mathrm{ERel} \\
\mathbf{E}(R) := \{(\tau, n_1, e_1, n_2, e_2) \mid (n_1, e_1 \hookrightarrow^{\omega} \wedge n_2, e_2 \hookrightarrow^{\omega}) \vee \\
\qquad (\exists (n_1', v_1, n_2', v_2) \in \overline{R}(\tau). \\
\qquad\quad n_1, e_1 \hookrightarrow^{*} n_1', v_1 \wedge n_2, e_2 \hookrightarrow^{*} n_2', v_2) \vee \\
\qquad (\exists (\tau', n_1', e_1', n_2', e_2') \in \mathbf{S}(R, R). \\
\qquad\ \exists (\_, K_1, \_, K_2) \in \mathbf{K}(R)(\tau', \tau). \\
\qquad\quad n_1, e_1 \hookrightarrow^{*} n_1', K_1[e_1'] \wedge n_2, e_2 \hookrightarrow^{*} n_2', K_2[e_2'])\}
\end{array}
$$

$$
\begin{array}{l}
R^{\$} := \{(\tau, n_1, e_1, n_2, e_2) \mid \exists (n_1', e_1', n_2', e_2') \in R(\tau). \\
\qquad\qquad n_1, e_1 \hookrightarrow n_1', e_1' \wedge n_2, e_2 \hookrightarrow n_2', e_2'\}
\end{array}
$$

$$
\mathrm{consistent}(L) := \forall G \in \mathrm{GK}(L).\ \mathbf{S}(L(G), G) \subseteq \mathbf{E}(G)^{\$}
$$

$$
\begin{array}{lll}
R(\cdot) & := & \{(\cdot, \emptyset, \cdot, \emptyset)\} \\
R(x{:}\tau, \Gamma) & := & \{(n_1 :: N_1, \gamma_1[x \mapsto v_1], n_2 :: N_2, \gamma_2[x \mapsto v_2]) \mid \\
 & & (n_1, v_1, n_2, v_2) \in R(\tau) \wedge (N_1, \gamma_1, N_2, \gamma_2) \in R(\Gamma)\}
\end{array}
$$

$$
\begin{array}{l}
\Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau := \ \forall G \in \mathrm{GK}(L).\ \forall (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma). \\
\qquad\qquad (\tau, f_1(N_1), \gamma_1 e_1, f_2(N_2), \gamma_2 e_2) \in \mathbf{E}(G)\}
\end{array}
$$

$$
\Gamma \vdash e_1 \sim e_2 : \tau := \ \exists L, f_1, f_2.\ \mathrm{consistent}(L) \wedge \Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau
$$

Fig. 2. Models for λ^{μ}: Original PB model (left) and new SPB model with logical reduction (right).

It remains to prove that $G^{\not\downarrow}$ does not relate $\lambda y.\, 0$ and $\lambda x.\, (\lambda y.\, 1)\ \ x$. Arguing by contradiction, assume it does. Then from $\mathrm{consistent}(L)$ and, say, $(\mathsf{int}, 42, 42) \in \overline{G^{\not\downarrow}}$ we know $(\mathsf{int}, (\lambda y.\, 0)\ \ 42, (\lambda x.\, (\lambda y.\, 1)\ \ x)\ \ 42) \in \mathbf{E}(G^{\not\downarrow})^{\$}$, *i.e.,* $(\mathsf{int}, 0, (\lambda y.\, 1)\ \ 42) \in \mathbf{E}(G^{\not\downarrow})$. Since 0 is a value, this can only mean that $(\lambda y.\, 1)\ \ 42$ reduces to a related value, *i.e.,* $(\mathsf{int}, 0, 1) \in \overline{G^{\not\downarrow}}$, which by definition of $\overline{G^{\not\downarrow}}$ is false. ■

To understand better what is going on here, let us now try to *prove* the eta law and see what goes wrong. As is evident from the reasoning at the beginning of the above disproof, we would have to construct a consistent local knowledge L such that any G subsuming it relates v_1 and $\lambda x.\, v_2\ x$ whenever it relates v_1 and v_2. Since the only leverage we have over G is what we put in L, the only way to force G to relate certain things is to choose L so that it relates them. Luckily, our definition of L may depend on G as a parameter, so in order to obtain the aforementioned closure property, we can attempt to define the local knowledge as follows:

$$
L_{\eta}(R) := \{(\tau' \to \tau, v_1, \lambda x.\, v_2\ x) \mid (\tau' \to \tau, v_1, v_2) \in R\}
$$

Intuitively, this local knowledge corresponds exactly to what we want to claim: if our context provides us with values v_1

and v_2 that are equivalent at $\tau' \to \tau$, then we are prepared to claim that v_1 and $\lambda x.\, v_2\, x$ are equivalent at the very same type. Unfortunately, this L_η may be inconsistent! Specifically, suppose we are given $G \in \mathrm{GK}(L_\eta)$ and related arguments $(\tau', v_1', v_2') \in \overline{G}$. For any $(v_1, v_2) \in G(\tau' \to \tau)$, we must show $(\tau, v_1\ v_1', (\lambda x.\, v_2\ x)\ v_2') \in \mathbf{E}(G)^\$$, *i.e.,* $(\tau, e_1, e_2) \in \mathbf{E}(G)$, where $v_1\ v_1' \hookrightarrow e_1$ and $(\lambda x.\, v_2\ x)\ v_2' \hookrightarrow e_2$. The trouble is that, while we know that $e_2 = v_2\ v_2'$, we have no way of knowing whether e_1 even exists. It is entirely possible, for instance, that v_1 is the integer 5, in which case $v_1\ v_1' \not\hookrightarrow$.

The problem here essentially is that the global knowledge G is under no obligation to be sound w.r.t. contextual equivalence. As a result, if we define a local knowledge like L_η that "re-exports" function values (like v_1) that it obtains from G, there is no way to know whether applications of such values reduce to well-behaved terms, or even if they are able to take a step of reduction at all.

4.2 Guardedness Revisited

As discussed at the end of Section 3, this requirement of "taking a step" in the definition of consistency is crucial in ensuring the soundness of PBs because it guarantees that the coinduction is suitably *guarded*. As the failed proof attempt shows, however, the guardedness condition appears to be a little too strict. Note that if we were not forced to take that step, then we could easily finish the proof of $\mathrm{consistent}(L_\eta)$ by appealing to $(\tau, v_1\ v_1', v_2\ v_2') \in \mathbf{E}(G)$, which follows from $(\tau' \to \tau, v_1, v_2) \in G$ and $(\tau', v_1', v_2') \in \overline{G}$ (both given), and $(\tau, \tau, \bullet, \bullet) \in \mathbf{K}(G)$.

Of course, we cannot simply drop the stepping requirement, since this would immediately result in unsoundness—we must have some way of ensuring "productivity" of proofs. What we want to do then, in order to obtain a model that validates the eta law, is to find a slightly weaker guardedness condition (leading to a weaker notion of consistency) that enables the sketched proof of the eta law to go through but is nevertheless strong enough to guarantee soundness of the model.

We achieve this by generalizing the physical notion of "taking a step" to a *logical* one.

4.3 Logical Reduction and the Stutter Budget

The idea is very simple. We introduce into the model what we call a *stutter budget* (or just *budget*, for short): two natural numbers, one for each program, that specify how many times one may "stutter"—*i.e.,* avoid taking a reduction step (thus seemingly making no progress)—before eventually taking a step. More precisely, a local knowledge will contain items of the form $(\tau, n_1, v_1, n_2, v_2)$ rather than just (τ, v_1, v_2). When proving consistency for such an item, *i.e.,* when showing that the applications of (v_1, v_2) to related arguments (v_1', v_2') are related, one then has to make a choice for each application before continuing the reasoning: either one reduces it by one physical step (as before), *or* one leaves it untouched but decreases the corresponding budget instead (n_1 for the application of v_1, and n_2 for the application of v_2). When one chooses the latter option, one temporarily shirks one's responsibility to make physical progress, passing the proof burden—or "hot potato" as we called it in the introduction—to the subgoal of showing that the applications are in the $\mathbf{E}$ relation. Using the "stuck" case, the $\mathbf{E}$ relation may then do the same thing and pass the hot potato back to the local knowledge. The important thing is that, each time around this seemingly circular proof path, the respective stutter budgets (n_1 and/or n_2) must be decreased, so we know the hot potato game cannot go on forever.

The way we formulate this is that one *is* actually required to perform a reduction step on both sides, as before, but only a *logical* one. (The exact changes to the model will be explained in a moment.) This logical reduction relation, operating on an expression and its budget, is defined as follows.

Definition 2 (Logical Reduction)**.**

$$\frac{e \hookrightarrow e'}{n, e \hookrightarrow n', e'} \qquad \frac{n' < n}{n, e \hookrightarrow n', e}$$

That is, a logical step is either a physical step, in which case one may pick an arbitrary budget n' to continue with, or a stutter step, in which case the budget must be decreased.

To get an intuition for why the proposed change to the model is sound, first observe that, since the stutter budget is finite, progress (in the form of a physical step) will eventually be made. Second, note that logical (non-)termination coincides with physical (non-)termination. Thus, logical reduction gives us more flexibility in terms of local reasoning about v_1 and v_2, and this added flexibility is perfectly sound in that it will not enable us to equate terminating and divergent programs.

4.4 Stuttering Parametric Bisimulations (SPBs) for λ^μ

Our new stuttering parametric bisimulation (SPB) model for λ^μ is given in the right column of Figure 2, adjacent to the old PB model so that it is easy to see the (modest) changes: ignoring the stutter budget, there is no difference.

First, wherever the old model related two expressions, the new model additionally carries their budget. One may think of an expression and its part of the budget as a *logical expression*, in which case the change is that the new model relates logical expressions. Next, the value closure $\overline{R}$ of a relation R does not care about the budget, except for function types, where it is passed on unmodified. For any other type, the closure relates values at any budget. (For the sake of readability we use an underscore, _, to stand for a fresh existentially quantified meta variable. When occurring in a binding position, such as the left side of a set comprehension, an underscore acts as a wildcard and matches anything.) The reason for this is that what really matters is the budget of functions, because we have to show consistency for them. If a pair or sum contains a function, then in order to access that function one already has to take a step (projection or case analysis, respectively), so progress is ensured regardless of the budget.

The definitions of LK and $\mathrm{GK}(L)$ are slightly modified from those in the PB model, in order to stipulate the condition of *budget monotonicity*. This property says that two related values must stay related when their budget is increased (of course, since it is always safe to give them more stuttering steps than they actually need). Unlike before, however, LK does not require the values to be proper syntactic function values. This is important because we want to be able to define local knowledges that, as in the proof sketch for the eta law, re-export values from the current global knowledge, which of course may be arbitrary junk. The reason for this restriction in the PB model had to do with an issue in the proof of transitivity, and we will see in Section 6.1 how we now instead make use of the stutter budget to resolve that issue.

Turning to the definition of the $\mathbf{S}$ relation, note that the only thing that counts is the budget of the *functions* being applied, not that of their arguments or continuations. This is important in the proof of transitivity. Intuitively, it is also sufficient because, before they can access their arguments or return to their continuations, the functions will have to take a physical beta-reduction step anyways. In the definition of the continuation relation, on the other hand, notice that the continuations' budgets are *added* to those of their input values. This corresponds to the idea that the continuations may stutter k_i times before passing the "hot potato" to their input values, which in turn demand stutter budgets of n_i.

Otherwise, the definition of the expression relation and of consistency are the same as before, just with physical reduction replaced by logical reduction.

Finally, in the equivalence judgment, we say that, besides a consistent local knowledge, there must exist a budget at which the programs are related. This budget may depend on the budgets of the values that are plugged in for free variables. It is not sufficient to just add up those budgets, because variables may occur multiple times and thus the dependency may not be linear. We therefore ask for a pair of budget functions f_1 and f_2, which, given a list of the input budgets, calculate e_1 and e_2's part of the final budget, respectively.

$$
\begin{aligned}
\mathrm{DRel} \quad &:= \mathbb{P}((\mathrm{CTyF} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \\
&\qquad \uplus (\mathrm{CTy} \times \mathbb{N} \times \mathrm{CCont} \times \mathbb{N} \times \mathrm{CCont})) \\
\mathrm{ERel} \quad &:= \mathbb{P}(\mathbb{N} \times \mathrm{CExp} \times \mathbb{N} \times \mathrm{CExp})
\end{aligned}
$$

$$\overline{R}(\mathsf{cont}\,\tau) := \{(\_, \mathsf{cont}\ K_1, \_, \mathsf{cont}\ K_2) \mid (\_, K_1, \_, K_2) \in R(\tau)\}$$

$$
\begin{aligned}
\mathrm{LK} := \{L \in \mathrm{DRel} \to \mathrm{DRel} \mid \forall R.\ &\forall(\tau, n_1, d_1, n_2, d_2) \in L(R). \\
&\forall R' \supseteq R, n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', d_1, n_2', d_2) \in L(R')\}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{GK}(L) := \{G \in \mathrm{DRel} \mid\ & G \supseteq L(G) \wedge (0, \bullet, 0, \bullet) \in G(\mathsf{int}) \wedge \\
& \forall(n_1, d_1, n_2, d_2) \in G.\ \forall n_1' \geq n_1, n_2' \geq n_2. \\
& (n_1', d_1, n_2', d_2) \in G\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{S} \in \mathrm{DRel} \times \mathrm{DRel} \to \mathrm{ERel} \\
&\mathbf{S}(R_c, R) := \{(k_1 + n_1, K_1[v_1], k_2 + n_2, K_2[v_2]) \mid \exists\tau. \\
&\qquad (\tau, k_1, K_1, k_2, K_2) \in R_c \wedge (\tau, n_1, v_1, n_2, v_2) \in \overline{R}\} \cup \\
&\qquad \{(n_1, K_1[v_1\ v_1'], n_2, K_2[v_2\ v_2']) \mid \exists\tau, \tau'. \\
&\qquad (\tau' \to \tau, n_1, v_1, n_2, v_2) \in R_c \wedge \\
&\qquad (\tau', \_, v_1', \_, v_2') \in \overline{R} \wedge (\tau, \_, K_1, \_, K_2) \in R\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{E} \in \mathrm{DRel} \to \mathrm{ERel} \\
&\mathbf{E}(R) := \{(n_1, e_1, n_2, e_2) \mid (e_1 \hookrightarrow^\omega \wedge e_2 \hookrightarrow^\omega) \vee \\
&\qquad (\exists(n_1', e_1', n_2', e_2') \in \mathbf{S}(R, R). \\
&\qquad\quad n_1, e_1 \hookrightarrow^* n_1', e_1' \wedge n_2, e_2 \hookrightarrow^* n_2', e_2')\}
\end{aligned}
$$

$$
\begin{aligned}
R^\$:= \{(n_1, e_1, n_2, e_2) \mid\ & \exists(n_1', e_1', n_2', e_2') \in R. \\
& n_1, e_1 \hookrightarrow n_1', e_1' \wedge n_2, e_2 \hookrightarrow n_2', e_2'\}
\end{aligned}
$$

$$\mathrm{consistent}(L) := \forall G \in \mathrm{GK}(L).\ \mathbf{S}(L(G), G) \subseteq \mathbf{E}(G)^\$$$

$$
\begin{aligned}
&\Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau := \forall G \in \mathrm{GK}(L). \\
&\qquad \forall(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma).\ \forall(k_1, K_1, k_2, K_2) \in G(\tau). \\
&\qquad (f_1(N_1) + k_1, K_1[\gamma_1 e_1], f_2(N_2) + k_2, K_2[\gamma_2 e_2]) \in \mathbf{E}(G)
\end{aligned}
$$

$$
\begin{aligned}
&\Gamma \vdash e_1 \sim e_2 : \tau := \\
&\qquad \exists L, f_1, f_2.\ \mathrm{consistent}(L) \wedge \Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau
\end{aligned}
$$

Fig. 3. Stuttering parametric bisimulations with continuation knowledge.

4.5 Eta Revisited

Using this model, we can now prove the eta law along the lines of the earlier attempt.

Theorem 1. $f : \tau' \to \tau \vdash f \sim (\lambda x.\, f\ x) : \tau' \to \tau$

Proof: We define the local knowledge L_η as follows:

$$
\begin{aligned}
L_\eta(R) := \{(\tau' \to \tau, n_1', v_1, \_, \lambda x.\, v_2\ x) \mid& \\
\exists n_1 < n_1'.\ (\tau' \to \tau, n_1, v_1, \_, v_2) \in R&\}
\end{aligned}
$$

We first show $f : \tau' \to \tau \vdash f \sim_{L,\mathrm{hd}+1,\tilde{0}} (\lambda x.\, f\ x) : \tau' \to \tau$, where hd returns the first (here: only) element of a list, $\mathrm{hd}+1$ is short for the meta function $\lambda l.\ \mathrm{hd}(l) + 1$, and $\tilde{0}$ is the constant-0 function.

- Suppose $G \in \mathrm{GK}(L_\eta)$ and $(n_1, v_1, n_2, v_2) \in G(\tau' \to \tau)$.
- We must show $(n_1{+}1, v_1, 0, \lambda x.\, v_2\ x) \in \mathbf{E}(G)(\tau' \to \tau)$.
- This follows from $L_\eta(G) \subseteq G$ by construction of L_η.

It remains to show $\mathrm{consistent}(L_\eta)$.

- Suppose $G \in \mathrm{GK}(L_\eta)$, $(n_1, v_1, \_, v_2) \in G(\tau' \to \tau)$, $n_1' > n_1$, n_2' arbitrary, and $(\_, v_1', \_, v_2') \in \overline{G}(\tau')$.
- We must show $(n_1', v_1\, v_1', n_2', (\lambda x.\, v_2\, x)\, v_2') \in \mathbf{E}(G)^\$(\tau)$.
- After taking a stutter step on the left side and a physical step on the right, it thus suffices to show $(n_1, v_1\ v_1', \_, v_2\ v_2') \in \mathbf{E}(G)(\tau)$.
- Since $(\_, \bullet, \_, \bullet) \in \mathbf{K}(G)(\tau, \tau)$ (trivial to prove), we are done if we can show $(n_1, v_1\ v_1', \_, v_2\ v_2') \in \mathbf{S}(G, G)(\tau)$.
- Indeed, this follows from $(n_1, v_1, \_, v_2) \in G(\tau' \to \tau)$ and $(\_, v_1', \_, v_2') \in \overline{G}(\tau')$. $\square$

5 – Stuttering Parametric Bisimulations (Second Step: Supporting Continuations)

In the previous section, we have seen how to enrich the PB model such that it validates the eta law. In this section, we will see how the added machinery, the stutter budget, enables us to also add support for control effects, such as first-class continuations.

The crucial characteristic of a language with control effects is that its semantics is context-sensitive, by which we mean that the following property does not hold universally:

$$e \hookrightarrow e' \implies K[e] \hookrightarrow K[e']$$

It is easy to check (see Figure 1) that λ^μ satisfies this property, while, due to callcc, $\lambda^\mu_{\mathrm{cc}}$ does not.

In the model from Section 4 (as well as in the PB model), proving two programs equivalent involves executing them directly in the empty context (*i.e.,* without a continuation). However, without the above property, this is clearly unsound. We therefore have to "contextualize" the model such that programs are always executed in an (arbitrary) evaluation context. We wish do this in a way that (1) is independent of first-class continuations—in order to scale to other context-sensitive features—and (2) requires only a modest change to actually support first-class continuations, *i.e.,* to obtain a sound model for $\lambda^{\mu}_{\mathrm{cc}}$.

5.1 Contextualizing the Model

Since we already have a continuation relation ($\mathbf{K}$), one may be tempted to use that in order to contextualize the model. However, this approach does not scale, as we will discuss in Section 7. Instead, the key to achieving the above two goals lies in allowing our knowledges to relate not only values but also continuations (thus rendering the $\mathbf{K}$ relation obsolete). The final model is presented in Figure 3. For now, ignore the budget-related aspects and the parts highlighted in red.

As can be seen, we define local and global knowledges the same as before except that now they can relate both values and continuations (see DRel). In the program equivalence judgment (at the bottom of the figure), we no longer consider the programs in isolation but in the context of (essentially arbitrary) continuations K_1, K_2 drawn from the global knowledge. Of course, we need to adapt the expression relation to account for this. In the new definition of $\mathbf{E}$, the termination case has disappeared. Instead, the $\mathbf{S}$ relation has been extended with a "return" case allowing the two expressions to reduce to related values v_i inside related continuations K_i. Since these K_i may be the initially given continuations, this subsumes the old termination case. Due to this generalization, the $\mathbf{E}$ and $\mathbf{S}$ relations need no longer be type-indexed.

Also observe that, in the "stuck function calls" case, the condition that the continuations of the calls must be related has shifted from $\mathbf{E}$ into $\mathbf{S}$ (explained below) and is now using the continuation knowledge. The earlier continuation relation $\mathbf{K}$, derived from $\mathbf{E}$, has thus disappeared. A remarkable consequence is that the definition of the $\mathbf{E}$ relation is no longer (co)recursive. Coinductive reasoning is now done entirely through the local knowledge and consistency.

The latter is defined the same way as before in terms of $\mathbf{S}$ and $\mathbf{E}$. Of course, the new "return" case in $\mathbf{S}$ results in an additional burden in proving consistency: whenever one relates two continuations in one's local knowledge, one will be forced to prove that they behave equivalently when given related values. Furthermore, in proving consistency for function values, we now apply the applications inside continuations, since the condition about continuations has been shifted from $\mathbf{E}$ to $\mathbf{S}$.

Why the Stutter Budget Matters. We now focus on the budget-related aspects of this model. First of all, observe that the treatment of the budget here very much follows that in the previous model. The only difference is that, since programs are now being run inside continuations, we add the budget of these continuations to that of the programs (in the program equivalence judgment, and, via $\mathbf{S}$, in consistency).

More interesting is why, as we claimed earlier, the stutter budget plays an essential role in facilitating the addition of continuation knowledge. The answer is simple: without the budget, the above contextualization would rule out many basic equivalences, rendering the model all but useless. Assume for a moment that we contextualized the PB model (Section 3) rather than the SPB model (Section 4), *i.e.,* imagine the model in Figure 3 had no budget and was using physical rather than logical reduction. Now consider the example equivalence

$$f : \mathsf{int} \rightarrow \mathsf{int} \vdash \mathsf{inl}\,(f\ 0) \sim \mathsf{inl}\,(f\ 0) : \mathsf{int} + \tau,$$

which certainly ought to hold (for any τ), and let us attempt to prove it. We define the following local knowledge, whose value part is empty:

$$L(R) := \{(\mathsf{int}, K_1[\mathsf{inl}\,\bullet], K_2[\mathsf{inl}\,\bullet]) \mid (K_1, K_2) \in R(\mathsf{int} + \tau)\}$$

Using the function call case in $\mathbf{S}$, it is easy to show that the two open programs are equivalent relative to L. It then remains to establish L's consistency. So suppose $G \in \mathrm{GK}(L)$, $(K_1, K_2) \in G(\mathsf{int} + \tau)$, and $(v_1, v_2) \in \overline{G}(\mathsf{int})$. We must show that there is $(e_1, e_2) \in \mathbf{E}(G)$ with $K_i[\mathsf{inl}\,v_i] \hookrightarrow e_i$. Since $\mathsf{inl}\,v_i$ is a value and we know nothing about K_i, we are stuck. As in the trouble with the eta law, the problem once again is that the guardedness condition of requiring terms—in this case, $K_i[\mathsf{inl}\,v_i]$—to take a *physical* step to e_i is too strong.

Also as before, the solution is to use the stutter budget to support a logical notion of progress, leading to a valid proof of the equivalence via the SPB model from Figure 3.

Proof: Define L as follows.

$$\begin{aligned} L(R) := \{&(\mathsf{int}, k_1', K_1[\mathsf{inl}\,\bullet], k_2', K_2[\mathsf{inl}\,\bullet]) \mid \\ &\exists k_1 < k_1', k_2 < k_2'.\ (k_1, K_1, k_2, K_2) \in R(\mathsf{int} + \tau)\} \end{aligned}$$

We show $f : \mathsf{int} \rightarrow \mathsf{int} \vdash \mathsf{inl}\,(f\ 0) \sim_{L,\mathrm{hd},\mathrm{hd}} \mathsf{inl}\,(f\ 0) : \mathsf{int} + \tau$.

- Suppose $G \in \mathrm{GK}(L)$, $(n_1, v_1, n_2, v_2) \in \overline{G}(\mathsf{int} \rightarrow \mathsf{int})$ and $(k_1, K_1, k_2, K_2) \in G(\mathsf{int} + \tau)$. We must show:
  $$(n_1 + k_1, K_1[\mathsf{inl}\,(v_1\ 0)], n_2 + k_2, K_2[\mathsf{inl}\,(v_2\ 0)]) \in \mathbf{E}(G)$$
- Immediately using the "function call" case in $\mathbf{S}$, it suffices to show $(\_, K_1[\mathsf{inl}\,\bullet], \_, K_2[\mathsf{inl}\,\bullet]) \in G(\mathsf{int})$.
- This follows from $G \supseteq L(G)$ by construction of L.

It remains to prove $\mathrm{consistent}(L)$.

- So suppose $G \in \mathrm{GK}(L)$, $(k_1, K_1, k_2, K_2) \in G(\mathsf{int} + \tau)$, $k_1' > k_1$, $k_2' > k_2$, and $(n_1, v_1, n_2, v_2) \in \overline{G}(\mathsf{int})$.
- To show: $(k_1' + n_1, K_1[\mathsf{inl}\,v_1], k_2' + n_2, K_2[\mathsf{inl}\,v_2]) \in \mathbf{E}(G)^{\$}$
- Taking a stutter step on either side, it suffices to show $(k_1, K_1[\mathsf{inl}\,v_1], k_2, K_2[\mathsf{inl}\,v_2]) \in \mathbf{E}(G)$, which is obvious due to $\mathbf{S}(G, G) \subseteq \mathbf{E}(G)$. $\square$

5.2 Supporting First-Class Continuations

As mentioned before, the contextualization is independent of first-class continuations. Consequently, the model in Figure 3 *excluding* the two parts highlighted in red is still a (sound) model for λ^{μ}. In order to obtain one for $\lambda^{\mu}_{\mathrm{cc}}$, all one then needs to do is include those parts.

The first one extends the definition of the value closure to give meaning to continuation types: two continuation values are related at type $\mathsf{cont}\,\tau$ iff the underlying continuations are related at τ by (the continuation part of) the global knowledge. Similarly to other cases of the value closure, the budget is ignored. The second part is an additional requirement that any global knowledge must relate the empty continuations at the system type int, for any budget. This is necessary in the presence of the isolate construct, which allows a programmer to make up continuations that escape the initial ones.

5.3 *Example:* callcc *in a Loop*

Of course, the eta law still holds in this generalized model. We could show the proof here but it is almost the same as the one in Section 4 (in fact, even the same local knowledge can be used). Instead, we show the proof of an interesting equivalence involving callcc. The example is adapted from Støvring and Lassen [18]. Consider the following two programs:

$$\begin{aligned}
&\tau := \mu\alpha.\, \mathsf{cont\,int} \to \alpha\\
&F_v := \mathsf{fix}\, f(x{:}\tau){:}\mathsf{int}.\, f\ (\mathsf{unroll}\ x\ v)\\
&e_1 := |\lambda y{:}\tau.\, \mathsf{callcc}_{\mathsf{int}}\, (k.\ F_k\ y)|\\
&e_2 := |\mathsf{fix}\, f(x{:}\tau){:}\mathsf{int}.\, \mathsf{callcc}_{\mathsf{int}}\, (k.\ f\ (\mathsf{unroll}\ x\ k))|
\end{aligned}$$

When called, both e_1 and e_2 loop. Both also capture the current continuation. The difference is that e_2 captures it once (namely at the very beginning of its execution), while e_1 does it in every loop iteration. However, since the continuation does not actually change, e_1 is always capturing the same one (namely K_1), and so the two programs behave equivalently.

Formally, in order to prove $\vdash e_1 \sim e_2 : \tau \to \mathsf{int}$ (for $\lambda^{\mu}_{\mathrm{cc}}$), we define the following local knowledge:

$$\begin{aligned}
L(R) := \{&(\tau \to \mathsf{int}, _, e_1, _, e_2)\}\\
\cup\ \{&(\tau, _, K_1[F_{(\mathsf{cont}\ K_1)}\ \bullet], _, K_2[e_2\ \bullet]) \mid\\
&\qquad (_, K_1, _, K_2) \in G(\mathsf{int})\}
\end{aligned}$$

(As obvious from this definition, the budget does not play an interesting role here.) We first show $\vdash e_1 \sim_{L,\tilde{0},\tilde{0}} e_2 : \tau \to \mathsf{int}$.

- Suppose $G \in \mathrm{GK}(L)$ and $(k_1, K_1, k_2, K_2) \in G(\tau \to \mathsf{int})$.
- We must show $(k_1, K_1[e_1], k_2, K_2[e_2]) \in \mathbf{E}(G)$.
- It suffices to show $(0, e_1, 0, e_2) \in \overline{G}(\tau \to \mathsf{int})$, which holds due to $G \supseteq L(G)$.

It remains to show $\mathrm{consistent}(L)$, which has two parts.

1) - Suppose $G \in \mathrm{GK}(L)$, n_1 and n_2 arbitrary, (*) $(\_, v_1, \_, v_2) \in \overline{G}(\tau)$, and $(\_, K_1, \_, K_2) \in G(\mathsf{int})$.
   - To show: $(n_1, K_1[e_1\ v_1], n_2, K_2[e_2\ v_2]) \in \mathbf{E}(G)^{\$}$
   - From (*) we know that there is $(\_, v_1', \_, v_2') \in \overline{G}(\mathsf{cont\,int} \to \tau)$ such that $v_i = \mathsf{roll}\ v_i'$.
   - Taking several physical steps, it thus suffices to show:
   $$\begin{aligned}(&\_, K_1[F_{(\mathsf{cont}\ K_1)}\ (v_1'\ (\mathsf{cont}\ K_1))],\\ &\_, K_2[e_2\ (v_2'\ (\mathsf{cont}\ K_2))]\qquad) \in \mathbf{S}(G, G)\end{aligned}$$
   - Since $(\_, \mathsf{cont}\ K_1, \_, \mathsf{cont}\ K_2) \in \overline{G}(\mathsf{cont\,int})$, this follows with $(\_, K_1[F_{(\mathsf{cont}\ K_1)}\ \bullet], \_, K_2[e_2\ \bullet]) \in G(\tau)$, which holds due to $G \supseteq L(G)$.
2) - Suppose $G \in \mathrm{GK}(L)$, $(\_, K_1, \_, K_2) \in G(\mathsf{int})$, n_1 and n_2 arbitrary, and (*) $(m_1, v_1, m_2, v_2) \in \overline{G}(\tau)$.
   - To show: $(n_1 + m_1, K_1[F_{(\mathsf{cont}\ K_1)}\ v_1],$ $n_2 + m_2, K_2[e_2\ v_2]\qquad) \in \mathbf{E}(G)^{\$}$
   - By (*) there is $(\_, v_1', \_, v_2') \in \overline{G}(\mathsf{cont\,int} \to \tau)$ such that $v_i = \mathsf{roll}\ v_i'$.
   - Taking several physical steps, it thus suffices to show
   $$\begin{aligned}(&\_, K_1[F_{(\mathsf{cont}\ K_1)}\ (v_1'\ (\mathsf{cont}\ K_1))],\\ &\_, K_2[e_2\ (v_2'\ (\mathsf{cont}\ K_2))]\qquad) \in \mathbf{S}(G, G),\end{aligned}$$
   which we do the same as in part (1).

5.4 *Using Parameterized Coinduction*

Writing down a suitable local knowledge at the beginning of a proof can be quite tedious for complex equivalences, even more so in the new model where one generally also has to add continuations. While not really an issue in paper proofs, this quickly becomes very tiresome in formal proofs such as these in our Coq formalization. Fortunately, we can employ *parameterized coinduction* [10, 14] to avoid this issue completely and instead write proofs in an incremental style, where we basically start with a knowledge containing just the programs in question, and extend it as the proof evolves. Indeed, this is how we prove a big part of soundness in Coq.

In order to get there, we need to express the property of a local knowledge being consistent as that local knowledge being a postfixed point of some monotone function. Then the greatest fixed point of that function is automatically the greatest consistent local knowledge, and so we can use the incremental reasoning principle from parameterized coinduction to do proofs about it.

Definition 3. We define the wanted function $\mathfrak{f} \in \mathrm{LK} \overset{\mathrm{mon}}{\to} \mathrm{LK}$.

$$\begin{aligned}\mathfrak{f}(L)(R) := \{&(\tau, n_1, d_1, n_2, d_2) \mid \forall G \in \mathrm{GK}(L).\\ &G \supseteq R \implies \mathbf{S}(\{(\tau, n_1, d_1, n_2, d_2)\}, G) \subseteq \mathbf{E}(G)^{\$}\}\end{aligned}$$

Lemma 2. $L \subseteq \mathfrak{f}(L) \iff \mathrm{consistent}(L)$

6 – Metatheory

We briefly state the main meta-theoretical results for SPBs. They apply to both models for λ^{μ} (the one without continuation knowledge, and the one with), and to the model for $\lambda^{\mu}_{\mathrm{cc}}$. Their proofs pretty much follow those for PBs [8] (but see below for details on transitivity). A number of the compatibility lemmas needed in showing reflexivity and congruence are proven in the appendix. For further proofs, we refer the reader to our Coq formalization of SPBs (in the setting of a richer language).

Theorem 3 (Reflexivity, Symmetry, Transitivity, Congruence)**.**

$$\frac{\Gamma \vdash p : \tau}{\Gamma \vdash |p| \sim |p| : \tau} \qquad \frac{\Gamma \vdash e_2 \sim e_1 : \tau}{\Gamma \vdash e_1 \sim e_2 : \tau}$$

$$\frac{\Gamma \vdash e_1 \sim e_2 : \tau \quad \Gamma \vdash e_2 \sim e_3 : \tau}{\Gamma \vdash e_1 \sim e_3 : \tau}$$

$$\frac{\Gamma' \vdash e_1 \sim e_2 : \tau' \quad \vdash C : (\Gamma'; \tau') \rightsquigarrow (\Gamma; \tau)}{\Gamma \vdash |C|[e_1] \sim |C|[e_2] : \tau}$$

Lemma 4 (Adequacy)**.** If $\vdash e_1 \sim e_2 : \mathsf{int}$, then either both $e_1 \hookrightarrow^\omega$ and $e_2 \hookrightarrow^\omega$, or both $e_1 \hookrightarrow^* n$ and $e_2 \hookrightarrow^* n$ for some integer value n.

Theorem 5 (Soundness)**.**

$$\frac{\Gamma \vdash p_1 : \tau \quad \Gamma \vdash |p_1| \sim |p_2| : \tau \quad \Gamma \vdash p_2 : \tau}{\Gamma \vdash p_1 \sim_{\mathrm{ctx}} p_2 : \tau}$$

6.1 Transitivity

As mentioned in the introduction, transitivity is of particular interest to us. We describe the proof of transitivity for PBs in detail in a technical report [9]. Developing this very complex proof required a lot of effort. Fortunately, adapting it to SPBs does not require many changes. We now briefly discuss the more interesting ones.

The transitivity proof must establish that, given consistent local knowledges L_1, L_2 with $\Gamma \vdash e_1 \sim_{L_1} e_2 : \tau$ and $\Gamma \vdash e_2 \sim_{L_2} e_3 : \tau$, there exists a consistent local knowledge L with $\Gamma \vdash e_1 \sim_L e_3 : \tau$. A key part of this proof is the "correct" decomposition of a global knowledge G for L into $G_{(1)}$ for L_1 and $G_{(2)}$ for L_2, using which L is defined. In the PB case, we have (where $\circ$ is relational composition):

$$L(R) := L_1(R_{(1)}) \circ L_2(R_{(2)})$$

The proof of L's consistency is very tricky. Amongst other things, it relies on the property that certain "bad" values can be related by a global knowledge but not by a local one. This is enforced in the PB model by a restriction on LK: values in a local knowledge must be syntactic functions, in contrast to a global knowledge. However, as explained in Section 4.4, we had to drop this discrimination in SPBs. As a consequence, we can no longer prove the consistency of such defined L.

Fortunately, the stutter budget enables a trick that lets us overcome this problem. The idea is to define L in a slightly different manner:

$$L(R) := (R_{(1)} \circ R_{(2)})^{++}$$

(Here $(-)^{++}$ increments both parts of each tuple's budget by 1.) At a first glance, this seems very odd. Previously, since L was defined as the composition of L_1 and L_2, we could rely on L_1 and L_2's consistency in proving L's. Now, however, L is defined in terms of global knowledges which may contain junk. Fortunately, thanks to increasing the budget in the definition of L, this is not an issue and we can pretty easily prove L consistent. Of course, there is no free lunch, and the price to pay for this parlor trick is that other parts of the transitivity proof become even more subtle than they already were (and the model had to be designed very carefully to make them all go through). We intend to report on the details of this new transitivity proof in a future, extended version of this paper.

7 – Discussion and Related Work

Modeling Higher-Order State and Abstract Types. Following in the footsteps of PBs, it is fairly straightforward to scale both the intermediate SPB model (Section 4) and the final contextualized one (Section 5) to a language with polymorphism, abstract types, and general references. We have proven all the meta-theoretic results of the previous section for this full model, and mechanized the proofs in Coq. Our appendix presents the definition of this full model, together with an example application of it (namely, Dreyer *et al.*'s challenging "well-bracketed state change" example [6]).

Concerning the extensions to abstract types and state, perhaps the only significant difference between SPBs and PBs is a (minor) generalization of local knowledges, necessitated by SPBs' relating of continuations in the local/global knowledge. In the full PB model, both local and global knowledges were indexed by states of a transition system used to describe invariants on a term's local mutable state. This holds true for the full SPB model as well. The difference is that in PBs, a local knowledge was only able to query the global knowledge at the "current" state of the transition system, whereas in SPBs, when defining a local knowledge, it is important to be able to query whether some other continuations were related by the global knowledge in *previous* states of the transition system.

Contextualization. In Section 5 we commented that using the $\mathbf{K}$ relation in order to contextualize the model does not scale. In particular, one might expect to see something like (ignoring budget and types)

$$\forall (K_1, K_2) \in \mathbf{K}(G).\ (K_1[\gamma_1 e_1], K_2[\gamma_2 e_2]) \in \mathbf{E}(G)$$

in the definition of program equivalence (and similarly for consistency). That is, one would consider the behavior of programs when run inside continuations related by $\mathbf{K}$. To understand why this approach does not scale, it is important to understand that in the model for the full language with state, equivalence is relative not only to a local knowledge, but to a whole *world* [1, 6, 8] that, amongst other things, limits the part of the heap that the programs may access. In particular, both the local term equivalence relation, $\mathbf{E}$, and the continuation relation, $\mathbf{K}$, require programs to conform to the world and are therefore *indexed* by a world W.

One key property then is that programs stay equivalent when the world is extended, *i.e.,* when access to additional parts of the heap is granted (intuitively, because the programs do not care about these parts). This is where things go wrong when using the contextualization sketched above. Assume two programs are equivalent in a world W, meaning (according to the definition sketched above) that they are related inside any continuations from $\mathbf{K}_W$. We would now have to show that the programs are also related when run inside continuations from $\mathbf{K}_{W'}$, for any larger world W'. However, since $\mathbf{K}_{W'}$ generally contains more continuations than $\mathbf{K}_W$—after all, those in $\mathbf{K}_{W'}$ may access a larger part of the heap than those in $\mathbf{K}_W$—this is clearly impossible to prove.

Well-Founded Bisimulations. Our technique of logical reduction is inspired by Namjoshi's *well-founded* [15, 13] bisimulations, which were developed as an alternative formulation of *stuttering* bisimulations [4] that can be checked by only reasoning about single transitions instead of infinite computations. In order to support finite but unbounded stuttering, well-founded bisimulations employ a "rank function" mapping states to some well-founded ordering, and insist (roughly) that, for states related in a bisimulation, either both make physical transitions to related states or else one side makes a transition while the rank of the pair of states decreases. In our model, we use the stutter budget to effectively bake a particular rank function into our bisimulations, which is sufficient for our purposes and convenient to work with. As far as we aware, this is the first time that the idea of well-founded bisimulations has been adapted for use in reasoning about open programs in a higher-order language setting.

Relational Reasoning About ML-Like Languages. As explained in the introduction, many methods have been developed for reasoning about equivalence in ML-like languages, but the most powerful methods to date (at least practically speaking, in terms of providing effective proof techniques) are step-indexed Kripke logical relations methods [1, 6, 7] and various bisimulation methods [11, 17, 19]. We proposed PBs [8] as a way of synthesizing, as it were, the best of both worlds, but our previous formulation of them was lacking in (1) its invalidation of the eta law, and (2) its inability to model control effects. Thus, for all their virtues, PBs could not be claimed to subsume other methods, since indeed they could not prove as many equivalences as, say, Dreyer *et al.*'s SKLRs [6], which are compatible with both control effects and eta.

In this paper, we have shown how the idea of logical reduction semantics employed by stuttering parametric bisimulations (SPBs) rectifies the inadequacies of PBs, thus offering a proof method on par with the state of the art. In addition, like PBs, SPBs continue to offer distinct advantages over SKLR and bisimulation methods, due to their transitivity and semantic nature, respectively. We hope to exploit those advantages in our future work on compositional compiler certification.

Acknowledgments

The research was partially supported by the EC FET project ADVENT.

References

[1] A. Ahmed, D. Dreyer, and A. Rossberg, "State-dependent representation independence," in *POPL*, 2009.

[2] N. Benton and C.-K. Hur, "Biorthogonality, step-indexing and compiler correctness," in *ICFP*, 2009.

[3] L. Birkedal and R. W. Harper, "Constructing interpretations of recursive types in an operational setting," *Information and Computation*, vol. 155, pp. 3–63, 1999.

[4] M. C. Browne, E. M. Clarke, and O. Grümberg, "Characterizing finite Kripke structures in propositional temporal logic," *Theor. Comput. Sci.*, vol. 59, no. 1-2, pp. 115–131, Jul. 1988.

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

[6] D. Dreyer, G. Neis, and L. Birkedal, "The impact of higher-order state and control effects on local relational reasoning," *J. Funct. Program.*, vol. 22, no. 4-5, pp. 477–528, 2012.

[7] C.-K. Hur and D. Dreyer, "A Kripke logical relation between ML and assembly," in *POPL*, 2011.

[8] C.-K. Hur, D. Dreyer, G. Neis, and V. Vafeiadis, "The marriage of bisimulations and Kripke logical relations," in *POPL*, 2012.

[9] C.-K. Hur, G. Neis, D. Dreyer, and V. Vafeiadis, "The transitive composability of relation transition systems," MPI-SWS, Tech. Rep. MPI-SWS-2012-002, May 2012.

[10] C.-K. Hur, G. Neis, D. Dreyer, and V. Vafeiadis, "The power of parameterization in coinductive proof," in *POPL*, 2013.

[11] V. Koutavas and M. Wand, "Small bisimulations for reasoning about higher-order imperative programs," in *POPL*, 2006.

[12] S. B. Lassen and P. B. Levy, "Typed normal form bisimulation for parametric polymorphism," in *LICS*, 2008.

[13] P. Manolios, "Mechanical verification of reactive systems," Ph.D. Thesis, Department of Computer Science, The University of Texas at Austin, Aug. 2001.

[14] L. S. Moss, "Parametric corecursion," *Theor. Comput. Sci.*, vol. 260, no. 1-2, pp. 139–163, Jun. 2001.

[15] K. S. Namjoshi, "A simple characterization of stuttering bisimulation," in *FSTTCS*, 1997, pp. 284–296.

[16] D. Sangiorgi, "The lazy lambda calculus in a concurrency scenario," *Information and Computation*, vol. 111, no. 1, pp. 120–153, 1994.

[17] D. Sangiorgi, N. Kobayashi, and E. Sumii, "Environmental bisimulations for higher-order languages," *TOPLAS*, vol. 33, no. 1, pp. 1–69, Jan. 2011.

[18] K. Støvring and S. Lassen, "A complete, co-inductive syntactic theory of sequential control and state," in *POPL*, 2007.

[19] E. Sumii, "A complete characterization of observational equivalence in polymorphic λ-calculus with general references," in *CSL*, 2009.

[20] E. Sumii and B. Pierce, "A bisimulation for type abstraction and recursion," *JACM*, vol. 54, no. 5, pp. 1–43, Oct. 2007.

APPENDIX A
SPBS FOR λ^μ AND $\lambda^\mu_{\mathrm{Cc}}$

A.1 Languages λ^μ and $\lambda^\mu_{\mathbf{cc}}$

The fragment λ^μ is obtained by removing first-class continuations (the parts highlighted in red).

A.1.1 Statics.

$$\begin{array}{rl} \sigma,\tau \in \mathrm{Ty} ::= & \alpha \mid \mathsf{int} \mid \sigma_1 \times \sigma_2 \mid \sigma_1 + \sigma_2 \mid \sigma_1 \to \sigma_2 \mid \mu\alpha.\,\sigma \mid \mathsf{cont}\,\sigma \\ p \in \mathrm{Prog} ::= & x \mid n \mid p_1 \odot p_2 \mid \mathsf{ifz}\ p\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2 \mid \langle p_1, p_2\rangle \mid p.1 \mid p.2 \mid \mathsf{inl}_\sigma\, p \mid \mathsf{inr}_\sigma\, p \mid \\ & (\mathsf{case}\, p\, \mathsf{of}\ \mathsf{inl}\, x \Rightarrow p_1 \mid \mathsf{inr}\, x \Rightarrow p_2) \mid \mathsf{fix}\, f(x{:}\sigma_1){:}\sigma_2.\, p \mid p_1\ p_2 \mid \mathsf{roll}_\sigma\ p \mid \mathsf{unroll}\ p \mid \\ & \mathsf{callcc}_\sigma\,(x.\ p) \mid \mathsf{throw}_\sigma\ p_1\ \mathsf{to}\ p_2 \mid \mathsf{isolate}\ p \end{array}$$

$$\text{Term environments} \quad \Gamma \quad ::= \quad \cdot \mid \Gamma, x{:}\tau$$

$\boxed{\Gamma \vdash p : \tau}$

$$\frac{\vdash \Gamma \quad x{:}\tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\vdash \Gamma}{\Gamma \vdash n : \mathsf{int}}$$

$$\frac{\Gamma \vdash p_1 : \mathsf{int} \quad \Gamma \vdash p_2 : \mathsf{int}}{\Gamma \vdash p_1 \odot p_2 : \mathsf{int}} \qquad \frac{\Gamma \vdash p_0 : \mathsf{int} \quad \Gamma \vdash p_1 : \tau \quad \Gamma \vdash p_2 : \tau}{\Gamma \vdash \mathsf{ifz}\ p_0\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2 : \tau}$$

$$\frac{\Gamma \vdash p_1 : \tau_1 \quad \Gamma \vdash p_2 : \tau_2}{\Gamma \vdash \langle p_1, p_2\rangle : \tau_1 \times \tau_2} \qquad \frac{\Gamma \vdash p : \tau_1 \times \tau_2}{\Gamma \vdash p.1 : \tau_1} \qquad \frac{\Gamma \vdash p : \tau_1 \times \tau_2}{\Gamma \vdash p.2 : \tau_2}$$

$$\frac{\Gamma \vdash p : \tau_1}{\Gamma \vdash \mathsf{inl}_{\tau_2}\, p : \tau_1 + \tau_2} \qquad \frac{\Gamma \vdash p : \tau_2}{\Gamma \vdash \mathsf{inr}_{\tau_1}\, p : \tau_1 + \tau_2}$$

$$\frac{\Gamma \vdash p : \tau_1 + \tau_2 \quad \Gamma, x{:}\tau_1 \vdash p_1 : \tau \quad \Gamma, x{:}\tau_2 \vdash p_2 : \tau}{\Gamma \vdash \mathsf{case}\, p\, \mathsf{of}\ \mathsf{inl}\, x \Rightarrow p_1 \mid \mathsf{inr}\, x \Rightarrow p_2 : \tau}$$

$$\frac{\Gamma, f{:}(\tau_1 \to \tau_2), x{:}\tau_1 \vdash p : \tau_2}{\Gamma \vdash \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.\, p : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash p_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash p_2 : \tau_1}{\Gamma \vdash p_1\ p_2 : \tau_2}$$

$$\frac{\Gamma \vdash p : \sigma[\mu\alpha.\,\sigma/\alpha]}{\Gamma \vdash \mathsf{roll}_{\mu\alpha.\,\sigma}\ p : \mu\alpha.\,\sigma} \qquad \frac{\Gamma \vdash p : \mu\alpha.\,\sigma}{\Gamma \vdash \mathsf{unroll}\ p : \sigma[\mu\alpha.\,\sigma/\alpha]}$$

$$\frac{\Gamma, x{:}\mathsf{cont}\,\tau \vdash p : \tau}{\Gamma \vdash \mathsf{callcc}_\tau\,(x.\ p) : \tau} \qquad \frac{\Gamma \vdash p' : \tau' \quad \Gamma \vdash p : \mathsf{cont}\,\tau'}{\Gamma \vdash \mathsf{throw}_\tau\ p'\ \mathsf{to}\ p : \tau} \qquad \frac{\Gamma \vdash p : \tau \to \mathsf{int}}{\Gamma \vdash \mathsf{isolate}\ p : \mathsf{cont}\,\tau}$$

A.1.2 Dynamics.

$$
\begin{array}{lcl}
v \in \mathrm{Val} & ::= & x \mid n \mid \langle v_1, v_2\rangle \mid \mathsf{inl}\ v \mid \mathsf{inr}\ v \mid \mathsf{fix}\ f(x).\,e \mid \mathsf{roll}\ v \mid \mathsf{cont}\ K \\
e \in \mathrm{Exp} & ::= & v \mid e_1 \odot e_2 \mid \mathsf{ifz}\ e_0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \langle e_1, e_2\rangle \mid e.1 \mid e.2 \mid \mathsf{inl}\ e \mid \mathsf{inr}\ e \mid \\
 & & (\mathsf{case}\ e\ \mathsf{of}\ \mathsf{inl}\ x \Rightarrow e_1 \mid \mathsf{inr}\ x \Rightarrow e_2) \mid e_1\ e_2 \mid \mathsf{roll}\ e \mid \mathsf{unroll}\ e \mid \\
 & & \mathsf{callcc}\ (x.\ e) \mid \mathsf{throw}\ e_1\ \mathsf{to}\ e_2 \mid \mathsf{isolate}\ e \\
K \in \mathrm{Cont} & ::= & \bullet \mid K \odot e \mid v \odot K \mid \mathsf{ifz}\ K\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \langle K, e\rangle \mid \langle v, K\rangle \mid K.1 \mid K.2 \mid \\
 & & \mathsf{inl}\ K \mid \mathsf{inr}\ K \mid (\mathsf{case}\ K\ \mathsf{of}\ \mathsf{inl}\ x \Rightarrow e_1 \mid \mathsf{inr}\ x \Rightarrow e_2) \mid K\ e \mid v\ K \mid \mathsf{roll}\ K \mid \\
 & & \mathsf{unroll}\ K \mid \mathsf{throw}\ K\ \mathsf{to}\ e \mid \mathsf{throw}\ v\ \mathsf{to}\ K \mid \mathsf{isolate}\ K
\end{array}
$$

$\boxed{e \hookrightarrow e}$

$$
\begin{array}{lcll}
K[n_1 \odot n_2] & \hookrightarrow & K[n] & (n = [\![n_1 \odot n_2]\!]) \\
K[\mathsf{ifz}\ 0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2] & \hookrightarrow & K[e_1] & \\
K[\mathsf{ifz}\ n\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2] & \hookrightarrow & K[e_2] & (n \neq 0) \\
K[\langle v_1, v_2\rangle.1] & \hookrightarrow & K[v_1] & \\
K[\langle v_1, v_2\rangle.2] & \hookrightarrow & K[v_2] & \\
K[\mathsf{case}\ \mathsf{inl}\ v\ \mathsf{of}\ \mathsf{inl}\ x \Rightarrow e_1 \mid \mathsf{inr}\ x \Rightarrow e_2] & \hookrightarrow & K[e_1[v/x]] & \\
K[\mathsf{case}\ \mathsf{inr}\ v\ \mathsf{of}\ \mathsf{inl}\ x \Rightarrow e_1 \mid \mathsf{inr}\ x \Rightarrow e_2] & \hookrightarrow & K[e_2[v/x]] & \\
K[(\mathsf{fix}\ f(x).\,e)\ v] & \hookrightarrow & K[e[(\mathsf{fix}\ f(x).\,e)/f, v/x]] & \\
K[\mathsf{unroll}\ (\mathsf{roll}\ v)] & \hookrightarrow & K[v] & \\
K[\mathsf{callcc}\ (x.\ e)] & \hookrightarrow & K[e[\mathsf{cont}\ K/x]] & \\
K[\mathsf{throw}\ v\ \mathsf{to}\ \mathsf{cont}\ K'] & \hookrightarrow & K'[v] & \\
K[\mathsf{isolate}\ v] & \hookrightarrow & K[\mathsf{cont}\ (v\ \bullet)] &
\end{array}
$$

A.2 Simple SPB Model for λ^μ

A.2.1 Definition.

$\boxed{n, e \hookrightarrow n', e'}$

$$
\frac{e \hookrightarrow e'}{n, e \hookrightarrow n', e'} \qquad \frac{n' < n}{n, e \hookrightarrow n', e}
$$

$$
\begin{array}{l}
\mathrm{CVal} := \{v \in \mathrm{Val} \mid \mathrm{fv}(v) = \emptyset\} \\
\mathrm{CExp} := \{e \in \mathrm{Exp} \mid \mathrm{fv}(e) = \emptyset\} \\
\mathrm{CCont} := \{K \in \mathrm{Cont} \mid \mathrm{fv}(K[\langle\rangle]) = \emptyset\} \\
\mathrm{CTy} := \{\tau \in \mathrm{Ty} \mid \mathrm{fv}(\tau) = \emptyset\} \\
\mathrm{CTyF} := \{\tau_1 \to \tau_2 \mid \tau_1, \tau_2 \in \mathrm{CTy}\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{VRelF} := \mathbb{P}(\mathrm{CTyF} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \\
\mathrm{KRel} := \mathbb{P}(\mathrm{CTy} \times \mathrm{CTy} \times \mathbb{N} \times \mathrm{CCont} \times \mathbb{N} \times \mathrm{CCont}) \\
\mathrm{VRel} := \mathbb{P}(\mathrm{CTy} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \\
\mathrm{ERel} := \mathbb{P}(\mathrm{CTy} \times \mathbb{N} \times \mathrm{CExp} \times \mathbb{N} \times \mathrm{CExp})
\end{array}
$$

$$
\begin{array}{lcl}
\multicolumn{3}{l}{\overline{(-)} \in \mathrm{VRelF} \to \mathrm{VRel}} \\
\overline{R}(\mathsf{int}) & := & \{(n_1, m, n_2, m)\} \\
\overline{R}(\tau \times \tau') & := & \{(n_1, \langle v_1, v_1'\rangle, n_2, \langle v_2, v_2'\rangle) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\tau)\} \\
 & \cap & \{(n_1, \langle v_1, v_1'\rangle, n_2, \langle v_2, v_2'\rangle) \mid \exists m_1', m_2'.\ (m_1', v_1', m_2', v_2') \in \overline{R}(\tau')\} \\
\overline{R}(\tau + \tau') & := & \{(n_1, \mathsf{inl}\ v_1, n_2, \mathsf{inl}\ v_2) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\tau)\} \\
 & \cup & \{(n_1, \mathsf{inr}\ v_1', n_2, \mathsf{inr}\ v_2') \mid \exists m_1', m_2'.\ (m_1', v_1', m_2', v_2') \in \overline{R}(\tau')\} \\
\overline{R}(\tau \to \tau') & := & \{(n_1, v_1, n_2, v_2) \in R(\tau \to \tau')\} \\
\overline{R}(\mu\alpha.\,\sigma) & := & \{(n_1, \mathsf{roll}\ v_1, n_2, \mathsf{roll}\ v_2) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\sigma[\mu\alpha.\,\sigma/\alpha])\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{LK} := \{L \in \mathrm{VRelF} \to \mathrm{VRelF} \mid \forall R.\ \forall (\tau, n_1, v_1, n_2, v_2) \in L(R). \\
\qquad\qquad \forall R' \supseteq R, n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', v_1, n_2', v_2) \in L(R')\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{GK}(L) := \{G \in \mathrm{VRelF} \mid L(G) \subseteq G\ \wedge \\
\qquad\qquad \forall (\tau, n_1, v_1, n_2, v_2) \in G.\ \forall n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', v_1, n_2', v_2) \in G\}
\end{array}
$$

$$
\begin{aligned}
&\mathrm{S} \in \mathrm{VRelF} \times \mathrm{VRelF} \to \mathrm{ERel}\\
&\mathrm{S}(R,G)(\tau) := \{(n_1, v_1\ v_1', n_2, v_2\ v_2') \mid\\
&\qquad\qquad \exists \tau', m_1, m_2.\ (n_1, v_1, n_2, v_2) \in R(\tau' \to \tau) \wedge (m_1, v_1', m_2, v_2') \in \overline{G}(\tau')\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{E} \in \mathrm{VRelF} \times \mathrm{VRelF} \to \mathrm{ERel}\\
&\mathrm{E}(R,G)(\tau) := \{(n_1, e_1, n_2, e_2) \mid (e_1 \hookrightarrow^\omega \wedge e_2 \hookrightarrow^\omega) \vee\\
&\qquad (\exists (n_1', v_1, n_2', v_2) \in \overline{G}(\tau).\ n_1, e_1 \hookrightarrow^* n_1', v_1 \wedge n_2, e_2 \hookrightarrow^* n_2', v_2) \vee\\
&\qquad (\exists (\tau', n_1', e_1', n_2', e_2') \in \mathrm{S}(R,G).\ \exists (k_1, K_1, k_2, K_2) \in \mathrm{K}(R,G)(\tau', \tau).\\
&\qquad\qquad n_1, e_1 \hookrightarrow^* n_1', K_1[e_1'] \wedge n_2, e_2 \hookrightarrow^* n_2', K_2[e_2'])\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{K} \in \mathrm{VRelF} \times \mathrm{VRelF} \to \mathrm{KRel}\\
&\mathrm{K}(R,G)(\tau', \tau) := \{(k_1, K_1, k_2, K_2) \mid \forall (n_1, v_1, n_2, v_2) \in \overline{G}(\tau').\\
&\qquad (k_1 + n_1, K_1[v_1], k_2 + n_2, K_2[v_2]) \in \mathrm{E}(R,G)(\tau)\}
\end{aligned}
$$

$$
R^{\$} := \{(\tau, n_1, e_1, n_2, e_2) \mid \exists (n_1', e_1', n_2', e_2') \in R(\tau).\ n_1, e_1 \hookrightarrow n_1', e_1' \wedge n_2, e_2 \hookrightarrow n_2', e_2'\}
$$

$$
\mathrm{consistent}(L) \ := \ \forall G \in \mathrm{GK}(L).\ \mathrm{S}(L(G), G) \subseteq \mathrm{E}(G,G)^{\$}
$$

$$
\Gamma \vdash e_1 \sim e_2 : \tau \ := \ \exists L, f_1, f_2.\ \mathrm{consistent}(L) \wedge \Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau
$$

$$
\begin{aligned}
&\Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau \ := \ \forall G \in \mathrm{GK}(L).\ \forall (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma).\\
&\qquad (f_1(N_1), \gamma_1 e_1, f_2(N_2), \gamma_2 e_2) \in \mathrm{E}(G,G)(\tau)\}
\end{aligned}
$$

$$
\begin{aligned}
&R(\cdot) := \{(\cdot, \mathrm{id}, \cdot, \mathrm{id})\}\\
&R(x{:}\tau, \Gamma) := \{(n_1 :: N_1, \gamma_1[x \mapsto v_1], n_2 :: N_2, \gamma_2[x \mapsto v_2]) \mid\\
&\qquad (n_1, v_1, n_2, v_2) \in R(\tau) \wedge (N_1, \gamma_1, N_2, \gamma_2) \in R(\Gamma)\}
\end{aligned}
$$

A.2.2 Properties.

Lemma 6. If

1) $(\tau, n_1', e_1', n_2', e_2') \in \mathrm{E}(R,G)$
2) $n_1, e_1 \hookrightarrow^* n_1', e_1'$
3) $n_2, e_2 \hookrightarrow^* n_2', e_2'$

then $(\tau, n_1, e_1, n_2, e_2) \in \mathrm{E}(R,G)$.

Lemma 7.

- $\mathrm{GK}(L) \cap \mathrm{GK}(L') = \mathrm{GK}(L \cup L')$
- $\mathrm{S}(R,G) \cup \mathrm{S}(R',G) = \mathrm{S}(R \cup R', G)$
- $\mathrm{consistent}(L) \wedge \mathrm{consistent}(L') \implies \mathrm{consistent}(L \cup L')$
- $\Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau \implies \Gamma \vdash e_1 \sim_{L \cup L', f_1, f_2} e_2 : \tau$

Definition 4.

$$
\mathrm{close}(R) := \{(\tau, n_1, e_1, n_2, e_2) \mid \exists m_1 \leq n_1, m_2 \leq n_2.\ (\tau, m_1, e_1, m_2, e_2) \in R\}
$$

Definition 5.

$$
n(N) := n \qquad (f + f')(N) := f(N) + f'(N)
$$

Lemma 8.

$$
\forall \tau, k_1, k_2, G.\ (\tau, \tau, k_1, \bullet, k_2, \bullet) \in \mathrm{K}(G)
$$

A.2.3 **Example: Eta Equivalence (lambda version)**.

$$e_1 := |\lambda f{:}\tau \to \tau'.\, \lambda x{:}\tau.\, f\ x|$$
$$e_2 := |\lambda f{:}\tau \to \tau'.\, f|$$

In order to prove $\vdash e_1 \sim e_2 : (\tau \to \tau') \to \tau \to \tau'$ for $\lambda^{\mu}_{\mathsf{cc}}$ (and thus λ^{μ}), we define a suitable local knowledge L as follows:

$$\begin{aligned} L(G) := \quad & \mathrm{close}\{((\tau \to \tau') \to \tau \to \tau', 0, e_1, 0, e_2)\} \\ \cup \quad & \mathrm{close}\{(\tau \to \tau', 0, \lambda x.\, v_1\ x, n_2 + 1, v_2) \mid (\_, v_1, n_2, v_2) \in G(\tau \to \tau')\} \end{aligned}$$

We first show $\vdash e_1 \sim_{L,0,0} e_2 : (\tau \to \tau') \to \tau \to \tau'$:

- Suppose $G \in \mathrm{GK}(L)$.
- We must show $(0, e_1, 0, e_2) \in \mathrm{E}(G, G)((\tau \to \tau') \to \tau \to \tau')$.
- It suffices to show $(0, e_1, 0, e_2) \in G((\tau \to \tau') \to \tau \to \tau')$, which holds due to $G \supseteq L(G)$.

It remains to show consistent(L):

1) - Suppose $G \in \mathrm{GK}(L)$, $m_1 \geq 0$, $m_2 \geq 0$, and $(\_, v_1, \_, v_2) \in \overline{G}(\tau \to \tau')$.
   - We must show $(m_1, e_1\ v_1, m_2, e_2\ v_2) \in \mathrm{E}(G, G)^{\$}(\tau \to \tau')$.
   - It suffices to show $(\_, \lambda x.\, v_1\ x, \_, v_2) \in \overline{G}(\tau \to \tau')$, which holds due to $G \supseteq L(G)$.
2) - Suppose $G \in \mathrm{GK}(L)$, $(\_, v_1, n_2, v_2) \in G(\tau \to \tau')$, $m_1 \geq 0$, $m_2 \geq 0$, and $(\_, v_1', \_, v_2') \in \overline{G}(\tau)$.
   - We must show $(m_1, (\lambda x.\, v_1\ x)\ v_1', n_2 + 1 + m_2, v_2\ v_2') \in \mathrm{E}(G, G)^{\$}(\tau')$.
   - It suffices to show $(\_, v_1\ v_1', n_2 + m_2, v_2\ v_2') \in \mathrm{E}(G, G)(\tau')$.
   - With Lemma 8, it suffices to show $(\_, v_1\ v_1', n_2, v_2\ v_2') \in \mathrm{S}(G, G)(\tau')$, which follows with $(\_, v_1, n_2, v_2) \in G(\tau \to \tau')$.

A.3 **SPB Models for λ^{μ} and $\lambda^{\mu}_{\mathsf{cc}}$**

The model for λ^{μ} is obtained by excluding the parts highlighted in red.

A.3.1 **Definition**.

$\boxed{n, e \hookrightarrow n', e'}$

$$\frac{e \hookrightarrow e'}{n, e \hookrightarrow n', e'} \qquad \frac{n' < n}{n, e \hookrightarrow n', e}$$

$$\begin{aligned} &\mathrm{CVal} := \{v \in \mathrm{Val} \mid \mathrm{fv}(v) = \emptyset\} \\ &\mathrm{CExp} := \{e \in \mathrm{Exp} \mid \mathrm{fv}(e) = \emptyset\} \\ &\mathrm{CCont} := \{K \in \mathrm{Cont} \mid \mathrm{fv}(K[\langle\rangle]) = \emptyset\} \\ &\mathrm{CTy} := \{\tau \in \mathrm{Ty} \mid \mathrm{fv}(\tau) = \emptyset\} \\ &\mathrm{CTyF} := \{\tau_1 \to \tau_2 \mid \tau_1, \tau_2 \in \mathrm{CTy}\} \end{aligned}$$

$$\begin{aligned} &\mathrm{DRel} := \mathbb{P}((\mathrm{CTyF} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \uplus (\mathrm{CTy} \times \mathbb{N} \times \mathrm{CCont} \times \mathbb{N} \times \mathrm{CCont})) \\ &\mathrm{VRel} := \mathbb{P}(\mathrm{CTy} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \\ &\mathrm{ERel} := \mathbb{P}(\mathbb{N} \times \mathrm{CExp} \times \mathbb{N} \times \mathrm{CExp}) \end{aligned}$$

$$\begin{aligned} &\overline{(-)} \in \mathrm{DRel} \to \mathrm{VRel} \\ &\overline{R}(\mathsf{int}) && := \{(n_1, m, n_2, m)\} \\ &\overline{R}(\tau \times \tau') && := \{(n_1, \langle v_1, v_1' \rangle, n_2, \langle v_2, v_2' \rangle) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\tau)\} \\ & && \cap\ \{(n_1, \langle v_1, v_1' \rangle, n_2, \langle v_2, v_2' \rangle) \mid \exists m_1', m_2'.\ (m_1', v_1', m_2', v_2') \in \overline{R}(\tau')\} \\ &\overline{R}(\tau + \tau') && := \{(n_1, \mathsf{inl}\, v_1, n_2, \mathsf{inl}\, v_2) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\tau)\} \\ & && \cup\ \{(n_1, \mathsf{inr}\, v_1', n_2, \mathsf{inr}\, v_2') \mid \exists m_1', m_2'.\ (m_1', v_1', m_2', v_2') \in \overline{R}(\tau')\} \\ &\overline{R}(\tau \to \tau') && := \{(n_1, v_1, n_2, v_2) \in R(\tau \to \tau')\} \\ &\overline{R}(\mu\alpha.\, \sigma) && := \{(n_1, \mathsf{roll}\, v_1, n_2, \mathsf{roll}\, v_2) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\sigma[\mu\alpha.\, \sigma/\alpha])\} \\ &\overline{R}(\mathsf{cont}\,\tau) && := \{(n_1, \mathsf{cont}\ K_1, n_2, \mathsf{cont}\ K_2) \mid \exists m_1, m_2.\ (m_1, K_1, m_2, K_2) \in R(\tau)\} \end{aligned}$$

$$\begin{aligned} \mathrm{LK} := \{L \in \mathrm{DRel} \to \mathrm{DRel} \mid\ & \forall R.\ \forall(\tau, n_1, d_1, n_2, d_2) \in L(R). \\ & \forall R' \supseteq R, n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', d_1, n_2', d_2) \in L(R')\} \end{aligned}$$

$$\begin{aligned} \mathrm{GK}(L) := \{G \in \mathrm{DRel} \mid\ & L(G) \subseteq G \wedge (0, \bullet, 0, \bullet) \in G(\mathsf{int}) \wedge \\ & \forall(\tau, n_1, d_1, n_2, d_2) \in G.\ \forall n_1' \geq n_1, n_2' \geq n_2.\ (\tau, n_1', d_1, n_2', d_2) \in G\} \end{aligned}$$

$$
\begin{aligned}
&\mathrm{S} \in \mathrm{DRel} \times \mathrm{DRel} \to \mathrm{ERel}\\
&\mathrm{S}(R, G) := \{(k_1 + n_1, K_1[v_1], k_2 + n_2, K_2[v_2]) \mid\\
&\qquad\qquad \exists \tau.\ (k_1, K_1, k_2, K_2) \in R(\tau) \wedge (n_1, v_1, n_2, v_2) \in \overline{G}(\tau)\} \cup\\
&\qquad \{(n_1, K_1[v_1\ v_1'], n_2, K_2[v_2\ v_2']) \mid\\
&\qquad\qquad \exists \tau, \tau', m_1, m_2, k_1, k_2.\ (n_1, v_1, n_2, v_2) \in R(\tau' \to \tau) \wedge\\
&\qquad\qquad (m_1, v_1', m_2, v_2') \in \overline{G}(\tau') \wedge (k_1, K_1, k_2, K_2) \in G(\tau)\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{E} \in \mathrm{DRel} \to \mathrm{ERel}\\
&\mathrm{E}(G) := \{(n_1, e_1, n_2, e_2) \mid (e_1 \hookrightarrow^{\omega} \wedge e_2 \hookrightarrow^{\omega}) \vee\\
&\qquad \exists (n_1', e_1', n_2', e_2') \in \mathrm{S}(G, G).\ n_1, e_1 \hookrightarrow^{*} n_1', e_1' \wedge n_2, e_2 \hookrightarrow^{*} n_2', e_2'\}
\end{aligned}
$$

$$
R^{\$} \ := \ \{(n_1, e_1, n_2, e_2) \mid n_1, n_2 > 0 \wedge (n_1 - 1, e_1, n_2 - 1, e_2) \in R\}
$$

$$
\mathrm{consistent}(L) \ := \ \forall G \in \mathrm{GK}(L).\ \mathrm{S}(L(G), G) \subseteq \mathrm{E}(G)^{\$}
$$

$$
\Gamma \vdash e_1 \sim e_2 : \tau \ := \ \exists L, f_1, f_2.\ \mathrm{consistent}(L) \wedge \Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau
$$

$$
\begin{aligned}
&\Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau \ := \ \forall G \in \mathrm{GK}(L).\ \forall (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma).\\
&\qquad \forall (k_1, K_1, k_2, K_2) \in G(\tau).\ (f_1(N_1) + k_1, K_1[\gamma_1 e_1], f_2(N_2) + k_2, K_2[\gamma_2 e_2]) \in \mathrm{E}(G)\}
\end{aligned}
$$

$$
\begin{aligned}
&R(\cdot) := \{(\cdot, \mathrm{id}, \cdot, \mathrm{id})\}\\
&R(x{:}\tau, \Gamma) := \{(n_1 :: N_1, \gamma_1[x \mapsto v_1], n_2 :: N_2, \gamma_2[x \mapsto v_2]) \mid\\
&\qquad (n_1, v_1, n_2, v_2) \in R(\tau) \wedge (N_1, \gamma_1, N_2, \gamma_2) \in R(\Gamma)\}
\end{aligned}
$$

A.3.2 Properties.

Lemma 9. If

1) $(n_1', e_1', n_2', e_2') \in \mathrm{E}(G)$
2) $n_1, e_1 \hookrightarrow^{*} n_1', e_1'$
3) $n_2, e_2 \hookrightarrow^{*} n_2', e_2'$

then $(n_1, e_1, n_2, e_2) \in \mathrm{E}(G)$.

Lemma 10.

- $\mathrm{GK}(L) \cap \mathrm{GK}(L') = \mathrm{GK}(L \cup L')$
- $\mathrm{S}(R, G) \cup \mathrm{S}(R', G) = \mathrm{S}(R \cup R', G)$
- $\mathrm{consistent}(L) \wedge \mathrm{consistent}(L') \implies \mathrm{consistent}(L \cup L')$
- $\Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau \implies \Gamma \vdash e_1 \sim_{L \cup L', f_1, f_2} e_2 : \tau$

Definition 6.

$$
\mathrm{close}(R) := \{(\tau, n_1, e_1, n_2, e_2) \mid \exists m_1 \leq n_1, m_2 \leq n_2.\ (\tau, m_1, e_1, m_2, e_2) \in R\}
$$

Definition 7.

$$
n(N) := n \qquad (f + f')(N) := f(N) + f'(N)
$$

Lemma 11.

$$
\frac{\vdash \Gamma \quad x{:}\tau \in \Gamma}{\Gamma \vdash x \sim x : \tau}
$$

Proof:

- Let i be the index of $x{:}\tau$ in Γ.
- We show $\Gamma \vdash x \sim_{\emptyset, \Pi_i, \Pi_i} x : \tau$.
- So assume $G \in \mathrm{GK}(\emptyset)$ and $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$ and $(k_1, K_1, k_2, K_2) \in G(\tau)$.
- We must show $(\Pi_i(N_1) + k_1, K_1[\gamma_1 x], \Pi_i(N_2) + k_2, K_2[\gamma_2 x]) \in \mathrm{E}(G)$.
- It suffices to show $(\Pi_i(N_1) + k_1, K_1[\gamma_1 x], \Pi_i(N_2) + k_2, K_2[\gamma_2 x]) \in \mathrm{S}(G, G)$, which is obvious because $(\Pi_i(N_1), \gamma_1 x, \Pi_i(N_2), \gamma_2 x) \in \overline{G}(\tau)$.

■

Lemma 12.

$$
\frac{\Gamma \vdash e_1 \sim e_2 : \tau \quad \Gamma \vdash e_1' \sim e_2' : \tau'}{\Gamma \vdash \langle e_1, e_1' \rangle \sim \langle e_2, e_2' \rangle : \tau \times \tau'}
$$

Proof:

- We know (1) $\Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau$ and (2) $\Gamma \vdash e'_1 \sim_{L',f'_1,f'_2} e'_2 : \tau'$ with $\mathrm{consistent}(L)$ and $\mathrm{consistent}(L')$.
- We define L'' as follows:

$$\begin{aligned} L''(G) := {} & L(G) \cup L'(G) \\ & \cup \mathrm{close}\{(\tau, f'_1(N_1) + 2 + k_1, K_1[\langle \bullet, \gamma_1 e'_1\rangle], f'_2(N_2) + 2 + k_2, K_2[\langle \bullet, \gamma_2 e'_2\rangle]) \mid \\ & \qquad (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma) \wedge (k_1, K_1, k_2, K_2) \in G(\tau \times \tau')\} \\ & \cup \mathrm{close}\{(\tau', 1 + k_1, K_1[\langle v_1, \bullet\rangle], 1 + k_2, K_2[\langle v_2, \bullet\rangle]) \mid \\ & \qquad (\_, v_1, \_, v_2) \in \overline{G}(\tau) \wedge (k_1, K_1, k_2, K_2) \in G(\tau \times \tau')\} \end{aligned}$$

- We first show $\Gamma \vdash \langle e_1, e'_1\rangle \sim_{L'',(f_1+f'_1+2),(f_2+f'_2+2)} \langle e_2, e'_2\rangle : \tau \times \tau'$.
  - Assume $G \in \mathrm{GK}(L'')$, $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$, and $(k_1, K_1, k_2, K_2) \in G(\tau \times \tau')$.
  - Using (1) and Lemma 10, it suffices to show

    $$(f'_1(N_1) + 2 + k_1, K_1[\langle \bullet, \gamma_1 e'_1\rangle], f'_2(N_2) + 2 + k_2, K_2[\langle \bullet, \gamma_2 e'_2\rangle]) \in G(\tau),$$

    which holds due to $G \supseteq L''(G)$.
- It remains to prove $\mathrm{consistent}(L'')$. With Lemma 10, this reduces to the following:
  1) – Suppose $G \in \mathrm{GK}(L'')$, $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$, $(k_1, K_1, k_2, K_2) \in G(\tau \times \tau')$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau)$.
     – We must show $(f'_1(N_1) + 1 + k_1 + m_1 + n_1, K_1[\langle v_1, \gamma_1 e'_1\rangle], f'_2(N_2) + 1 + k_2 + m_2 + n_2, K_2[\langle v_2, \gamma_2 e'_2\rangle]) \in \mathrm{E}(G)$.
     – Using (2) and Lemma 10, it suffices to show

     $$(1 + k_1 + m_1 + n_1, K_1[\langle v_1, \bullet\rangle], 1 + k_2 + m_2 + n_2, K_2[\langle v_2, \bullet\rangle]) \in G(\tau'),$$

     which holds due to $G \supseteq L''(G)$.
  2) – Suppose $G \in \mathrm{GK}(L'')$, $(\_, v_1, \_, v_2) \in \overline{G}(\tau)$, $(k_1, K_1, k_2, K_2) \in G(\tau \times \tau')$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n'_1, v'_1, n'_2, v'_2) \in \overline{G}(\tau')$.
     – We must show $(k_1 + m_1 + n'_1, K_1[\langle v_1, v'_1\rangle], k_2 + m_2 + n'_2, K_2[\langle v_2, v'_2\rangle]) \in \mathrm{E}(G)$.
     – This follows from $\mathrm{E}(G) \supseteq \mathrm{S}(G, G)$, $(k_1, K_1, k_2, K_2) \in G(\tau \times \tau')$, and $(m_1 + n'_1, \langle v_1, v'_1\rangle, m_2 + n'_2, \langle v_2, v'_2\rangle) \in \overline{G}(\tau \times \tau')$.

■

Lemma 13.

$$\frac{\Gamma \vdash e_1 \sim e_2 : \tau \times \tau'}{\Gamma \vdash e_1.1 \sim e_2.1 : \tau}$$

Proof:

- We know (*) $\Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau \times \tau'$ with $\mathrm{consistent}(L)$.
- We define L' as follows:

$$\begin{aligned} L'(G) := {} & L(G) \\ & \cup \mathrm{close}\{(\tau \times \tau', k_1 + 1, K_1[\bullet.1], k_2 + 1, K_2[\bullet.1]) \mid \\ & \qquad (k_1, K_1, k_2, K_2) \in G(\tau)\} \end{aligned}$$

- We first show $\Gamma \vdash e_1.1 \sim_{L',f_1+1,f_2+1} e_2.1 : \tau$.
  - Assume $G \in \mathrm{GK}(L)$ and $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$ and $(k_1, K_1, k_2, K_2) \in G(\tau)$.
  - Using (1) and Lemma 10, it suffices to show

    $$(k_1 + 1, K_1[\bullet.1], k_2 + 1, K_2[\bullet.1]) \in G(\tau \times \tau'),$$

    which holds due to $G \supseteq L'(G)$.
- It remains to show $\mathrm{consistent}(L')$. With Lemma 10, this reduces to the following.
  - Suppose $G \in \mathrm{GK}(L')$, $m_1 \geq 0$, $m_2 \geq 0$, $(k_1, K_1, k_2, K_2) \in G(\tau)$, and (**) $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau \times \tau')$.
  - We must show $(k_1 + m_1 + n_1, K_1[v_1.1], k_2 + m_2 + n_2, K_2[v_2.1]) \in \mathrm{E}(G)$.
  - From (**) we know there are v'_1, v''_1, v'_2, v''_2 such that $v_i = \langle v'_i, v''_i\rangle$ and $(\_, v'_1, \_, v'_2) \in \overline{G}(\tau)$.
  - It thus suffices to show $(\_, K_1[v'_1], \_, K_2[v'_2]) \in \mathrm{S}(G, G)$, which is obvious.

■

Lemma 14.

$$\frac{\Gamma, f{:}(\tau' \to \tau), x{:}\tau' \vdash e_1 \sim e_2 : \tau}{\Gamma \vdash \mathsf{fix}\, f(x).\, e_1 \sim \mathsf{fix}\, f(x).\, e_2 : \tau' \to \tau}$$

Proof:

- We know (*) $\Gamma, f{:}(\tau' \to \tau), x{:}\tau' \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau$ with $\mathrm{consistent}(L)$.
- We define L' as follows:

$$\begin{aligned} L'(G) &:= L(G) \\ &\quad \cup \mathrm{close}\{(\tau' \to \tau, 0, \mathsf{fix}\, f(x).\, \gamma_1 e_1, 0, \mathsf{fix}\, f(x).\, \gamma_2 e_2) \mid \\ &\qquad (\_, \gamma_1, \_, \gamma_2) \in \overline{G}(\Gamma)\} \end{aligned}$$

- We first show $\Gamma \vdash \mathsf{fix}\, f(x).\, e_1 \sim_{L',0,0} \mathsf{fix}\, f(x).\, e_2 : \tau' \to \tau$.
  - Assume $G \in \mathrm{GK}(L')$, $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$, and $(k_1, K_1, k_2, K_2) \in G(\tau' \to \tau)$.
  - It suffices to show $(k_1, K_1[\mathsf{fix}\, f(x).\, \gamma_1 e_1], k_2, K_2[\mathsf{fix}\, f(x).\, \gamma_2 e_2]) \in \mathrm{S}(G, G)$, which follows with $G \supseteq L'(G)$.
- It remains to show $\mathrm{consistent}(L')$. With Lemma 10, this reduces to the following.
  - Suppose $G \in \mathrm{GK}(L')$, $(\_, \gamma_1, \_, \gamma_2) \in \overline{G}(\Gamma)$, $m_1 \geq 0$, $m_2 \geq 0$, $(\_, K_1, \_, K_2) \in G(\tau)$, and $(\_, v_1, \_, v_2) \in \overline{G}(\tau')$.
  - We must show $(m_1, K_1[(\mathsf{fix}\, f(x).\, \gamma_1 e_1)\ v_1], m_2, K_2[(\mathsf{fix}\, f(x).\, \gamma_2 e_2)\ v_2]) \in \mathrm{E}(G)$.
  - By Lemma 9 it suffices to show $(\_, K_1[\gamma_1' e_1], \_, K_2[\gamma_2' e_2]) \in \mathrm{E}(G)$, where $\gamma_i' := \gamma_i, f {\mapsto} (\mathsf{fix}\, f(x).\, \gamma_i e_i), x {\mapsto} v_i$.
  - Using Lemma 10, this follows from (*) if we can show

    $$(\_, \mathsf{fix}\, f(x).\, \gamma_1 e_1, \_, \mathsf{fix}\, f(x).\, \gamma_2 e_2) \in \overline{G}(\tau' \to \tau)$$

    and

    $$(\_, v_1, \_, v_2) \in \overline{G}(\tau').$$

  - The former follows again from $G \supseteq L'(G)$ and the latter is given.

■

Lemma 15.

$$\frac{\Gamma \vdash e_1 \sim e_2 : \tau' \to \tau \qquad \Gamma \vdash e_1' \sim e_2' : \tau'}{\Gamma \vdash e_1\ e_1' \sim e_2\ e_2' : \tau}$$

Proof:

- We know (1) $\Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau' \to \tau$ and (2) $\Gamma \vdash e_1' \sim_{L',f_1',f_2'} e_2' : \tau'$ with $\mathrm{consistent}(L)$ and $\mathrm{consistent}(L')$.
- We define L'' as follows:

$$\begin{aligned} L''(G) &:= L(G) \cup L'(G) \\ &\quad \cup \mathrm{close}\{(\tau' \to \tau, f_1'(N_1) + 2, K_1[\bullet\ \gamma_1 e_1'], f_2'(N_2) + 2, K_2[\bullet\ \gamma_2 e_2']) \mid \\ &\qquad (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma) \wedge (\_, K_1, \_, K_2) \in G(\tau)\} \\ &\quad \cup \mathrm{close}\{(\tau', 1 + n_1, K_1[v_1\ \bullet], 1 + n_2, K_2[v_2\ \bullet]) \mid \\ &\qquad (\_, K_1, \_, K_2) \in G(\tau) \wedge (n_1, v_1, n_2, v_2) \in \overline{G}(\tau' \to \tau)\} \end{aligned}$$

- We first show $\Gamma \vdash e_1\ e_1' \sim_{L'',(f_1+f_1'+2),(f_2+f_2'+2)} e_2\ e_2' : \tau$.
  - Assume $G \in \mathrm{GK}(L'')$, $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$, and $(k_1, K_1, k_2, K_2) \in G(\tau)$.
  - Using (1) and Lemma 10, it suffices to show $(f_1'(N_1) + 2 + k_1, K_1[\bullet\ \gamma_1 e_1'], f_2'(N_2) + 2 + k_2, K_2[\bullet\ \gamma_2 e_2']) \in G(\tau' \to \tau)$, which holds due to $G \supseteq L''(G)$.
- It remains to show $\mathrm{consistent}(L'')$. With Lemma 10, this reduces to the following.
  1) - Suppose $G \in \mathrm{GK}(L'')$, $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$, $(\_, K_1, \_, K_2) \in G(\tau)$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau' \to \tau)$.
     - We must show $(f_1'(N_1) + 1 + m_1 + n_1, K_1[v_1\ \gamma_1 e_1'], f_2'(N_2) + 1 + m_2 + n_2, K_2[v_2\ \gamma_2 e_2']) \in \mathrm{E}(G)$.
     - Using (2) and Lemma 10, it then suffices to show

       $$(1 + m_1 + n_1, K_1[v_1\ \bullet], 1 + m_2 + n_2, K_2[v_2\ \bullet]) \in G(\tau'),$$

       which holds due to $G \supseteq L''(G)$.
  2) - Suppose $G \in \mathrm{GK}(L'')$, $(\_, K_1, \_, K_2) \in G(\tau)$, $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau' \to \tau)$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n_1', v_1', n_2', v_2') \in \overline{G}(\tau')$.
     - We must show $(n_1 + m_1 + n_1', K_1[v_1\ v_1'], n_2 + m_2 + n_2', K_2[v_2\ v_2']) \in \mathrm{E}(G)$.
     - It suffices to show $(n_1, K_1[v_1\ v_1'], n_2, K_2[v_2\ v_2']) \in \mathrm{S}(G, G)$, which is obvious.

■

Lemma 16.

$$\frac{\Gamma, x{:}\mathsf{cont}\,\tau \vdash e_1 \sim e_2 : \tau}{\Gamma \vdash \mathsf{callcc}\,(x.\ e_1) \sim \mathsf{callcc}\,(x.\ e_2) : \tau}$$

Proof:

- We know (*) $\Gamma, x{:}\mathsf{cont}\,\tau \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau$ with consistent(L).
- We show $\Gamma \vdash \mathsf{callcc}\,(x.\ e_1) \sim_{L,0,0} \mathsf{callcc}\,(x.\ e_2) : \tau$.
- So assume $G \in \mathrm{GK}(L)$ and $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$ and $(k_1, K_1, k_2, K_2) \in G(\tau)$.
- We must show $(k_1, K_1[\mathsf{callcc}\,(x.\ \gamma_1 e_1)], k_2, K_2[\mathsf{callcc}\,(x.\ \gamma_2 e_2)]) \in \mathrm{E}(G)$.
- By Lemma 9 it suffices to show $(\_, K_1[\gamma_1' e_1], \_, K_2[\gamma_2' e_2]) \in \mathrm{E}(G)$, where $\gamma_i' := \gamma_i, x \mapsto (\mathsf{cont}\ K_i)$.
- This follows from (*) if we can show $(\_, \mathsf{cont}\ K_1, \_, \mathsf{cont}\ K_2) \in \overline{G}(\mathsf{cont}\,\tau)$ and $(\_, K_1, \_, K_2) \in G(\tau)$.
- The former follows from the latter and the latter is given.

■

Lemma 17.

$$\frac{\Gamma \vdash e_1' \sim e_2' : \tau' \quad \Gamma \vdash e_1 \sim e_2 : \mathsf{cont}\,\tau'}{\Gamma \vdash \mathsf{throw}\ e_1'\ \mathsf{to}\ e_1 \sim \mathsf{throw}\ e_2'\ \mathsf{to}\ e_2 : \tau}$$

Proof:

- We know (1) $\Gamma \vdash e_1' \sim_{L',f_1',f_2'} e_2' : \tau'$ and (2) $\Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \mathsf{cont}\,\tau'$ with consistent(L) and consistent(L').
- We define L'' as follows:

$$\begin{aligned} L''(G) :=\ & L(G) \cup L'(G) \\ & \cup \mathrm{close}\{(\tau', f_1(N_1)+1, K_1[\mathsf{throw}\ \bullet\ \mathsf{to}\ \gamma_1 e_1], f_2(N_2)+1, K_2[\mathsf{throw}\ \bullet\ \mathsf{to}\ \gamma_2 e_2]) \mid \\ & \qquad (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma) \wedge (\_, K_1, \_, K_2) \in G(\tau)\} \\ & \cup \mathrm{close}\{(\mathsf{cont}\,\tau', 0, K_1[\mathsf{throw}\ v_1'\ \mathsf{to}\ \bullet], 0, K_2[\mathsf{throw}\ v_2'\ \mathsf{to}\ \bullet]) \mid \\ & \qquad (\_, K_1, \_, K_2) \in G(\tau) \wedge (\_, v_1', \_, v_2') \in \overline{G}(\tau')\} \end{aligned}$$

- We first show $\Gamma \vdash \mathsf{throw}\ e_1'\ \mathsf{to}\ e_1 \sim_{L'',(f_1'+f_1+1),(f_2'+f_2+1)} \mathsf{throw}\ e_2'\ \mathsf{to}\ e_2 : \tau$.
  - Assume $G \in \mathrm{GK}(L'')$ and $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$ and $(k_1, K_1, k_2, K_2) \in G(\tau)$.
  - Using (1) and Lemma 10, it suffices to show $(f_1(N_1)+1+k_1, K_1[\mathsf{throw}\ \bullet\ \mathsf{to}\ \gamma_1 e_1], f_2(N_2)+1+k_2, K_2[\mathsf{throw}\ \bullet\ \mathsf{to}\ \gamma_2 e_2]) \in G(\tau')$, which holds due to $G \supseteq L''(G)$.
- It remains to show consistent(L''). With Lemma 10, this reduces to the following:
  1) – Suppose $G \in \mathrm{GK}(L'')$, $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$, $(\_, K_1, \_, K_2) \in G(\tau)$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n_1', v_1', n_2', v_2') \in \overline{G}(\tau')$.
     – We must show $(f_1(N_1)+m_1+n_1', K_1[\mathsf{throw}\ v_1'\ \mathsf{to}\ \gamma_1 e_1], f_2(N_2)+m_2+n_2', K_2[\mathsf{throw}\ v_2'\ \mathsf{to}\ \gamma_2 e_2]) \in \mathrm{E}(G)$.
     – Using (2) and Lemma 10, it then suffices to show

$$(m_1+n_1', K_1[\mathsf{throw}\ v_1'\ \mathsf{to}\ \bullet], m_2+n_2', K_2[\mathsf{throw}\ v_2'\ \mathsf{to}\ \bullet]) \in G(\mathsf{cont}\,\tau'),$$

     which holds due to $G \supseteq L''(G)$.
  2) – Suppose $G \in \mathrm{GK}(L'')$, $(\_, K_1, \_, K_2) \in G(\tau)$, $(\_, v_1', \_, v_2') \in \overline{G}(\tau')$, $(n_1, v_1, n_2, v_2) \in \overline{G}(\mathsf{cont}\,\tau')$, $m_1+n_1 > 0$, and $m_2+n_2 > 0$.
     – We must show $(m_1+n_1-1, K_1[\mathsf{throw}\ v_1'\ \mathsf{to}\ v_1], m_2+n_2-1, K_2[\mathsf{throw}\ v_2'\ \mathsf{to}\ v_2]) \in \mathrm{E}(G)$.
     – We know $v_1 = \mathsf{cont}\ K_1'$ and $v_2 = \mathsf{cont}\ K_2'$ for some $(\_, K_1', \_, K_2') \in G(\tau')$.
     – It thus suffices to show $(\_, K_1'[v_1'], \_, K_2'[v_2']) \in \mathrm{S}(G, G)$, which is obvious.

■

Lemma 18.

$$\frac{\Gamma \vdash e_1 \sim e_2 : \tau \to \mathsf{int}}{\Gamma \vdash \mathsf{isolate}\ e_1 \sim \mathsf{isolate}\ e_2 : \mathsf{cont}\,\tau}$$

Proof:

- We know (*) $\Gamma \vdash e_1 \sim_{L,f_1,f_2} e_2 : \tau \to \mathsf{int}$ with consistent(L).
- We define L' as follows:

$$\begin{aligned} L'(G) :=\ & L(G) \\ & \cup \mathrm{close}\{(\tau \to \mathsf{int}, 0, K_1[\mathsf{isolate}\ \bullet], 0, K_2[\mathsf{isolate}\ \bullet]) \mid \\ & \qquad (\_, K_1, \_, K_2) \in G(\mathsf{cont}\,\tau)\} \\ & \cup \mathrm{close}\{(\tau, n_1+1, v_1\ \bullet, n_2+1, v_2\ \bullet) \mid \\ & \qquad (n_1, v_1, n_2, v_2) \in G(\tau \to \mathsf{int})\} \end{aligned}$$

- We first show $\Gamma \vdash \mathsf{isolate}\ e_1 \sim_{L',(f_1+1),(f_2+1)} \mathsf{isolate}\ e_2 : \mathsf{cont}\,\tau$.
  - Assume $G \in \mathrm{GK}(L')$ and $(N_1, \gamma_1, N_2, \gamma_2) \in \overline{G}(\Gamma)$ and $(k_1, K_1, k_2, K_2) \in G(\mathsf{cont}\,\tau)$.
  - We must show $(f_1(N_1)+k_1, K_1[\mathsf{isolate}\ \gamma_1 e_1], f_2(N_2)+k_2, K_2[\mathsf{isolate}\ \gamma_2 e_2]) \in \mathrm{E}(G)$.

– Using (*) and Lemma 10, it suffices to show $(k_1, K_1[\mathsf{isolate}\ \bullet], k_2, K_2[\mathsf{isolate}\ \bullet]) \in G(\tau \to \mathsf{int})$, which holds due to $G \supseteq L'(G)$.

- It remains to show $\mathrm{consistent}(L')$. With Lemma 10, this reduces to the following:

1) – Suppose $G \in \mathrm{GK}(L')$, $(\_, K_1, \_, K_2) \in G(\mathsf{cont}\,\tau)$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau \to \mathsf{int})$.
   – We must show $(m_1 + n_1, K_1[\mathsf{isolate}\ v_1], m_2 + n_2, K_2[\mathsf{isolate}\ v_2]) \in \mathrm{E}(G)$.
   – It suffices to show $(\_, K_1[\mathsf{cont}\ (v_1\ \bullet)], \_, K_2[\mathsf{cont}\ (v_2\ \bullet)]) \in \mathrm{S}(G, G)$.
   – This follows from $(\_, v_1\ \bullet, \_, v_2\ \bullet) \in G(\tau)$, which holds due to $G \supseteq L'(G)$.
2) – Suppose $G \in \mathrm{GK}(L')$, $(n_1, v_1, n_2, v_2) \in G(\tau \to \mathsf{int})$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n'_1, v'_1, n'_2, v'_2) \in \overline{G}(\tau)$.
   – We must show $(n_1 + m_1 + n'_1, v_1\ v'_1, n_2 + m_2 + n'_2, v_2\ v'_2) \in \mathrm{E}(G)$.
   – It suffices to show $(n_1, v_1\ v'_1, n_2, v_2\ v'_2) \in \mathrm{S}(G, G)$, which follows from $(\_, \bullet, \_, \bullet) \in G(\mathsf{int})$.

■

A.3.3 Example: Callcc in a Loop.

$$
\begin{aligned}
&\tau := \mu\alpha.\, \mathsf{cont}\,\mathsf{int} \to \alpha \\
&e_1 := |\lambda y{:}\tau.\, \mathsf{callcc}_{\mathsf{int}}\,(k.\ F_k\ y)| \qquad \text{where } F_v := \mathsf{fix}\,f(x).\, f\ (\mathsf{unroll}\ x\ v) \\
&e_2 := |\mathsf{fix}\,f(x{:}\tau){:}\mathsf{int}.\, \mathsf{callcc}_{\mathsf{int}}\,(k.\ f\ (\mathsf{unroll}\ x\ k))|
\end{aligned}
$$

In order to prove $\vdash e_1 \sim e_2 : \tau \to \mathsf{int}$ (in $\lambda^{\mu}_{\mathrm{cc}}$), we define a suitable local knowledge L as follows:

$$
\begin{aligned}
L(G) := \quad & \mathrm{close}\{(\tau \to \mathsf{int}, 0, e_1, 0, e_2)\} \\
\cup \quad & \mathrm{close}\{(\tau, 0, K_1[F_{(\mathsf{cont}\ K_1)}\ \bullet], 0, K_2[e_2\ \bullet]) \mid (\_, K_1, \_, K_2) \in G(\mathsf{int})\}
\end{aligned}
$$

We first show $\vdash e_1 \sim_{L,0,0} e_2 : \tau \to \mathsf{int}$:

- Suppose $G \in \mathrm{GK}(L)$ and $(k_1, K_1, k_2, K_2) \in G(\tau \to \mathsf{int})$.
- We must show $(k_1, K_1[e_1], k_2, K_2[e_2]) \in \mathrm{E}(G)$.
- It suffices to show $(0, e_1, 0, e_2) \in G(\tau \to \mathsf{int})$, which holds due to $G \supseteq L(G)$.

It remains to show $\mathrm{consistent}(L)$:

1) - Suppose $G \in \mathrm{GK}(L)$, $m_1 \geq 0$, $m_2 \geq 0$, (*) $(\_, v_1, \_, v_2) \in \overline{G}(\tau)$, and $(\_, K_1, \_, K_2) \in G(\mathsf{int})$.
   - We must show $(m_1, K_1[e_1\ v_1], m_2, K_2[e_2\ v_2]) \in \mathrm{E}(G)^{\$}$.
   - From (*) we know that there is $(\_, v'_1, \_, v'_2) \in \overline{G}(\mathsf{cont}\,\mathsf{int} \to \tau)$ such that $v_i = \mathsf{roll}\ v'_i$.
   - It thus suffices to show $(\_, K_1[F_{(\mathsf{cont}\ K_1)}\ (v'_1\ (\mathsf{cont}\ K_1))], \_, K_2[e_2\ (v'_2\ (\mathsf{cont}\ K_2))]) \in \mathrm{S}(G, G)$.
   - Since $(\_, \mathsf{cont}\ K_1, \_, \mathsf{cont}\ K_2) \in \overline{G}(\mathsf{cont}\,\mathsf{int})$, this follows from $(\_, K_1[F_{(\mathsf{cont}\ K_1)}\ \bullet], \_, K_2[e_2\ \bullet]) \in G(\tau)$, which holds due to $G \supseteq L(G)$.
2) - Suppose $G \in \mathrm{GK}(L)$, $(\_, K_1, \_, K_2) \in G(\mathsf{int})$, $m_1 \geq 0$, $m_2 \geq 0$, and (*) $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau)$.
   - We must show $(m_1 + n_1, K_1[F_{(\mathsf{cont}\ K_1)}\ v_1], m_2 + n_2, K_2[e_2\ v_2]) \in \mathrm{E}(G)^{\$}$.
   - From (*) we know that there is $(\_, v'_1, \_, v'_2) \in \overline{G}(\mathsf{cont}\,\mathsf{int} \to \tau)$ such that $v_i = \mathsf{roll}\ v'_i$.
   - It thus suffices to show $(\_, K_1[F_{(\mathsf{cont}\ K_1)}\ (v'_1\ (\mathsf{cont}\ K_1))], \_, K_2[e_2\ (v'_2\ (\mathsf{cont}\ K_2))]) \in \mathrm{S}(G, G)$.
   - Since $(\_, \mathsf{cont}\ K_1, \_, \mathsf{cont}\ K_2) \in \overline{G}(\mathsf{cont}\,\mathsf{int})$, this follows from $(\_, K_1[F_{(\mathsf{cont}\ K_1)}\ \bullet], \_, K_2[e_2\ \bullet]) \in G(\tau)$, which holds due to $G \supseteq L(G)$.

A.3.4 Example: Eta Equivalence (lambda version).

$$
\begin{aligned}
&e_1 := |\lambda f{:}\tau \to \tau'.\, \lambda x{:}\tau.\, f\ x| \\
&e_2 := |\lambda f{:}\tau \to \tau'.\, f|
\end{aligned}
$$

In order to prove $\vdash e_1 \sim e_2 : (\tau \to \tau') \to \tau \to \tau'$ for $\lambda^{\mu}_{\mathrm{cc}}$ (and thus λ^{μ}), we define a suitable local knowledge L as follows:

$$
\begin{aligned}
L(G) := \quad & \mathrm{close}\{((\tau \to \tau') \to \tau \to \tau', 0, e_1, 0, e_2)\} \\
\cup \quad & \mathrm{close}\{(\tau \to \tau', 0, \lambda x.\, v_1\ x, n_2 + 1, v_2) \mid (\_, v_1, n_2, v_2) \in G(\tau \to \tau')\}
\end{aligned}
$$

We first show $\vdash e_1 \sim_{L,0,0} e_2 : (\tau \to \tau') \to \tau \to \tau'$:

- Suppose $G \in \mathrm{GK}(L)$ and $(k_1, K_1, k_2, K_2) \in G((\tau \to \tau') \to \tau \to \tau')$.
- We must show $(k_1, K_1[e_1], k_2, K_2[e_2]) \in \mathrm{E}(G)$.
- It suffices to show $(0, e_1, 0, e_2) \in G((\tau \to \tau') \to \tau \to \tau')$, which holds due to $G \supseteq L(G)$.

It remains to show $\mathrm{consistent}(L)$:

1) - Suppose $G \in \mathrm{GK}(L)$, $m_1 \geq 0$, $m_2 \geq 0$, $(\_, v_1, \_, v_2) \in \overline{G}(\tau \to \tau')$, and $(\_, K_1, \_, K_2) \in G(\tau \to \tau')$.
   - We must show $(m_1, K_1[e_1\ v_1], m_2, K_2[e_2\ v_2]) \in \mathrm{E}(G)^{\$}$.

- It suffices to show $(\_, K_1[\lambda x.\, v_1\ x], \_, K_2[v_2]) \in \mathrm{S}(G, G)$.
- This follows from $(\_, \lambda x.\, v_1\ x, \_, v_2) \in \overline{G}(\tau \to \tau')$, which holds due to $G \supseteq L(G)$.

2)
- Suppose $G \in \mathrm{GK}(L)$, $(\_, v_1, n_2, v_2) \in G(\tau \to \tau')$, $m_1 \geq 0$, $m_2 \geq 0$, $(\_, v_1', \_, v_2') \in \overline{G}(\tau)$ and $(\_, K_1, \_, K_2) \in G(\tau')$.
- We must show $(m_1, K_1[(\lambda x.\, v_1\ x)\ v_1'], n_2 + 1 + m_2, K_2[v_2\ v_2']) \in \mathrm{E}(G)^{\$}$.
- It suffices to show $(\_, K_1[v_1\ v_1'], n_2, K_2[v_2\ v_2']) \in \mathrm{S}(G, G)$, which follows with $(\_, v_1, n_2, v_2) \in G(\tau \to \tau')$.

A.3.5 Example: Syntactic Minimal Invariance.

$$
\begin{aligned}
&\tau := \mu\alpha.\, \mathsf{int} + (\alpha \to \alpha)\\
&e_1 := |\mathsf{fix}\, f(x{:}\tau){:}\tau.\, \mathsf{case}\, \mathsf{unroll}\ x\, \mathsf{of}\ \mathsf{inl}\, y \Rightarrow \mathsf{roll}\ (\mathsf{inl}\ y)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mid \mathsf{inr}\, g \Rightarrow \mathsf{roll}\ (\mathsf{inr}\ \lambda y{:}\tau.\, f\ (g\ (f\ y)))|\\
&e_2 := |\lambda x{:}\tau.\, x|
\end{aligned}
$$

In order to prove $\vdash e_1 \sim e_2 : \tau \to \tau$ for $\lambda^{\mu}_{\mathrm{cc}}$ (and thus λ^{μ}), we define a suitable local knowledge L as follows:

$$
\begin{aligned}
L(G) :=\ & \mathrm{close}\{(\tau \to \tau, 0, e_1, 0, e_2)\}\\
\cup\ & \mathrm{close}\{(\tau \to \tau, 0, \lambda y.\, e_1\ (v_1\ (e_1\ y)), n_2 + 1, v_2) \mid (\_, v_1, n_2, v_2) \in G(\tau \to \tau)\}\\
\cup\ & \mathrm{close}\{(\tau, 0, K_1[e_1\ \bullet], k_2 + 1, K_2) \mid (\_, K_1, k_2, K_2) \in G(\tau)\}
\end{aligned}
$$

We first show $\vdash e_1 \sim_{L,0,0} e_2 : \tau \to \tau$:

- Suppose $G \in \mathrm{GK}(L)$ and $(k_1, K_1, k_2, K_2) \in G(\tau \to \tau)$.
- We must show $(k_1, K_1[e_1], k_2, K_2[e_2]) \in \mathrm{E}(G)$.
- It suffices to show $(0, e_1, 0, e_2) \in G(\tau \to \tau)$, which holds due to $G \supseteq L(G)$.

To simplify the proof of consistency, we now show the following property (†):

$$\forall G \in \mathrm{GK}(L), (\_, v_1, n_2, v_2) \in \overline{G}(\tau).\ \forall K.\ \exists v_1'.\ K[e_1\ v_1] \hookrightarrow^{+} K[v_1'] \wedge (\_, v_1', n_2, v_2) \in \overline{G}(\tau)$$

- From $(\_, v_1, n_2, v_2) \in \overline{G}(\tau)$ we know that there are v_1', v_2' such that $v_i = \mathsf{roll}\ v_i'$ and (*) $(\_, v_1', \_, v_2') \in \overline{G}(\mathsf{int} + (\tau \to \tau))$.
- From (*) we know that there are v_1'', v_2'' such that either (a) $v_i' = \mathsf{inl}\ v_i''$ and $(\_, v_1'', \_, v_2'') \in \overline{G}(\mathsf{int})$ or (b) $v_i' = \mathsf{inr}\ v_i''$ and $(\_, v_1'', \_, v_2'') \in \overline{G}(\tau \to \tau)$.
- In case (a) we know $K[e_1\ v_1] \hookrightarrow^{+} K[v_1]$, and we are done.
- Now consider case (b).
- Here we know $K[e_1\ v_1] \hookrightarrow^{+} K[\widehat{v_1}]$ for $\widehat{v_1} := \mathsf{roll}\ (\mathsf{inr}\ \lambda y.\, e_1\ (v_1''\ (e_1\ y)))$, so it remains to show $(\_, \widehat{v_1}, n_2, \mathsf{roll}\ (\mathsf{inr}\ v_2'')) \in \overline{G}(\tau)$.
- This follows from $(\_, \lambda y.\, e_1\ (v_1''\ (e_1\ y)), \_, v_2'') \in \overline{G}(\tau \to \tau)$, which holds due to $G \supseteq L(G)$.

It remains to show $\mathrm{consistent}(L)$:

1)
- Suppose $G \in \mathrm{GK}(L)$, $m_1 \geq 0$, $m_2 \geq 0$, $(\_, v_1, \_, v_2) \in \overline{G}(\tau)$, and $(\_, K_1, \_, K_2) \in G(\tau)$.
- We must show $(m_1, K_1[e_1\ v_1], m_2, K_2[e_2\ v_2]) \in \mathrm{E}(G)^{\$}$.
- With the help of (†), it suffices to show $(\_, K_1[v_1'], \_, K_2[v_2]) \in \mathrm{S}(G, G)$ for any v_1' with $(\_, v_1', \_, v_2) \in \overline{G}(\tau)$, which is obvious.

2)
- Suppose $G \in \mathrm{GK}(L)$, $(\_, v_1, n_2, v_2) \in G(\tau \to \tau)$, $m_1 \geq 0$, $m_2 \geq 0$, $(\_, v_1', \_, v_2') \in \overline{G}(\tau)$, and $(\_, K_1, \_, K_2) \in G(\tau)$.
- We must show $(m_1, K_1[(\lambda y.\, e_1\ (v_1\ (e_1\ y)))\ v_1'], n_2 + 1 + m_2, K_2[v_2\ v_2']) \in \mathrm{E}(G)^{\$}$.
- With the help of (†), it suffices to show $(\_, K_1[e_1\ (v_1\ v_1'')], n_2, K_2[v_2\ v_2']) \in \mathrm{S}(G, G)$ for any v_1'' with $(\_, v_1'', \_, v_2') \in \overline{G}(\tau)$.
- This follows from $(\_, K_1[e_1\ \bullet], \_, K_2) \in G(\tau)$, which holds due to $G \supseteq L(G)$.

3)
- Suppose $G \in \mathrm{GK}(L)$, $(\_, K_1, k_2, K_2) \in G(\tau)$, $m_1 \geq 0$, $m_2 \geq 0$, and $(n_1, v_1, n_2, v_2) \in \overline{G}(\tau)$.
- We must show $(m_1 + n_1, K_1[e_1\ v_1], k_2 + 1 + m_2 + n_2, K_2[v_2]) \in \mathrm{E}(G)^{\$}$.
- With the help of (†), it suffices to show $(\_, K_1[v_1'], k_2 + n_2, K_2[v_2]) \in \mathrm{S}(G, G)$ for any v_1' with $(\_, v_1', n_2, v_2) \in \overline{G}(\tau)$, which is obvious.

A.3.6 Using Paco (parameterized coinduction).

Definition 8.

$$
\begin{aligned}
&\mathfrak{f} \in \mathrm{LK} \overset{\mathrm{mon}}{\to} \mathrm{LK}\\
&\mathfrak{f}(L)(G)(\tau) := \{(n_1, d_1, n_2, d_2) \mid\\
&\qquad \forall G' \in \mathrm{GK}(L).\ G' \supseteq G \implies S(\{(\tau, n_1, d_1, n_2, d_2)\}, G') \subseteq \mathrm{E}(G')^{\$}\}
\end{aligned}
$$

Using Lemmas 9 and 10, it is easy to verify that $\mathfrak{f}$ is well-defined, *i.e.,* that it returns valid local knowledges and is itself monotone.

Lemma 19.

$$L \subseteq \mathfrak{f}(L) \implies \text{consistent}(L)$$

Proof:

- Assume $L \subseteq \mathfrak{f}(L)$ and suppose $G \in \text{GK}(L)$ and $(n_1, e_1, n_2, e_2) \in \text{S}(L(G), G)$.
- We must show $(n_1, e_1, n_2, e_2) \in \text{E}(G)^{\$}$.
- From $(n_1, e_1, n_2, e_2) \in \text{S}(L(G), G)$ we know that there is $(\tau, m_1, d_1, m_2, d_2) \in L(G)$ such that $(n_1, e_1, n_2, e_2) \in \text{S}(\{(\tau, m_1, d_1, m_2, d_2)\}, G)$.
- Using the assumption, we have $(\tau, m_1, d_1, m_2, d_2) \in \mathfrak{f}(L)(G)$.
- Exploiting this, we are done because $G \supseteq G$.

■

Lemma 20.

$$\text{consistent}(L) \implies L \subseteq \mathfrak{f}(L)$$

Proof:

- Assume $\text{consistent}(L)$ and suppose $G \in \text{DRel}$ and $(\tau, n_1, d_1, n_2, d_2) \in L(G)$.
- We must show $(\tau, n_1, d_1, n_2, d_2) \in \mathfrak{f}(L)(G)$.
- So suppose $G' \in \text{GK}(L)$, $G' \supseteq G$, and $(m_1, e_1, m_2, e_2) \in \text{S}(\{(\tau, n_1, d_1, n_2, d_2)\}, G')$.
- Using monotonicity of $\text{S}(-, \text{G}')$ and of L, this implies $(m_1, e_1, m_2, e_2) \in \text{S}(L(G'), G')$
- From $\text{consistent}(L)$ we then get $(m_1, e_1, m_2, e_2) \in \text{E}(G')^{\$}$.

■

Definition 9.

$$\begin{array}{l} \mathfrak{L} \in \text{LK} \to \text{LK} \\ \mathfrak{L}(L) := \nu X.\ \mathfrak{f}(X \cup L) \end{array}$$

Lemma 21 ($\mathfrak{L}(\emptyset)$ is the greatest consistent local knowledge)**.**

- $\text{consistent}(\mathfrak{L}(\emptyset))$
- $\text{consistent}(L) \implies L \subseteq \mathfrak{L}(\emptyset)$

Proof: Since $\mathfrak{L}(\emptyset)$ is by definition the greatest postfixed point of $\mathfrak{f}$, this follows from Lemmas 19 and 20. ■

Lemma 22.

$$\Gamma \vdash e_1 \sim e_2 : \tau \iff \exists f_1, f_2.\ \Gamma \vdash e_1 \sim_{\mathfrak{L}(\emptyset), f_1, f_2} e_2 : \tau$$

Proof: The $\Longleftarrow$-direction is trivial with Lemma 21. Consider the $\implies$-direction:

- From $\Gamma \vdash e_1 \sim e_2 : \tau$ we know $\Gamma \vdash e_1 \sim_{L, f_1, f_2} e_2 : \tau$ with $\text{consistent}(L)$.
- The latter implies $L \subseteq \mathfrak{L}(\emptyset)$ by Lemma 21.
- We are done by Lemma 10.

■

Lemma 23 (ACCUMULATE)**.**

$$L \subseteq \mathfrak{L}(L') \iff L \subseteq \mathfrak{L}(L \cup L')$$

Proof: See our Paco paper (The Power of Parameterization in Coinductive Proof). ■

All this can be done for the other models/languages too. See our Coq formalization.

APPENDIX B
SPBS FOR $\mathsf{F}^{\mu!}$ AND $\mathsf{F}^{\mu!}_{\mathsf{cc}}$

B.1 Languages $\mathsf{F}^{\mu!}$ and $\mathsf{F}^{\mu!}_{\mathbf{cc}}$

The fragment $\mathsf{F}^{\mu!}$ is obtained by removing first-class continuations (the parts highlighted in red).

B.1.1 Statics.

$$\begin{array}{rl} \sigma,\tau \in \mathrm{Ty} ::= & \alpha \mid \mathsf{int} \mid \mathsf{unit} \mid \sigma_1 \times \sigma_2 \mid \sigma_1 + \sigma_2 \mid \sigma_1 \to \sigma_2 \mid \mu\alpha.\,\sigma \mid \mathsf{cont}\,\sigma \mid \forall\alpha.\,\sigma \mid \exists\alpha.\,\sigma \mid \mathsf{ref}\,\sigma \\ p \in \mathrm{Prog} ::= & x \mid n \mid \langle\rangle \mid p_1 \odot p_2 \mid \mathsf{ifz}\ p\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2 \mid \langle p_1, p_2\rangle \mid p.1 \mid p.2 \mid \mathsf{inl}_\sigma\, p \mid \mathsf{inr}_\sigma\, p \mid \\ & (\mathsf{case}\, p\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow p_1 \mid \mathsf{inr}\, x \Rightarrow p_2) \mid \mathsf{fix}\, f(x{:}\sigma_1){:}\sigma_2.\, p \mid p_1\ p_2 \mid \mathsf{roll}_\sigma\ p \mid \mathsf{unroll}\ p \mid \\ & \mathsf{callcc}_\sigma\,(x.\ p) \mid \mathsf{throw}_\sigma\ p_1\ \mathsf{to}\ p_2 \mid \mathsf{isolate}\ p \mid \Lambda\alpha.\,p \mid p[\sigma] \mid \mathsf{pack}\ \langle\sigma_1, p\rangle\ \mathsf{as}\ \sigma_2 \mid \\ & \mathsf{unpack}\ p_1\ \mathsf{as}\ \langle\alpha, x\rangle\ \mathsf{in}\ p_2 \mid \mathsf{ref}\ p \mid p_1 := p_2 \mid p_1 == p_2 \mid\ !p \end{array}$$

$$\begin{array}{lcll} \text{Type environments} & \Delta & ::= & \cdot \mid \Delta, \alpha \\ \text{Term environments} & \Gamma & ::= & \cdot \mid \Gamma, x{:}\sigma \end{array}$$

$\boxed{\Delta \vdash \sigma}$

$$\frac{\mathrm{fv}(\sigma) \subseteq \Delta \qquad \mathrm{names}(\sigma) = \emptyset}{\Delta \vdash \sigma}$$

$\boxed{\Delta \vdash \Gamma}$

$$\frac{\forall x{:}\sigma \in \Gamma.\ \Delta \vdash \sigma}{\Delta \vdash \Gamma}$$

$\boxed{\Delta;\Gamma \vdash p : \tau}$

$$\frac{\vdash \Delta;\Gamma \qquad x{:}\tau \in \Delta;\Gamma}{\Delta;\Gamma \vdash x : \tau} \qquad \frac{\vdash \Delta;\Gamma}{\Delta;\Gamma \vdash n : \mathsf{int}}$$

$$\frac{\Delta;\Gamma \vdash p_1 : \mathsf{int} \qquad \Delta;\Gamma \vdash p_2 : \mathsf{int}}{\Delta;\Gamma \vdash p_1 \odot p_2 : \mathsf{int}} \qquad \frac{\Delta;\Gamma \vdash p_0 : \mathsf{int} \qquad \Delta;\Gamma \vdash p_1 : \tau \qquad \Delta;\Gamma \vdash p_2 : \tau}{\Delta;\Gamma \vdash \mathsf{ifz}\ p_0\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2 : \tau}$$

$$\frac{\Delta;\Gamma \vdash p_1 : \tau_1 \qquad \Delta;\Gamma \vdash p_2 : \tau_2}{\Delta;\Gamma \vdash \langle p_1, p_2\rangle : \tau_1 \times \tau_2} \qquad \frac{\Delta;\Gamma \vdash p : \tau_1 \times \tau_2}{\Delta;\Gamma \vdash p.1 : \tau_1} \qquad \frac{\Delta;\Gamma \vdash p : \tau_1 \times \tau_2}{\Delta;\Gamma \vdash p.2 : \tau_2}$$

$$\frac{\Delta;\Gamma \vdash p : \tau_1}{\Delta;\Gamma \vdash \mathsf{inl}_{\tau_2}\, p : \tau_1 + \tau_2} \qquad \frac{\Delta;\Gamma \vdash p : \tau_2}{\Delta;\Gamma \vdash \mathsf{inr}_{\tau_1}\, p : \tau_1 + \tau_2}$$

$$\frac{\Delta;\Gamma \vdash p : \tau_1 + \tau_2 \qquad \Delta;\Gamma, x{:}\tau_1 \vdash p_1 : \tau \qquad \Delta;\Gamma, x{:}\tau_2 \vdash p_2 : \tau}{\Delta;\Gamma \vdash \mathsf{case}\, p\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow p_1 \mid \mathsf{inr}\, x \Rightarrow p_2 : \tau}$$

$$\frac{\Delta;\Gamma, f{:}(\tau_1 \to \tau_2), x{:}\tau_1 \vdash p : \tau_2}{\Delta;\Gamma \vdash \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.\, p : \tau_1 \to \tau_2} \qquad \frac{\Delta;\Gamma \vdash p_1 : \tau_1 \to \tau_2 \qquad \Delta;\Gamma \vdash p_2 : \tau_1}{\Delta;\Gamma \vdash p_1\ p_2 : \tau_2}$$

$$\frac{\Delta;\Gamma \vdash p : \sigma[\mu\alpha.\,\sigma/\alpha]}{\Delta;\Gamma \vdash \mathsf{roll}_{\mu\alpha.\,\sigma}\ p : \mu\alpha.\,\sigma} \qquad \frac{\Delta;\Gamma \vdash p : \mu\alpha.\,\sigma}{\Delta;\Gamma \vdash \mathsf{unroll}\ p : \sigma[\mu\alpha.\,\sigma/\alpha]}$$

$$\frac{\Delta;\Gamma, x{:}\mathsf{cont}\,\tau \vdash p : \tau}{\Delta;\Gamma \vdash \mathsf{callcc}_\tau\,(x.\ p) : \tau} \qquad \frac{\Delta;\Gamma \vdash p' : \tau' \qquad \Delta;\Gamma \vdash p : \mathsf{cont}\,\tau'}{\Delta;\Gamma \vdash \mathsf{throw}_\tau\ p'\ \mathsf{to}\ p : \tau} \qquad \frac{\Delta;\Gamma \vdash p : \tau \to \mathsf{int}}{\Delta;\Gamma \vdash \mathsf{isolate}\ p : \mathsf{cont}\,\tau}$$

$$\frac{\Delta,\alpha;\Gamma \vdash p : \sigma}{\Delta;\Gamma \vdash \Lambda\alpha.\,p : \forall\alpha.\,\sigma} \qquad \frac{\Delta;\Gamma \vdash p : \forall\alpha.\,\sigma \qquad \Delta \vdash \sigma'}{\Delta;\Gamma \vdash p[\sigma'] : \sigma[\sigma'/\alpha]}$$

$$\frac{\Delta \vdash \sigma_1 \quad \Delta; \Gamma \vdash p : \sigma_2[\sigma_1/\alpha]}{\Delta; \Gamma \vdash \mathsf{pack}\ \langle \sigma_1, p\rangle\ \mathsf{as}\ \exists \alpha.\, \sigma_2 : \exists \alpha.\, \sigma_2} \qquad \frac{\Delta; \Gamma \vdash p_1 : \exists \alpha.\, \sigma_1 \quad \Delta, \alpha; \Gamma, x{:}\sigma_1 \vdash p_2 : \sigma_2 \quad \Delta \vdash \sigma_2}{\Delta; \Gamma \vdash \mathsf{unpack}\ p_1\ \mathsf{as}\ \langle \alpha, x\rangle\ \mathsf{in}\ p_2 : \sigma_2}$$

$$\frac{\Delta; \Gamma \vdash p : \sigma}{\Delta; \Gamma \vdash \mathsf{ref}\ p : \mathsf{ref}\ \sigma} \qquad \frac{\Delta; \Gamma \vdash p_1 : \mathsf{ref}\ \sigma \quad \Delta; \Gamma \vdash p_2 : \sigma}{\Delta; \Gamma \vdash p_1 := p_2 : \mathsf{unit}}$$

$$\frac{\Delta; \Gamma \vdash p : \mathsf{ref}\ \sigma}{\Delta; \Gamma \vdash\ !p : \sigma} \qquad \frac{\Delta; \Gamma \vdash p_1 : \mathsf{ref}\ \sigma \quad \Delta; \Gamma \vdash p_2 : \mathsf{ref}\ \sigma}{\Delta; \Gamma \vdash p_1 == p_2 : \mathsf{int}}$$

B.1.2 Dynamics.

$$\begin{array}{lcl}
v \in \mathrm{Val} & ::= & x \mid n \mid \langle\rangle \mid l \mid \langle v_1, v_2\rangle \mid \mathsf{inl}\ v \mid \mathsf{inr}\ v \mid \mathsf{fix}\, f(x).\, e \mid \mathsf{roll}\ v \mid \mathsf{cont}\ K \mid \Lambda.\, e \mid \mathsf{pack}\ e \\
e \in \mathrm{Exp} & ::= & v \mid e_1 \odot e_2 \mid \mathsf{ifz}\ e_0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \langle e_1, e_2\rangle \mid e.1 \mid e.2 \mid \mathsf{inl}\ e \mid \mathsf{inr}\ e \mid \\
& & (\mathsf{case}\, e\, \mathsf{of}\ \mathsf{inl}\, x \Rightarrow e_1 \,|\, \mathsf{inr}\, x \Rightarrow e_2) \mid e_1\ e_2 \mid \mathsf{roll}\ e \mid \mathsf{unroll}\ e \mid \\
& & \mathsf{callcc}\ (x.\ e) \mid \mathsf{throw}\ e_1\ \mathsf{to}\ e_2 \mid \mathsf{isolate}\ e \mid e[] \mid \mathsf{unpack}\ e_1\ \mathsf{as}\ x\ \mathsf{in}\ e_2 \mid \\
& & \mathsf{ref}\ e \mid e_1 := e_2 \mid e_1 == e_2 \mid !e \\
K \in \mathrm{Cont} & ::= & \bullet \mid K \odot e \mid v \odot K \mid \mathsf{ifz}\ K\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \langle K, e\rangle \mid \langle v, K\rangle \mid K.1 \mid K.2 \mid \\
& & \mathsf{inl}\ K \mid \mathsf{inr}\ K \mid (\mathsf{case}\, K\, \mathsf{of}\ \mathsf{inl}\, x \Rightarrow e_1 \,|\, \mathsf{inr}\, x \Rightarrow e_2) \mid K\ e \mid v\ K \mid \mathsf{roll}\ K \mid \\
& & \mathsf{unroll}\ K \mid \mathsf{throw}\ K\ \mathsf{to}\ e \mid \mathsf{throw}\ v\ \mathsf{to}\ K \mid \mathsf{isolate}\ K \mid K[] \mid \mathsf{pack}\ K \mid \\
& & \mathsf{unpack}\ K\ \mathsf{as}\ x\ \mathsf{in}\ e \mid \mathsf{ref}\ K \mid K := e \mid v := K \mid K == e \mid v == K \mid !K
\end{array}$$

$\boxed{h, e \hookrightarrow h', e'}$

$$\begin{array}{lcll}
h, K[n_1 \odot n_2] & \hookrightarrow & h, K[n] & (n = [\![n_1 \odot n_2]\!]) \\
h, K[\mathsf{ifz}\ 0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2] & \hookrightarrow & h, K[e_1] & \\
h, K[\mathsf{ifz}\ n\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2] & \hookrightarrow & h, K[e_2] & (n \neq 0) \\
h, K[\langle v_1, v_2\rangle.1] & \hookrightarrow & h, K[v_1] & \\
h, K[\langle v_1, v_2\rangle.2] & \hookrightarrow & h, K[v_2] & \\
h, K[\mathsf{case}\, \mathsf{inl}\ v\, \mathsf{of}\ \mathsf{inl}\, x \Rightarrow e_1 \,|\, \mathsf{inr}\, x \Rightarrow e_2] & \hookrightarrow & h, K[e_1[v/x]] & \\
h, K[\mathsf{case}\, \mathsf{inr}\ v\, \mathsf{of}\ \mathsf{inl}\, x \Rightarrow e_1 \,|\, \mathsf{inr}\, x \Rightarrow e_2] & \hookrightarrow & h, K[e_2[v/x]] & \\
h, K[(\mathsf{fix}\, f(x).\, e)\ v] & \hookrightarrow & h, K[e[(\mathsf{fix}\, f(x).\, e)/f, v/x]] & \\
h, K[\mathsf{unroll}\ (\mathsf{roll}\ v)] & \hookrightarrow & h, K[v] & \\
h, K[\mathsf{callcc}\ (x.\ e)] & \hookrightarrow & h, K[e[\mathsf{cont}\ K/x]] & \\
h, K[\mathsf{throw}\ v\ \mathsf{to}\ \mathsf{cont}\ K'] & \hookrightarrow & h, K'[v] & \\
h, K[\mathsf{isolate}\ v] & \hookrightarrow & h, K[\mathsf{cont}\ (v\ \bullet)] & \\
h, K[(\Lambda.\, e)[]] & \hookrightarrow & h, K[e] & \\
h, K[\mathsf{unpack}\ (\mathsf{pack}\ v)\ \mathsf{as}\ x\ \mathsf{in}\ e] & \hookrightarrow & h, K[e[v/x]] & \\
h, K[\mathsf{ref}\ v] & \hookrightarrow & h \uplus [\ell \mapsto v], K[\ell] & \\
h \uplus [\ell \mapsto v], K[!\ell] & \hookrightarrow & h \uplus [\ell \mapsto v], K[v] & \\
h \uplus [\ell \mapsto v], K[\ell := v'] & \hookrightarrow & h \uplus [\ell \mapsto v'], K[\langle\rangle] & \\
h, K[\ell == \ell] & \hookrightarrow & h, K[1] & \\
h, K[\ell == \ell'] & \hookrightarrow & h, K[0] & (\ell \neq \ell')
\end{array}$$

B.2 SPB Models for $\mathsf{F}^{\mu!}$ and $\mathsf{F}^{\mu!}_{\mathbf{cc}}$

The model for $\mathsf{F}^{\mu!}$ is obtained by *excluding* the parts highlighted in red and *including* the parts highlighted in blue. Conversely, the model for $\mathsf{F}^{\mu!}_{\mathrm{cc}}$ is obtained by *excluding* the parts highlighted in blue and *including* the parts highlighted in red. Accordingly, the model for $\mathsf{F}^{\mu!}_{\mathrm{cc}}$ allows only one transition relation ($\sqsupseteq_{\mathrm{pub}}$) in a world. This is because, in the presence of callcc, every transition is observable and thus public [5].

B.2.1 Definition.

$\boxed{h, n, e \hookrightarrow h', n', e'}$

$$\frac{h, e \hookrightarrow h', e'}{h, n, e \hookrightarrow h', n', e'} \qquad \frac{n' < n}{h, n, e \hookrightarrow h, n', e}$$

For the construction of the model, we extend the syntax of types with type names $\mathbf{n}$. They are used in modelling existential and universal types.

$$\begin{aligned}
&\mathrm{CVal} := \{v \in \mathrm{Val} \mid \mathrm{fv}(v) = \emptyset\}\\
&\mathrm{CExp} := \{e \in \mathrm{Exp} \mid \mathrm{fv}(e) = \emptyset\}\\
&\mathrm{CCont} := \{K \in \mathrm{Cont} \mid \mathrm{fv}(K[\langle\rangle]) = \emptyset\}\\
&\mathrm{CTy} := \{\tau \in \mathrm{Ty} \mid \mathrm{fv}(\tau) = \emptyset\}\\
&\mathrm{CTyF} := \{\tau_1 \to \tau_2 \mid \tau_1, \tau_2 \in \mathrm{CTy}\}
\end{aligned}$$

$$\begin{aligned}
&\mathrm{DRel} := \mathbb{P}((\mathrm{CTyF} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal}) \uplus (\mathrm{CTy} \times \mathbb{N} \times \mathrm{CCont} \times \mathbb{N} \times \mathrm{CCont}))\\
&\mathrm{VRel} := \mathbb{P}(\mathrm{CTyF} \times \mathbb{N} \times \mathrm{CVal} \times \mathbb{N} \times \mathrm{CVal})\\
&\mathrm{ERel} := \mathbb{P}(\mathbb{N} \times \mathrm{CExp} \times \mathbb{N} \times \mathrm{CExp})\\
&\mathrm{HRel} := \mathbb{P}(\mathrm{Heap} \times \mathrm{Heap})
\end{aligned}$$

$$\begin{array}{lcl}
\multicolumn{3}{l}{\overline{(-)} \in \mathrm{DRel} \to \mathrm{VRel}}\\
\overline{R}(\mathbf{n}) & := & \{(n_1, v_1, n_2, v_2) \in R(\mathbf{n})\}\\
\overline{R}(\mathsf{int}) & := & \{(n_1, m, n_2, m)\}\\
\overline{R}(\tau \times \tau') & := & \{(n_1, \langle v_1, v_1'\rangle, n_2, \langle v_2, v_2'\rangle) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\tau)\}\\
 & \cap & \{(n_1, \langle v_1, v_1'\rangle, n_2, \langle v_2, v_2'\rangle) \mid \exists m_1', m_2'.\ (m_1', v_1', m_2', v_2') \in \overline{R}(\tau')\}\\
\overline{R}(\tau + \tau') & := & \{(n_1, \mathsf{inl}\, v_1, n_2, \mathsf{inl}\, v_2) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\tau)\}\\
 & \cup & \{(n_1, \mathsf{inr}\, v_1', n_2, \mathsf{inr}\, v_2') \mid \exists m_1', m_2'.\ (m_1', v_1', m_2', v_2') \in \overline{R}(\tau')\}\\
\overline{R}(\tau \to \tau') & := & \{(n_1, v_1, n_2, v_2) \in R(\tau \to \tau')\}\\
\overline{R}(\exists\alpha.\, \sigma) & := & \{(n_1, \mathsf{pack}\ v_1, n_2, \mathsf{pack}\ v_2) \mid \exists m_1, m_2, \tau.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\sigma[\tau/\alpha])\}\\
\overline{R}(\forall\alpha.\, \sigma) & := & \{(n_1, v_1, n_2, v_2) \in R(\forall\alpha.\, \sigma)\}\\
\overline{R}(\mu\alpha.\, \sigma) & := & \{(n_1, \mathsf{roll}\ v_1, n_2, \mathsf{roll}\ v_2) \mid \exists m_1, m_2.\ (m_1, v_1, m_2, v_2) \in \overline{R}(\sigma[\mu\alpha.\, \sigma/\alpha])\}\\
\overline{R}(\mathsf{ref}\, \tau) & := & \{(n_1, v_1, n_2, v_2) \in R(\mathsf{ref}\, \tau)\}\\
\overline{R}(\mathsf{cont}\, \tau) & := & \{(n_1, \mathsf{cont}\ K_1, n_2, \mathsf{cont}\ K_2) \mid \exists m_1, m_2.\ (m_1, K_1, m_2, K_2) \in R(\tau)\}
\end{array}$$

$$
\begin{aligned}
\mathrm{DepWorld}(P) := \{(&\mathrm{S} \in \mathrm{Set},\\
&\sqsubseteq \in \mathbb{P}(\mathrm{S} \times \mathrm{S}),\\
&\sqsubseteq_{\mathrm{pub}} \in \mathbb{P}(\mathrm{S} \times \mathrm{S}),\\
&\mathrm{N} \in \mathbb{P}(\mathrm{TyNam}),\\
&\mathrm{L} \in (\mathrm{S}_P \to \mathrm{S} \to \mathrm{DRel}) \to \mathrm{S}_P \to \mathrm{S} \to \mathrm{DRel},\\
&\mathrm{H} \in (\mathrm{S}_P \to \mathrm{S} \to \mathrm{DRel}) \to \mathrm{S}_P \to \mathrm{S} \to \mathrm{HRel}) \mid\\
&\mathrm{countable}(\mathrm{S}) \wedge \mathrm{preorder}(\mathrm{S}, \sqsubseteq) \wedge \mathrm{preorder}(\mathrm{S}, \sqsubseteq_{\mathrm{pub}}) \wedge\\
&(\forall R, s_P, s.\ \forall (\tau, n_1, d_1, n_2, d_2) \in \mathrm{L}(R)(s_P)(s).\\
&\quad \forall R' \supseteq R, s'_P \sqsupseteq_P s_P, s' \sqsupseteq_{\mathrm{pub}} s, n'_1 \geq n_1, n'_2 \geq n_2.\\
&\quad (\tau, n'_1, d_1, n'_2, d_2) \in \mathrm{L}(R')(s'_P)(s')) \wedge\\
&(\forall R, s_P, s.\ \forall R' \supseteq R \cap (\mathrm{S}_P \to \mathrm{S} \to \mathrm{VRel}), s' \sqsupseteq s.\\
&\quad \mathrm{L}(R')(s_P)(s') \supseteq \mathrm{L}(R)(s_P)(s) \cap \mathrm{VRel}) \wedge\\
&(\forall R.\ \forall R' \supseteq R \cap (\mathrm{S}_P \to \mathrm{S} \to \mathrm{VRel}).\\
&\quad \mathrm{H}(R') \supseteq \mathrm{H}(R) \cap (\mathrm{S}_P \to \mathrm{S} \to \mathrm{VRel})) \wedge\\
&(\forall R, s_P, s.\ \forall (\mathbf{n}, n_1, v_1, n_2, v_2) \in \mathrm{L}(R)(s_P)(s).\ \mathbf{n} \in \mathrm{N})\}
\end{aligned}
$$

where $P = (\mathrm{S}_P \in \mathrm{Set}, \sqsubseteq_P : \mathbb{P}(\mathrm{S}_P \times \mathrm{S}_P))$

$$
\mathrm{World} := \{W \in \mathrm{DepWorld}(\{*\}, \{(*, *)\})\}
$$

$$
\begin{aligned}
&W_{\mathrm{ref}}.\mathrm{S} := \{s_{\mathrm{ref}} \in \mathbb{P}_{\mathrm{fin}}(\mathrm{CTy} \times \mathrm{Loc} \times \mathrm{Loc}) \mid\\
&\quad \forall (\tau, l_1, l_2) \in s_{\mathrm{ref}}.\ \forall (\tau', l'_1, l'_2) \in s_{\mathrm{ref}}.\\
&\qquad (l_1 = l'_1 \implies \tau = \tau' \wedge l_2 = l'_2) \wedge\\
&\qquad (l_2 = l'_2 \implies \tau = \tau' \wedge l_1 = l'_1)\}\\
&W_{\mathrm{ref}}.\sqsubseteq := \subseteq\\
&W_{\mathrm{ref}}.\sqsubseteq_{\mathrm{pub}} := \subseteq\\
&W_{\mathrm{ref}}.\mathrm{N} := \emptyset\\
&W_{\mathrm{ref}}.\mathrm{L}(R)(s_{\mathrm{ref}}) := \{(\mathsf{ref}\,\tau, l_1, l_2) \mid (\tau, l_1, l_2) \in s_{\mathrm{ref}}\}\\
&W_{\mathrm{ref}}.\mathrm{H}(R)(s_{\mathrm{ref}}) := \{(h_1, h_2) \mid\\
&\qquad \mathrm{dom}(h_1) = \{l_1 \mid \exists \tau, l_2.\ (\tau, l_1, l_2) \in s_{\mathrm{ref}}\} \wedge\\
&\qquad \mathrm{dom}(h_2) = \{l_2 \mid \exists \tau, l_1.\ (\tau, l_1, l_2) \in s_{\mathrm{ref}}\} \wedge\\
&\qquad \forall (\tau, l_1, l_2) \in s_{\mathrm{ref}}.\ \exists n_1, n_2.\ (n_1, h_1(l_1), n_2, h_2(l_2)) \in \overline{R(s_{\mathrm{ref}})}(\tau)\}
\end{aligned}
$$

$$
\mathrm{LWorld} := \{w \in \mathrm{DepWorld}(W_{\mathrm{ref}}.\mathrm{S}, W_{\mathrm{ref}}.\sqsubseteq_{\mathrm{pub}}\}) \mid \forall R, s_{\mathrm{ref}}, s, \tau.\ \mathrm{L}(R)(s_{\mathrm{ref}})(s)(\mathsf{ref}\,\tau) = \emptyset\}
$$

$$
\begin{aligned}
R^{s_2}_{[1]}(s_{\mathrm{ref}})(s_1) \quad := \quad & \{(\tau, n_1, d_1, n_2, d_2) \mid \exists s'_1 \sqsubseteq_{\mathrm{pub}} s_1, n'_1 \leq n_1, n'_2 \leq n_2.\\
& \qquad (\tau, n'_1, d_1, n'_2, d_2) \in R(s_{\mathrm{ref}})(s'_1, s_2)\}\\
\cup \quad & \{(\tau, n_1, v_1, n_2, v_2) \mid \exists s'_1 \sqsubseteq s_1, n'_1 \leq n_1, n'_2 \leq n_2.\\
& \qquad (\tau, n'_1, v_1, n'_2, v_2) \in R(s_{\mathrm{ref}})(s'_1, s_2)\}\\
R^{s_1}_{[2]}(s_{\mathrm{ref}})(s_2) \quad := \quad & \{(\tau, n_1, d_1, n_2, d_2) \mid \exists s'_2 \sqsubseteq_{\mathrm{pub}} s_2, n'_1 \leq n_1, n'_2 \leq n_2.\\
& \qquad (\tau, n'_1, d_1, n'_2, d_2) \in R(s_{\mathrm{ref}})(s_1, s'_2)\}\\
\cup \quad & \{(\tau, n_1, v_1, n_2, v_2) \mid \exists s'_2 \sqsubseteq s_2, n'_1 \leq n_1, n'_2 \leq n_2.\\
& \qquad (\tau, n'_1, v_1, n'_2, v_2) \in R(s_{\mathrm{ref}})(s_1, s'_2)\}
\end{aligned}
$$

$$
\begin{aligned}
(w_1 \otimes w_2).\mathrm{S} \quad &:= \quad w_1.\mathrm{S} \times w_2.\mathrm{S}\\
(w_1 \otimes w_2).\sqsubseteq \quad &:= \quad \{(p, p') \mid p.1 \sqsubseteq p'.1 \wedge p.2 \sqsubseteq p'.2\}\\
(w_1 \otimes w_2).\sqsubseteq_{\mathrm{pub}} \quad &:= \quad \{(p, p') \mid p.1 \sqsubseteq_{\mathrm{pub}} p'.1 \wedge p.2 \sqsubseteq_{\mathrm{pub}} p'.2\}\\
(w_1 \otimes w_2).\mathrm{N} \quad &:= \quad w_1.\mathrm{N} \uplus w_2.\mathrm{N}\\
(w_1 \otimes w_2).\mathrm{L}(R)(s_{\mathrm{ref}})(s_1, s_2) \quad &:= \quad w_1.\mathrm{L}(R^{s_2}_{[1]})(s_{\mathrm{ref}})(s_1) \cup w_2.\mathrm{L}(R^{s_1}_{[2]})(s_{\mathrm{ref}})(s_2)\\
(w_1 \otimes w_2).\mathrm{H}(R)(s_{\mathrm{ref}})(s_1, s_2) \quad &:= \quad w_1.\mathrm{H}(R^{s_2}_{[1]})(s_{\mathrm{ref}})(s_1) \otimes w_2.\mathrm{H}(R^{s_1}_{[2]})(s_{\mathrm{ref}})(s_2)
\end{aligned}
$$

$$
\begin{aligned}
w{\uparrow}.\mathrm{S} \quad &:= \quad W_{\mathrm{ref}}.\mathrm{S} \times w.\mathrm{S}\\
w{\uparrow}.\sqsubseteq \quad &:= \quad \{(p, p') \mid p.1 \sqsubseteq p'.1 \wedge p.2 \sqsubseteq p'.2\}\\
w{\uparrow}.\sqsubseteq_{\mathrm{pub}} \quad &:= \quad \{(p, p') \mid p.1 \sqsubseteq_{\mathrm{pub}} p'.1 \wedge p.2 \sqsubseteq_{\mathrm{pub}} p'.2\}\\
w{\uparrow}.\mathrm{N} \quad &:= \quad w.\mathrm{N}\\
w{\uparrow}.\mathrm{L}(R)(s_{\mathrm{ref}}, s) \quad &:= \quad W_{\mathrm{ref}}.\mathrm{L}(\emptyset)(s_{\mathrm{ref}}) \cup w.\mathrm{L}(R(-,-))(s_{\mathrm{ref}})(s)\\
w{\uparrow}.\mathrm{H}(R)(s_{\mathrm{ref}}, s) \quad &:= \quad W_{\mathrm{ref}}.\mathrm{H}(R(-, s))(s_{\mathrm{ref}}) \otimes w.\mathrm{H}(R(-,-))(s_{\mathrm{ref}})(s)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{GK}(W) := \{ & G \in W.\mathrm{S} \to \mathrm{DRel} \mid \\
& G \supseteq W.\mathrm{L}(G) \land \\
& (\forall s, \tau.\ G(s)(\mathsf{ref}\,\tau) \subseteq W.\mathrm{L}(G)(s)(\mathsf{ref}\,\tau)) \land \\
& (\forall s.\ \forall \mathbf{n} \in W.\mathrm{N}.\ G(s)(\mathbf{n}) \subseteq W.\mathrm{L}(G)(s)(\mathbf{n})) \land \\
& (\forall s.\ (0, \bullet, 0, \bullet) \in G(s)(\mathsf{int})) \land \\
& (\forall s.\ \forall s' \sqsupseteq s.\ G(s') \supseteq G(s) \cap \mathrm{VRel}) \land \\
& (\forall s.\ \forall (\tau, n_1, d_1, n_2, d_2) \in G(s).\ \forall s' \sqsupseteq_{\mathrm{pub}} s, n_1' \geq n_1, n_2' \geq n_2. \\
& \quad (\tau, n_1', d_1, n_2', d_2) \in G(s'))\}
\end{aligned}
$$

$$
\begin{aligned}
& \mathrm{S} \in \mathrm{DRel} \times \mathrm{DRel} \to \mathrm{ERel} \\
& \mathrm{S}(R, G) := \{(k_1 + n_1, K_1[v_1], k_2 + n_2, K_2[v_2]) \mid \\
& \qquad \exists \tau.\ (k_1, K_1, k_2, K_2) \in R(\tau) \land (n_1, v_1, n_2, v_2) \in \overline{G}(\tau)\} \cup \\
& \quad \{(n_1, K_1[v_1\ v_1'], n_2, K_2[v_2\ v_2']) \mid \\
& \qquad \exists \tau, \tau', m_1, m_2, k_1, k_2.\ (n_1, v_1, n_2, v_2) \in R(\tau' \to \tau) \land \\
& \qquad (m_1, v_1', m_2, v_2') \in \overline{G}(\tau') \land (k_1, K_1, k_2, K_2) \in G(\tau)\} \cup \\
& \quad \{(n_1, K_1[v_1[]], n_2, K_2[v_2[]]) \mid \\
& \qquad \exists \sigma, \tau, k_1, k_2.\ (n_1, v_1, n_2, v_2) \in R(\forall \alpha.\, \sigma) \land \\
& \qquad (k_1, K_1, k_2, K_2) \in G(\sigma[\tau/\alpha])\}
\end{aligned}
$$

$$
\begin{aligned}
& \mathrm{E}_W \in \mathrm{GK}(W) \to \mathrm{DRel} \to W.\mathrm{S} \to \mathrm{ERel} \\
& \mathrm{E}_W(G)(s) := \{(n_1, e_1, n_2, e_2) \mid \forall (h_1, h_2) \in W.\mathrm{H}(G)(s).\ \forall h_1^{\mathrm{F}} \# h_1, h_2^{\mathrm{F}} \# h_2. \\
& \qquad (h_1 \uplus h_1^{\mathrm{F}}, e_1 \hookrightarrow^{\omega} \land h_2 \uplus h_2^{\mathrm{F}}, e_2 \hookrightarrow^{\omega}) \lor \\
& \qquad (\exists s'.\ s' \sqsupseteq_{\mathrm{pub}} s \land s' \sqsupseteq s \land \\
& \qquad\quad \exists (n_1', e_1', n_2', e_2') \in \mathrm{S}(G(s'), G(s')).\ \exists (h_1', h_2') \in W.\mathrm{H}(G)(s'). \\
& \qquad\quad h_1 \uplus h_1^{\mathrm{F}}, n_1, e_1 \hookrightarrow^* h_1' \uplus h_1^{\mathrm{F}}, n_1', e_1' \land h_2 \uplus h_2^{\mathrm{F}}, n_2, e_2 \hookrightarrow^* h_2' \uplus h_2^{\mathrm{F}}, n_2', e_2')
\end{aligned}
$$

$$
\begin{aligned}
R^{\$} := \{ & (n_1, e_1, n_2, e_2) \mid \exists (n_1', e_1', n_2', e_2') \in R. \\
& (\forall h.\ h, n_1, e_1 \hookrightarrow h, n_1', e_1') \land (\forall h.\ h, n_2, e_2 \hookrightarrow h, n_2', e_2')\}
\end{aligned}
$$

$$
\mathrm{consistent}(W) \ := \ \forall G \in \mathrm{GK}(W), s \in W.\mathrm{S}.\ \mathrm{S}(W.\mathrm{L}(G)(s), G(s)) \subseteq \mathrm{E}_W(G)(s)^{\$}
$$

$$
\begin{aligned}
\mathrm{stable}(w) \ := \ & \forall G \in \mathrm{GK}(w\!\uparrow).\ \forall s_{\mathrm{ref}}, s.\ \forall (h_1, h_2) \in w.\mathrm{H}(G(-,-))(s_{\mathrm{ref}})(s). \\
& \forall s_{\mathrm{ref}}' \sqsupseteq_{\mathrm{pub}} s_{\mathrm{ref}}.\ \forall (h_1', h_2') \in W_{\mathrm{ref}}.\mathrm{H}(G(-,s))(s_{\mathrm{ref}}'). \\
& h_1' \# h_1 \land h_2' \# h_2 \implies \exists s' \sqsupseteq_{\mathrm{pub}} s.\ (h_1, h_2) \in w.\mathrm{H}(G(-,-))(s_{\mathrm{ref}}')(s')
\end{aligned}
$$

$$
\mathrm{inhabited}(W) \ := \ \exists s.\ \forall G \in \mathrm{GK}(W).\ (\emptyset, \emptyset) \in W.\mathrm{H}(G)(s)
$$

$$
\begin{aligned}
& R(\cdot) := \{(\cdot, \mathrm{id}, \cdot, \mathrm{id})\} \\
& R(x{:}\tau, \Gamma) := \{(n_1 :: N_1, \gamma_1[x \mapsto v_1], n_2 :: N_2, \gamma_2[x \mapsto v_2]) \mid \\
& \qquad (n_1, v_1, n_2, v_2) \in R(\tau) \land (N_1, \gamma_1, N_2, \gamma_2) \in R(\Gamma)\}
\end{aligned}
$$

$$
\begin{aligned}
\Delta; \Gamma \vdash e_1 \sim e_2 : \tau \ := \ & \forall \mathcal{N} \in \mathbb{P}(\mathrm{TyNam}).\ \mathcal{N} \text{ countably infinite} \implies \exists w, f_1, f_2. \\
& w.\mathrm{N} \subseteq \mathcal{N} \land \mathrm{stable}(w) \land \mathrm{consistent}(w\!\uparrow) \land \mathrm{inhabited}(w\!\uparrow) \land \Delta; \Gamma \vdash e_1 \sim_{w\uparrow, f_1, f_2} e_2 : \tau
\end{aligned}
$$

$$
\begin{aligned}
\Gamma \vdash e_1 \sim_{W, f_1, f_2} e_2 : \tau \ := \ & \forall G \in \mathrm{GK}(W).\ \forall s.\ \forall \delta \in \Delta \to \mathrm{CTy}.\ \forall (N_1, \gamma_1, N_2, \gamma_2) \in \overline{G(s)}(\delta\Gamma). \\
& \forall (k_1, K_1, k_2, K_2) \in G(s)(\delta\tau).\ (f_1(N_1) + k_1, K_1[\gamma_1 e_1], f_2(N_2) + k_2, K_2[\gamma_2 e_2]) \in \mathrm{E}_W(G)(s)\}
\end{aligned}
$$

B.2.2 Example: Well-Bracketed State Change.

For $\mathsf{F}^{\mu !}$, we prove $\vdash v_1 \sim e_2 : \tau$, where:

$$
\begin{aligned}
\tau & := (\mathsf{unit} \to \mathsf{unit}) \to \mathsf{int} \\
v_1 & := \lambda f.\, (f\ \langle\rangle; f\ \langle\rangle; 1) \\
v_2 & := \lambda f.\, (x := 0; f\ \langle\rangle; x := 1; f\ \langle\rangle; !x) \\
e_2 & := \mathsf{let}\ x = \mathsf{ref}\ 0\ \mathsf{in}\ v_2
\end{aligned}
$$

Constructing a Suitable World. We construct a world $w\!\uparrow$ that we will then show to be consistent and to relate v_1 and e_2.

Since the programs don't involve abstract types, we can define $w.\mathrm{N}$ to be empty. A state $s \in w.\mathrm{S}$ is to be understood as follows: for each running instance of e_2, identified by the location l that that instance initially allocated, $s(l)$ says whether the instance is in the 0-state (l points to 0) or in the 1-state (l points to 1). Accordingly, the heap relation $w.\mathrm{H}$ at state s is just $\{(\emptyset, s)\}$. Finally, the local knowledge $w.\mathrm{L}$ at state s relates v_1 with $v_2[l/x]$ for any location l belonging to an instance.

$$\begin{array}{ll}
w.\mathrm{S} & := \mathrm{Loc} \xrightarrow{\text{fin}} \{0,1\} \subseteq \mathrm{Heap} \\
w.\sqsubseteq & := \{(s, s') \mid \mathrm{dom}(s) \subseteq \mathrm{dom}(s')\} \\
w.\sqsubseteq_{\mathrm{pub}} & := \{(s, s') \in w.\sqsubseteq \mid \forall (l, 1) \in s.\ (l, 1) \in s'\} \\
w.\mathrm{N} & := \emptyset \\
w.\mathrm{L}(R)(s_{\mathrm{ref}})(s) & := \{(\tau, _, v_1, _, v_2[l/x]) \mid l \in \mathrm{dom}(s)\} \\
& \cup\ \{(\mathsf{unit}, _, K_1[\bullet; v_1'\ \langle\rangle; 1], _, K_2[\bullet; l := 1; v_2'\ \langle\rangle; !l]) \mid \exists s'. \\
& \quad\ s \sqsupseteq_{\mathrm{pub}} (s' \backslash l) \uplus [l \mapsto 0] \wedge l \in \mathrm{dom}(s') \wedge \\
& \quad\ (_, v_1', _, v_2') \in R(s_{\mathrm{ref}})(s)(\mathsf{unit} \to \mathsf{unit}) \wedge (_, K_1, _, K_2) \in R(s_{\mathrm{ref}})(s')(\mathsf{int})\} \\
& \cup\ \{(\mathsf{unit}, _, K_1[\bullet; 1], _, K_2[\bullet; !l]) \mid s(l) = 1 \wedge (_, K_1, _, K_2) \in R(s_{\mathrm{ref}})(s)(\mathsf{int})\} \\
w.\mathrm{H}(R)(s_{\mathrm{ref}})(s) & := \{(\emptyset, s)\}
\end{array}$$

It is easy to see that $w \in \mathrm{LWorld}$. In particular, $w.\mathrm{L}$ and $w.\mathrm{H}$ are monotone as required. Note that $\mathrm{stable}(w)$ (the dependency is vacuous) and that $\mathrm{inhabited}(w\!\uparrow)$ for $s_0 = \emptyset$. To show $\vdash v_1 \sim e_2 : \tau$, two parts remain.

Consistency. Establishing $\mathrm{consistent}(w\!\uparrow)$ is the real meat of the proof.

1) Let $G \in \mathrm{GK}(w\!\uparrow)$ and consider two functions related by $w\!\uparrow.\mathrm{L}(G)$ at a state (s_{ref}, s). Clearly, one is v_1 and the other is $v_2[l/x]$ for some $l \in \mathrm{dom}(s)$. So suppose we are given related continuations $(\_, K_1, \_, K_2) \in G(s_{\mathrm{ref}}, s)(\mathsf{int})$ and arguments $(v_1', v_2') \in \overline{G(s_{\mathrm{ref}}, s)}(\mathsf{unit} \to \mathsf{unit})$ and let n_1, n_2 be arbitrary. After performing a beta step on either side, we need to show:
$$(\_, K_1[v_1'\ \langle\rangle; v_1'\ \langle\rangle; 1], \_, K_2[l := 0; v_2'\ \langle\rangle; l := 1; v_2'\ \langle\rangle; !l]) \in \mathbf{E}_{w\uparrow}(G)(s_{\mathrm{ref}}, s)$$
Note that for $(h_1, h_2) \in w.\mathrm{H}(G(-,-))(s_{\mathrm{ref}})(s)$ we know by construction that $h_1 = \emptyset$ and $h_2 = s$. Consequently, for any frame heaps $h_1^{\mathrm{F}}, h_2^{\mathrm{F}}$, we have
$$\begin{array}{l} h_2 \uplus h_2^{\mathrm{F}}, K_2[l := 0; v_2'\ \langle\rangle; l := 1; v_2'\ \langle\rangle; !l] \hookrightarrow^* \\ (s \backslash l) \uplus [l \mapsto 0] \uplus h_2^{\mathrm{F}}, K_2[v_2'\ \langle\rangle; l := 1; v_2'\ \langle\rangle; !l] \end{array}$$
where $s \backslash l$ denotes the restriction of s to domain $\mathrm{dom}(s) \setminus \{l\}$. It thus suffices, by the "stuck function call" case, to find $s' \sqsupseteq s$ such that:
   a) $(\emptyset, (s \backslash l) \uplus [l \mapsto 0]) \in w.\mathrm{H}(G)(s_{\mathrm{ref}})(s')$
   b) $(\_, K_1[\bullet; v_1'\ \langle\rangle; 1], \_, K_2[\bullet; l := 1; v_2'\ \langle\rangle; !l]) \in G(s_{\mathrm{ref}}, s')(\mathsf{unit})$

   Naturally, we pick $s' = (s \backslash l) \uplus [l \mapsto 0] \sqsupseteq s$. Then both (a) and (b) hold by construction of w.
2) Now suppose $G \in \mathrm{GK}(L)$, s_{ref} and s arbitrary, $s' \sqsupseteq_{\mathrm{pub}} (s \backslash l) \uplus [l \mapsto 0]$, $l \in \mathrm{dom}(s)$, $(\_, v_1', \_, v_2') \in G(s_{\mathrm{ref}}, s')(\mathsf{unit} \to \mathsf{unit})$ and $(\_, K_1, \_, K_2) \in G(s_{\mathrm{ref}}, s)(\mathsf{int})$. We must show (after one step of reduction):
$$(\_, K_1[v_1'\ \langle\rangle; 1], \_, K_2[l := 1; v_2'\ \langle\rangle; !l]) \in \mathbf{E}(G)(s_{\mathrm{ref}}, s')$$
After repeating the previous procedure one more time, we arrive at the goal of finding $s'' \sqsupseteq s'$ such that:
   a) $(\emptyset, (s' \backslash l) \uplus [l \mapsto 1]) \in w.\mathrm{H}(G)(s_{\mathrm{ref}})(s'')$
   b) $(\_, K_1[\bullet; 1], \_, K_2[\bullet; !l]) \in G(s_{\mathrm{ref}}, s'')(\mathsf{unit})$

   Naturally, we pick $s'' = (s' \backslash l) \uplus [l \mapsto 1] \sqsupseteq s'$. Then both (a) and (b) hold by construction of w, where, for (b), we rely on $s'' \sqsupseteq_{\mathrm{pub}} s$.
3) Finally, suppose $G \in \mathrm{GK}(L)$, s_{ref} and s arbitrary, $s(l) = 1$ and $(\_, K_1, \_, K_2) \in G(s_{\mathrm{ref}}, s)(\mathsf{int})$. We must show (after one step of reduction):
$$(\_, K_1[1], \_, K_2[!l]) \in \mathbf{E}(G)(s_{\mathrm{ref}}, s)$$
Since $s(l) = 1$, we know for any $(h_1, h_2) \in w.\mathrm{H}(s_{\mathrm{ref}})(s)(G)(s_{\mathrm{ref}}, s)$ by construction that $h_2(l) = 1$. Consequently, for any frame heap h_2^{F} we have:
$$h_2 \uplus h_2^{\mathrm{F}}, K_2[!l] \hookrightarrow h_2 \uplus h_2^{\mathrm{F}}, K_2[1]$$
Since of course $(1, 1) \in \overline{G(s_{\mathrm{ref}}, s)}(\mathsf{int})$ by definition, we are done with $(\_, K_1, \_, K_2) \in G(s_{\mathrm{ref}}, s)(\mathsf{int})$.

Showing the Programs Related. Given how we constructed our world, this final goal is fairly easy to accomplish. Formally, we prove $\vdash v_1 \sim_{w\uparrow,0,0} e_2 : \tau$, *i.e.,* we must show

$$(k_1, K_1[v_1], k_2, K_2[e_2]) \in \mathbf{E}_{w\uparrow}(G)(s_{\mathrm{ref}}, s)$$

for any $G \in \mathrm{GK}(w\uparrow)$, $(k_1, K_1, k_2, K_2) \in G(s_{\mathrm{ref}}, s)(\tau)$, s_{ref}, s. Note that if $(h_1, h_2) \in w.\mathrm{H}(G(-,-))(s_{\mathrm{ref}})(s)$, then for any frame heap h_2^{F} and some fresh l we have $h_2 \uplus h_2^{\mathrm{F}}, K_2[e_2] \hookrightarrow h_2 \uplus [l{\mapsto}0] \uplus h_2^{\mathrm{F}}, K_2[v_2[l/x]]$. It therefore suffices to find $s' \sqsupseteq s$ such that the following hold:

4) $(k_1, K_1, \_, K_2) \in G(s_{\mathrm{ref}}, s')(\tau)$
5) $(0, v_1, \_, v_2[l/x]) \in \overline{G(s_{\mathrm{ref}}, s')}(\tau)$
6) $(h_1, h_2 \uplus [l{\mapsto}0]) \in w\uparrow.\mathsf{H}(G)(s_{\mathrm{ref}}, s')$

We pick $s' = s \uplus [l{\mapsto}0]$. Note that s' is well-defined because l is fresh (so $l \notin \mathrm{dom}(s)$), and also that $s' \sqsupseteq_{\mathrm{pub}} s$. The latter and monotonicity of the continuation knowledge imply (5). To show (6), it suffices by definition of GK to show $(0, v_1, \_, v_2[l/x]) \in w.\mathsf{L}(G(-,-))(s_{\mathrm{ref}})(s')(\tau)$. This holds by construction of w and s', and so does (7).