A Complete Characterization of Observational Equivalence in Polymorphic λ-Calculus with General References[⋆]

Eijiro Sumii

Tohoku University
sumii@ecei.tohoku.ac.jp

Abstract. We give the first sound and complete proof method for observational equivalence in full polymorphic λ-calculus with existential types and first-class, higher-order references. Our method is syntactic and elementary in the sense that it only employs simple structures such as relations on terms. It is nevertheless powerful enough to prove many interesting equivalences that can and cannot be proved by previous approaches, including the latest work by Ahmed, Dreyer and Rossberg (POPL 2009).

1 Introduction

Data abstraction and local state are both known to introduce interesting properties—in particular, observational equivalences—into computer programs. Methodology for reasoning about such properties has been a major challenge in the fundamental research on programming languages (e.g., [14, 20]).

Recently, Ahmed, Dreyer and Rossberg [3] developed a technique, based on step-indexed Kripke-style logical relations, for proving observational equivalence in polymorphic λ-calculus with *both* abstract types and references. While their technique is (to our knowledge) the first "direct"—i.e., without encoding into other languages such as polymorphic π-calculus [18] or continuation passing style [13]—proof method for observational equivalence in this language, it is incomplete and specialized for particular cases (generative abstract data types). Indeed, some interesting equivalences cannot be proved by their method [3, Section 5.7 and 5.8]. Independently, Birkedal, Støvring and Thamsborg [6, 7] have also developed logical relations in a language with polymorphic (and recursive) types and references. However, they are also incomplete and not yet useful enough for reasoning about observational equivalence involving local state in general [7, Section 1 and 6][6, Section 1].

In this paper, we take a different approach, based on Sumii et al.'s environmental bisimulation [11, 21, 25, 26], and give the first sound and complete proof method for observational equivalence in call-by-value λ-calculus with impredicative universal and existential types, as well as "full" references (i.e., references

⋆ January 23, 2009. Last revised on August 4, 2010.

are first-class values and all values can be referred to, including functions and references themselves). Our development is not only complete, but also arguably simpler than other theories in that it only requires elementary notions such as terms, values, relations and sets without explicit need for metric spaces or step indices. Although observational equivalence is clearly undecidable in any Turing-complete language (in our case, general recursion can be encoded via higher-order references [17, Exercise 13.5.8]), we believe that our approach is useful for understanding and reasoning about information hiding in a wide range of settings including ours.

The rest of this paper is structured as follows. Section 2 discusses related work and our contributions with respect to it. Section 3 describes our language. Section 4 defines environmental bisimulation for this language and Section 5 develops up-to techniques. Section 6 proves the characterization theorem by using the up-to techniques. Section 7 proves examples of equivalences from [3] and Section 8 concludes with more comments.

2 Related work

The classic references on observational equivalences introduced by polymorphic types are Reynolds' relational parametricity [20] and Mitchell's representation independence [15]. Establishing a similar theory for local state has turned out to be highly challenging. The classic reference here is Meyer and Sieber [14]. Pitts and Stark [19] developed syntactic logical relations (i.e., logical relations over the term model) for λ-calculus with ML-like references. Their references are limited to the first order in the sense that only integers—not functions, nor references themselves—can be referred to. This is due to a difficulty involved in the circularity of "references to (functions containing) references." Ahmed, Appel and Virga [2] used step indices [4] to break this circularity, though they worked on a unary (rather than binary) model and type safety (rather than observational equivalence). A recent paper by Ahmed, Dreyer and Rossberg [3]—discussed in Section 1, 7 and 8—follows this line of work. Another line of work has been carried out by Birkedal et al. [6–8]. To the best of our knowledge, our method can prove strictly more examples of equivalences than all of the above approaches (though this is hard to prove generally, because completeness by itself does not always mean an automatic proof; recall that observational equivalence is undecidable in our language).

Abramsky, Honda and McCusker [1] (as well as Laird [12] and Tzevelekos [27]) developed a fully abstract game (or trace) semantics for simply typed λ-calculus with general references. They did not treat polymorphism, however. Our theory is more elementary in the sense that it requires very little machinery other than the syntax and operational semantics of the language itself, enabling the simple treatment of complex combinations like polymorphism and state. (Of course, this does *not* devalue game semantics at all: their whole point is syntax-freedom, while ours is the complete opposite, i.e., "semantics-freedom.")

Environmental bisimulation was first devised for untyped λ-calculus with encryption [25]. Since then, it has been applied to various languages, including polymorphic λ-calculus with existential types [26] and untyped λ-calculus with general references [11][21, Section 4]. The technical contributions of the present paper with respect to these are: (1) the *combination* of polymorphic types and references, which required careful handling of store typing (e.g., Definition 3 and 4), (2) the combination of small-step semantics and existential types, requiring a subtle adjustment to the context closure operation (Definition 9) and therefore to the up-to context technique (Definition 10), (3) a more powerful up-to reduction technique (Definition 7 and 8) that allows renaming of fresh locations, and (4) bisimulation proofs for non-trivial examples of observational equivalence in the present language, many of which have been considered hard traditionally (see, e.g., [3, 14, 19]).

Gordon [10] (as well as a number of papers that followed) considered bisimulations for functional languages with input and output effects (and for object calculi). To our knowledge, none of them treated references. Lassen et al. [13, 23] developed normal form bisimulations for polymorphic λ-calculus with control operators (and references) or in continuation passing style. Normal form bisimulations are generally incomplete with respect to contextual equivalence in languages without control operators (or in direct style) [23, Section 1][13, Section 1.1].

To summarize, the main thrust of our work is to give actual evidence that environmental bisimulations scale easily to various languages, including the present one with both polymorphism and state, which has been considered difficult in the long history of research in this area (again see [3, 14, 19] for instance).

3 The language

The syntax of our language is given in Figure 1. It is a standard polymorphic λ-calculus with existential types and references. We assume an infinite set of locations *Loc* and write $loc(M)$ for the set of locations that appear in term M. We use meta-variables $C, D, \dots$ for location-free (and possibly open) terms. We often omit type annotations when they are unimportant. We adopt the standard notion of bound variables and α-equivalence, and write $FV(M)$ and $FTV(\tau)$ for free variables and free type variables of M and τ, respectively. We use the nullary tuple $\langle\rangle$ and the nullary product type $\mathbf{1}$ as the unit value and the unit type.

The typing rules and the left-to-right, call-by-value reduction relation are given by judgments of the forms $S, \Gamma, \Sigma \vdash M : \tau$ and $s \triangleright M \to t \triangleright N$, where S is a set of type variables, Γ is a type environment (a partial map from variables to types), Σ is a store typing (a partial map from locations to closed types), and s and t are stores (a partial map from locations to values). Their definitions are standard [17]. Key rules involving polymorphism and state are shown in Figure 2 in the appendices. As usual, S, Γ, and Σ are omitted when they are empty. We write $\twoheadrightarrow$ for the reflexive and transitive closure of $\to$. In examples,

$\rho, \sigma, \tau ::=$	type
α	type variable
$\tau \to \sigma$	function type
$\forall \alpha.\, \tau$	universal type
$\exists \alpha.\, \tau$	existential type
$\tau_1 \times \cdots \times \tau_n$	product type
τ `ref`	reference type

$U, V, W ::=$	value
x	variable
$\lambda x : \tau.\, M$	function
$\Lambda \alpha.\, M$	type function
`pack` (τ, V) `as` $\exists \alpha.\, \sigma$	package
$\langle V_1, \ldots, V_n \rangle$	tuple
ℓ	location

$L, M, N, C, D ::=$	term
x	variable
$\lambda x : \tau.\, M$	abstraction
MN	application
$\Lambda \alpha.\, M$	type abstraction
$M[\tau]$	type application
`pack` (τ, M) `as` $\exists \alpha.\, \sigma$	packing
`open` M `as` (α, x) `in` N	opening
$\langle M_1, \ldots, M_n \rangle$	tupling
$\#_i(M)$	projection
ℓ	location
`ref` M	allocation
$!\,M$	dereference
$M := N$	update
$M_1 \overset{ptr}{=} M_2 \ ?\ N_1 : N_2$	pointer equality

$E ::=$	evaluation context
$[\,]$	hole
EM	application (left)
VE	application (right)
`pack` (τ, E) `as` $\exists \alpha.\, \sigma$	packing
`open` E `as` (α, x) `in` M	opening
$\langle V_1, \ldots, V_m, E, M_{m+1}, \ldots, M_n \rangle$	tupling
$\#_i(E)$	projection
`ref` E	allocation
$!\,E$	dereference
$E := M$	update (left)
$V := E$	update (right)
$E \overset{ptr}{=} M \ ?\ N_1 : N_2$	pointer equality (left)
$V \overset{ptr}{=} E \ ?\ N_1 : N_2$	pointer equality (right)

Fig. 1. Syntax

we use integers and Booleans, which are easy to add as primitives or encode as functions. We write $\{x \mapsto v\}$ for a finite map $\{(x, v)\}$ in general. We also write $f\{x \mapsto v\}$ for $\{(x, v)\} \cup \{(y, f(y)) \mid y \neq x\}$, and $f \uplus \{x \mapsto v\}$ for $f\{x \mapsto v\}$ only if $x \notin dom(f)$ (it is undefined otherwise).

For simplicity, we (very) often use the abbreviation $\overline{a}$ to mean the sequence $a_1, \ldots, a_n$ when n is unimportant, for any kind of meta-variable a. Furthermore, we often write $op(\overline{a}, \overline{b}, \ldots, \overline{c})$ to mean the sequence $op(a_1, b_1, \ldots, c_1), \ldots, op(a_n, b_n, \ldots, c_n)$ for various (meta-level) operators op. For instance, $\overline{x} : \overline{\tau}$ means $x_1 : \tau_1, \ldots, x_n : \tau_n$. We always take care that these notations do not create confusion or introduce ambiguity.

The following lemma is important for the "up-to reduction" technique explained in Section 5. Here, we use permutations π on locations because they behave better than substitutions [9].

Lemma 1. *Reduction is deterministic up to renaming of fresh locations. That is, if $s \triangleright M \to t \triangleright N$ and $s \triangleright M \to t_0 \triangleright N_0$, then $t_0 \triangleright N_0 = \pi(t \triangleright N)$ for some permutation π on $Loc \setminus dom(s)$.*

Proof. By induction on the derivation of $s \triangleright M \to t \triangleright N$.

Note that the above property is *not* trivial. For instance, reduction would be non-deterministic (even modulo renaming of fresh locations) under the presence of deallocation [24], which disallows the general up-to reduction technique.

The following definition and lemma observe that contexts are reduced either "by themselves without using the value in the hole" or else "by destructing the value in the hole."

Definition 1. *Variable x is* at the destruction position *in term M if M is of the form $E[xV]$, $E[x[\tau]]$, $E[\texttt{open}\ x\ \texttt{as}\ (\alpha, y)\ \texttt{in}\ N]$, $E[\#_i(x)]$, $E[!\,x]$, $E[x := V]$, $E[x \overset{ptr}{=} V\ ?\ N_1 : N_2]$ or $E[V \overset{ptr}{=} x\ ?\ N_1 : N_2]$.*

Note that destruction positions are different from redex positions, e.g., when $M = Vx$. Recall also that our reduction is call-by-value.

Lemma 2 (context reduction). *Suppose $\overline{\alpha}, \overline{x} : \overline{\tau} \vdash C_1 : \tau$. If C_1 is not a value and not of the form $E[\texttt{ref}\ V]$, and if no $x_i \in \{\overline{x}\}$ is at the destruction position in C_1, then for some C_2 with $\overline{\alpha}, \overline{x} : \overline{\tau} \vdash C_2 : \tau$, we have*

$$s \triangleright \theta C_1 \longrightarrow s \triangleright \theta C_2$$

for any s and $\theta = [\overline{V}/\overline{x}][\overline{\rho}/\overline{\alpha}]$ with $\Sigma \vdash s$ and $\Sigma \vdash \overline{V} : \theta\overline{\tau}$.

Proof. By induction on the syntax of C_1. All cases are trivial, given the standard type soundness theorems (i.e., progress and preservation).

4 Environmental bisimulation

We now define our environmental bisimulation. Readers are referred to previous work for more comprehensive introduction to environmental bisimulations, for polymorphic types (with big-step semantics) [26] or local state (with small-step semantics) [21, Section 1 and 4]. (A subsection in a recent paper [24, Section 1.3] would perhaps be the easiest introduction, even though their language is untyped and includes deallocation.)

Definition 2. *A* concretion environment *Δ is a partial map from type variables to pairs of closed types. We define $\Delta^1(\alpha) = \sigma$ and $\Delta^2(\alpha) = \sigma'$ if $\Delta(\alpha) = (\sigma, \sigma')$. We extend their domain from type variables to types in the obvious manner.*

Intuitively, $\Delta(\alpha) = (\sigma, \sigma')$ means that the abstract type α is implemented by σ on the left hand side of equivalence, and by σ' on the right.

Definition 3. *A* typed value relation *$\mathcal{R}$ is a set of triples of the form (V, V', τ). We write $\Delta, (\Sigma, \Sigma') \vdash \mathcal{R}$ if $\Sigma \vdash V : \Delta^1(\tau)$ and $\Sigma' \vdash V' : \Delta^2(\tau)$ for all $(V, V', \tau) \in \mathcal{R}$.*

Intuitively, $\mathcal{R}$ represents the "knowledge" of a context and $(V, V', \tau) \in \mathcal{R}$ means V (resp. V') is known under type τ to the context on the left (resp. right) hand side. Note that τ may be open (with $FTV(\tau) \subseteq dom(\Delta)$), while V and V' are closed (though they may still contain locations). Intuitively, free type variables in τ represent names of abstract data types.

Definition 4. *An* environmental relation *X is a set of tuples of the form $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau)$ or $(\Delta, \mathcal{R}, s, s')$ with appropriate typing, i.e.,*

– $\Delta, (\Sigma, \Sigma') \vdash \mathcal{R}$,
– $\Sigma \vdash M : \Delta^1(\tau)$ *with* $\Sigma \vdash s$, *and*
– $\Sigma' \vdash M' : \Delta^2(\tau)$ *with* $\Sigma' \vdash s'$

for some Σ *and* Σ'.

Again, note that τ may be open and contain free (i.e., abstract) type variables, while M and M' are closed.

Informally, X is a set of the states of a program and a context. $(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \in X$ means program M (resp. M') of type τ is running under store s (resp. s') on the left (resp. right) hand side, while $(\Delta, \mathcal{R}, s, s') \in X$ means that the two programs have stopped with stores s and s', respectively. In both cases, $\mathcal{R}$ represents the knowledge of the context (i.e., the environment) that has already been given out by the programs.

Definition 5. *The* context closure $(\Delta, \mathcal{R})^\star$ *of* $\mathcal{R}$ *under* Δ *is defined as:* $\{\ ([\overline{V}/\overline{x}]\Delta^1(C),\ [\overline{V}'/\overline{x}]\Delta^2(C),\ \tau)\ \mid\ dom(\Delta), \overline{x} : \overline{\tau} \vdash C : \tau,\ \ (\overline{V}, \overline{V}', \overline{\tau}) \in \mathcal{R}\ \}$

Informally, context closure represents synthesis of knowledge by contexts. With types omitted and infix notation used, it simply says: if $\overline{V}\mathcal{R}\overline{V}'$, then $([\overline{V}/\overline{x}]C)\mathcal{R}^\star$ $([\overline{V}'/\overline{x}]C)$. Recall that our context C is just a term with free variables $\overline{x}$.

The intuitions above lead to the following definition of environmental bisimulation, which asserts that X is preserved by execution (reduction and evaluation) of the program and by observations (application, type application, opening, projection, allocation, dereference, update, and pointer equality) from the context.

Definition 6. X *is an* environmental simulation *if:*

1. *For any* $(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \in X$,
   (a) [Reduction] *If* $s \rhd M \rightarrow t \rhd N$, *then* $s' \rhd M' \twoheadrightarrow t' \rhd N'$ *for some* t' *and* N' *with* $(\Delta, \mathcal{R}, t \rhd N, t' \rhd N', \tau) \in X$.
   (b) [Evaluation] *If* $M = V$, *then* $s' \rhd M' \twoheadrightarrow t' \rhd V'$ *for some* t' *and* V' *with* $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, s, t') \in X$.
2. *For any* $(\Delta, \mathcal{R}, s, s') \in X$,
   (a) [Application] *If* $(\lambda x\!:\!\Delta^1(\tau_1).\, M,\ \ \lambda x\!:\!\Delta^2(\tau_1).\, M',\ \ \tau_1 \rightarrow \tau_2) \in \mathcal{R}$, *then* $(\Delta,\ \mathcal{R},\ s \rhd [W/x]M,\ s' \rhd [W'/x]M',\ \tau_2) \in X$ *for any* $(W, W', \tau_1) \in (\Delta, \mathcal{R})^\star$.
   (b) [Type Application] *If* $(\Lambda\alpha.\, M,\ \Lambda\alpha.\, M',\ \forall\alpha.\, \tau) \in \mathcal{R}$, *then* $(\Delta,\ \mathcal{R},\ s \rhd [\Delta^1(\sigma)/\alpha]M,\ s' \rhd [\Delta^2(\sigma)/\alpha]M',\ [\sigma/\alpha]\tau) \in X$ *for any* σ *with* $FTV(\sigma) \subseteq dom(\Delta)$.
   (c) [Opening] *If* $(\texttt{pack}\ (\sigma, V)\ \texttt{as}\ \exists\alpha.\, \Delta^1(\tau),\ \texttt{pack}\ (\sigma', V')\ \texttt{as}\ \exists\alpha.\, \Delta^2(\tau),\ \exists\alpha.\, \tau) \in \mathcal{R}$, *then* $(\Delta \cup \{\alpha \mapsto (\sigma, \sigma')\}, \mathcal{R} \cup \{(V, V', \tau)\}, s, s') \in X$ *for some* α, *or else* $(\Delta, \mathcal{R} \cup \{(V, V', [\rho/\alpha]\tau)\}, s, s') \in X$ *for some* ρ *with* $FTV(\rho) \subseteq dom(\Delta)$, $\sigma = \Delta^1(\rho)$ *and* $\sigma' = \Delta^2(\rho)$.
   (d) [Projection] *If* $(\langle V_1, \ldots, V_n\rangle, \langle V_1', \ldots, V_n'\rangle, \tau_1 \times \cdots \times \tau_n) \in \mathcal{R}$, *then* $(\Delta, \mathcal{R} \cup \{(V_i, V_i', \tau_i)\}, s, s') \in X$ *for any* $i \in \{1, ..., n\}$.
   (e) [Allocation] $(\Delta, \mathcal{R} \cup \{(\ell, \ell', \tau\ \texttt{ref})\}, s \uplus \{\ell \mapsto W\}, s' \uplus \{\ell' \mapsto W'\}) \in X$ *for any* $\ell \notin dom(s)$, $\ell' \notin dom(s')$ *and* $(W, W', \tau) \in (\Delta, \mathcal{R})^\star$.

(f) *If* $(\ell, \ell', \tau\ \texttt{ref}) \in \mathcal{R}$*, then*

i. [Dereference] $(\Delta, \mathcal{R} \cup \{(s(\ell), s'(\ell'), \tau)\}, s, s') \in X$.

ii. [Update] $(\Delta, \mathcal{R}, s\{\ell \mapsto W\}, s'\{\ell' \mapsto W'\}) \in X$ *for any* $(W, W', \tau) \in (\Delta, \mathcal{R})^\star$.

(g) [Pointer Equality] *If* $(\ell, \ell'_1, \tau\ \texttt{ref}) \in \mathcal{R}$ *and* $(\ell, \ell'_2, \tau\ \texttt{ref}) \in \mathcal{R}$*, then* $\ell'_1 = \ell'_2$.

X is an environmental bisimulation if both X and X^{-1} are environmental simulations, where

$$\begin{aligned}
X^{-1} &= \{(\Delta^{-1}, \mathcal{R}^{-1}, s' \rhd M', s \rhd M, \tau) \mid (\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \in X\} \\
&\quad \cup \{(\Delta^{-1}, \mathcal{R}^{-1}, s', s) \mid (\Delta, \mathcal{R}, s, s') \in X\} \\
\mathcal{R}^{-1} &= \{(V', V, \tau) \mid (V, V', \tau) \in \mathcal{R}\}
\end{aligned}$$

and Δ^{-1} is defined (at the risk of confusion with the inverse map) by $dom(\Delta^{-1}) = dom(\Delta)$ and $\Delta^{-1}(\alpha) = (\sigma', \sigma)$ for any α with $\Delta(\alpha) = (\sigma, \sigma')$. Environmental bisimilarity $\sim$ is the largest environmental bisimulation, which exists because all the conditions above are monotone on X, i.e., the union of environmental (bi)simulations is again an environmental (bi)simulation.

5 Up-to techniques

As we shall prove in Section 6, environmental bisimilarity characterizes observational equivalence. However, the above definition by itself is not yet convenient enough for proving instances of observational equivalence between programs. As in concurrency theory [22], various up-to techniques are useful for getting rid of this inconvenience. Below, we report a few of such up-to techniques in our setting.

Definition 7. *The* reduction (and renaming) closure $X^{\rightarrow}$ *of X is defined as*

$$\begin{aligned}
X^{\rightarrow} = &\{(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \mid \\
&\qquad s \rhd M \twoheadrightarrow t \rhd N,\ \ s' \rhd M' \twoheadrightarrow t' \rhd N',\ \ (\Delta, \mathcal{R}, t \rhd N, t' \rhd N', \tau) \in \pi^1(X)\} \\
&\cup \{(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \mid s \rhd M \text{ diverges}\} \\
&\cup \{(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \mid \\
&\qquad s \rhd M \twoheadrightarrow t \rhd V,\ \ s' \rhd M' \twoheadrightarrow t' \rhd V',\ \ (\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, t, t') \in \pi^1(X)\} \\
&\cup \{(\Delta, \mathcal{R}, s, s') \mid (\Delta, \mathcal{R}, s, s') \in \pi^1(X)\}
\end{aligned}$$

where

$$\begin{aligned}
\pi^1(X) &= \{(\Delta, \pi^1(\mathcal{R}), \pi(s) \rhd \pi(M), s' \rhd M', \tau) \mid (\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \in X\} \\
&\quad \cup \{(\Delta, \pi^1(\mathcal{R}), \pi(s), s') \mid (\Delta, \mathcal{R}, s, s') \in X\} \\
\pi^1(\mathcal{R}) &= \{(\pi(V), V', \tau) \mid (V, V', \tau) \in \mathcal{R}\}.
\end{aligned}$$

In short, $X^{\rightarrow}$ is the set of elements that reduce or evaluate to some element of X (modulo renaming of locations). We "cross-sell" up-to reduction and up-to renaming, because our reduction is deterministic only up to renaming of fresh locations (Lemma 1).

Definition 8. *X is an* environmental simulation up-to reduction (and renaming) *if the conditions of Definition 6 hold with all the positive occurrences of X replaced by $X^{\rightarrow}$. To spell out,*

1. *For any $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X$,*
   (a) *If $s \triangleright M \rightarrow t \triangleright N$, then $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$ for some t' and N' with $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in X^{\rightarrow}$.*
   (b) *If $M = V$, then $s' \triangleright M' \twoheadrightarrow t' \triangleright V'$ for some t' and V' with $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, s, t') \in X^{\rightarrow}$.*
2. *For any $(\Delta, \mathcal{R}, s, s') \in X$,*
   (a) *If $(\lambda x : \Delta^1(\tau_1).\, M, \lambda x : \Delta^2(\tau_1).\, M', \tau_1 \rightarrow \tau_2) \in \mathcal{R}$, then $(\Delta, \mathcal{R}, s \triangleright [W/x]M, s' \triangleright [W'/x]M', \tau_2) \in X^{\rightarrow}$ for any $(W, W', \tau_1) \in (\Delta, \mathcal{R})^{\star}$.*
   (b) *If $(\Lambda\alpha.\, M, \Lambda\alpha.\, M', \forall\alpha.\, \tau) \in \mathcal{R}$, then $(\Delta, \mathcal{R}, s \triangleright [\Delta^1(\sigma)/\alpha]M, s' \triangleright [\Delta^2(\sigma)/\alpha]M', [\sigma/\alpha]\tau) \in X^{\rightarrow}$ for any σ with $FTV(\sigma) \subseteq dom(\Delta)$.*
   (c) *If $(\texttt{pack } (\sigma, V) \texttt{ as } \exists\alpha.\, \Delta^1(\tau), \texttt{pack } (\sigma', V') \texttt{ as } \exists\alpha.\, \Delta^2(\tau), \exists\alpha.\, \tau) \in \mathcal{R}$, then $(\Delta \cup \{\alpha \mapsto (\sigma, \sigma')\}, \mathcal{R} \cup \{(V, V', \tau)\}, s, s') \in X^{\rightarrow}$ for some α, or else $(\Delta, \mathcal{R} \cup \{(V, V', [\rho/\alpha]\tau)\}, s, s') \in X^{\rightarrow}$ for some ρ with $FTV(\rho) \subseteq dom(\Delta)$, $\sigma = \Delta^1(\rho)$ and $\sigma' = \Delta^2(\rho)$.*
   (d) *If $(\langle V_1, \ldots, V_n \rangle, \langle V_1', \ldots, V_n' \rangle, \tau_1 \times \cdots \times \tau_n) \in \mathcal{R}$, then $(\Delta, \mathcal{R} \cup \{(V_i, V_i')\}, s, s') \in X^{\rightarrow}$ for any $i \in \{1, ..., n\}$.*
   (e) *$(\Delta, \mathcal{R} \cup \{(\ell, \ell', \tau \texttt{ ref})\}, s \uplus \{\ell \mapsto W\}, s' \uplus \{\ell' \mapsto W'\}) \in X^{\rightarrow}$ for any $\ell \notin dom(s)$, $\ell' \notin dom(s')$ and $(W, W', \tau) \in (\Delta, \mathcal{R})^{\star}$.*
   (f) *If $(\ell, \ell', \tau \texttt{ ref}) \in \mathcal{R}$, then*
      i. *$(\Delta, \mathcal{R} \cup \{(s(\ell), s'(\ell'), \tau)\}, s, s') \in X^{\rightarrow}$.*
      ii. *$(\Delta, \mathcal{R}, s\{\ell \mapsto W\}, s'\{\ell' \mapsto W'\}) \in X^{\rightarrow}$ for any $(W, W', \tau) \in (\Delta, \mathcal{R})^{\star}$.*
   (g) *If $(\ell, \ell_1', \tau \texttt{ ref}) \in \mathcal{R}$ and $(\ell, \ell_2', \tau \texttt{ ref}) \in \mathcal{R}$, then $\ell_1' = \ell_2'$.*

Lemma 3 (soundness of up-to reduction). *Suppose X is an environmental simulation up-to reduction. Then $X^{\rightarrow}$ is an environmental simulation.*

Proof. We check each condition of Definition 6 for each element of $X^{\rightarrow}$ by expanding Definition 7. Details are found in Appendix A.

The next up-to technique is the most powerful one.

Definition 9. *The context (and environment) closure $X^{\star}$ of X is defined as:*

$$
\begin{aligned}
X^{\star} = \{(&\Delta,\ \mathcal{R},\ s \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)[M],\ s' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[M'],\ \tau) \mid \\
&(\Delta_0, \mathcal{S}, s \triangleright M, s' \triangleright M', \tau_0) \in X, \\
&\Delta \subseteq \Delta_0,\quad \mathcal{R} \subseteq (\Delta_0, \mathcal{S})^{\star},\quad FTV(\mathcal{R}) \subseteq dom(\Delta), \\
&(\overline{V}, \overline{V}', \overline{\tau}) \in \mathcal{S},\quad dom(\Delta_0), \overline{y} : \overline{\tau} \vdash E[\tau_0] : \tau,\quad FTV(\tau) \subseteq dom(\Delta)\} \\
\cup \{(&\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \mid \\
&(\Delta_0, \mathcal{S}, s, s') \in X,\quad \Delta \subseteq \Delta_0,\quad \mathcal{R} \subseteq (\Delta_0, \mathcal{S})^{\star},\quad FTV(\mathcal{R}) \subseteq dom(\Delta) \\
&(M, M', \tau) \in (\Delta_0, \mathcal{S})^{\star},\quad FTV(\tau) \subseteq dom(\Delta)\} \\
\cup \{(&\Delta, \mathcal{R}, s, s') \mid \\
&(\Delta_0, \mathcal{S}, s, s') \in X,\quad \Delta \subseteq \Delta_0,\quad \mathcal{R} \subseteq (\Delta_0, \mathcal{S})^{\star},\quad FTV(\mathcal{R}) \subseteq dom(\Delta)\}
\end{aligned}
$$

Here, $E[\tau_0]$ denotes an evaluation context E with a hole of type τ_0, rather than a type application.

As in Definition 5, the above definition is easier to understand if we omit types (and stores), and use an infix notation $\mathcal{S} \models M \; X \; M'$ for $(\mathcal{S}, M, M') \in X$.

- If $\mathcal{S} \models M \; X \; M'$ and $\overline{V} \; \mathcal{S} \; \overline{V}'$, then $\mathcal{R} \models ([\overline{V}/\overline{y}]E[M]) \; X^\star \; ([\overline{V}'/\overline{y}]E[M'])$ for any $\mathcal{R} \subseteq \mathcal{S}^\star$.
- If $\mathcal{S} \in X$ and $M \; \mathcal{S}^\star \; M'$, then $\mathcal{R} \models M \; X^\star \; M'$ for any $\mathcal{R} \subseteq \mathcal{S}^\star$.
- If $\mathcal{S} \in X$, then $\mathcal{R} \in X^\star$ for any $\mathcal{R} \subseteq \mathcal{S}^\star$.

Here, up-to context and up-to environment (the subset inclusion $\mathcal{R} \subseteq \mathcal{S}^\star$) are cross-sold because of small-step semantics: during reduction under context E or C, newly known values need to be substituted into their holes, but they cannot be added to $\mathcal{R}$ until the reduction terminates.

In the first item above, the restriction to evaluation contexts E is important (the up-to technique would otherwise be unsound in general) but is *not* a weakness of our approach: see Section 6.

Definition 10. *X is an* environmental simulation up-to reduction and context (and environment) *if the conditions of Definition 6 hold with all the positive occurrences of X replaced by $(X^\star)^\rightarrow$. To spell out,*

1. For any $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X$,
  (a) If $s \triangleright M \rightarrow t \triangleright N$, then $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$ for some t' and N' with $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in (X^\star)^\rightarrow$.
  (b) If $M = V$, then $s' \triangleright M' \twoheadrightarrow t' \triangleright V'$ for some t' and V' with $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, s, t') \in (X^\star)^\rightarrow$.

2. For any $(\Delta, \mathcal{R}, s, s') \in X$,
  (a) If $(\lambda x : \Delta^1(\tau_1).\, M, \lambda x : \Delta^2(\tau_1).\, M', \tau_1 \rightarrow \tau_2) \in \mathcal{R}$, then $(\Delta, \mathcal{R}, s \triangleright [W/x]M, s' \triangleright [W'/x]M', \tau_2) \in (X^\star)^\rightarrow$ for any $(W, W', \tau_1) \in (\Delta, \mathcal{R})^\star$.
  (b) If $(\Lambda\alpha.\, M, \Lambda\alpha.\, M', \forall\alpha.\, \tau) \in \mathcal{R}$, then $(\Delta, \mathcal{R}, s \triangleright [\Delta^1(\sigma)/\alpha]M, s' \triangleright [\Delta^2(\sigma)/\alpha]M', [\sigma/\alpha]\tau) \in (X^\star)^\rightarrow$ for any σ with $FTV(\sigma) \subseteq dom(\Delta)$.
  (c) If $(\texttt{pack } (\sigma, V) \texttt{ as } \exists\alpha.\, \Delta^1(\tau), \texttt{pack } (\sigma', V') \texttt{ as } \exists\alpha.\, \Delta^2(\tau), \exists\alpha.\, \tau) \in \mathcal{R}$, then $(\Delta \cup \{\alpha \mapsto (\sigma, \sigma')\}, \mathcal{R} \cup \{(V, V', \tau)\}, s, s') \in (X^\star)^\rightarrow$ for some α, or else $(\Delta, \mathcal{R} \cup \{(V, V', [\rho/\alpha]\tau)\}, s, s') \in (X^\star)^\rightarrow$ for some ρ with $FTV(\rho) \subseteq dom(\Delta)$, $\sigma = \Delta^1(\rho)$ and $\sigma' = \Delta^2(\rho)$.
  (d) If $(\langle V_1, \ldots, V_n \rangle, \langle V_1', \ldots, V_n' \rangle, \tau_1 \times \cdots \times \tau_n) \in \mathcal{R}$, then $(\Delta, \mathcal{R} \cup \{(V_i, V_i')\}, s, s') \in (X^\star)^\rightarrow$ for any $i \in \{1, ..., n\}$.
  (e) $(\Delta, \mathcal{R} \cup \{(\ell, \ell', \tau \; \texttt{ref})\}, s \uplus \{\ell \mapsto W\}, s' \uplus \{\ell' \mapsto W'\}) \in (X^\star)^\rightarrow$ for any $\ell \notin dom(s)$, $\ell' \notin dom(s')$ and $(W, W', \tau) \in (\Delta, \mathcal{R})^\star$.
  (f) If $(\ell, \ell', \tau \; \texttt{ref}) \in \mathcal{R}$, then
    i. $(\Delta, \mathcal{R} \cup \{(s(\ell), s'(\ell'), \tau)\}, s, s') \in (X^\star)^\rightarrow$.
    ii. $(\Delta, \mathcal{R}, s\{\ell \mapsto W\}, s'\{\ell' \mapsto W'\}) \in (X^\star)^\rightarrow$ for any $(W, W', \tau) \in (\Delta, \mathcal{R})^\star$.
  (g) If $(\ell, \ell_1', \tau \; \texttt{ref}) \in \mathcal{R}$ and $(\ell, \ell_2', \tau \; \texttt{ref}) \in \mathcal{R}$, then $\ell_1' = \ell_2'$.

It is also possible to consider just $X^\star$ in place of $(X^\star)^\rightarrow$. We here consider the latter because we often want to use up-to reduction and up-to context at the same time.

Lemma 4 (soundness of up-to reduction and context). *Suppose X is an environmental simulation up-to reduction and context. Then $X^\star$ is an environmental simulation up-to reduction (so $(X^\star)^\rightarrow$ is an environmental simulation).*

Proof. We check each condition of Definition 8 for each element of $X^\star$ by expanding Definition 9. Details are in Appendix B.

The last one is specific to calculi with generative names (like our locations).

Definition 11. *The* allocation closure X^ν *of X is defined as*

$$\begin{aligned} X^\nu = &\{(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \mid (\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \in X\} \\ &\cup \{(\Delta, \mathcal{R}, s, s') \mid (\Delta, \mathcal{S}, t, t') \in X, \quad (\mathcal{R}, s, s') \in (\Delta, \mathcal{S}, t, t')^\nu\} \end{aligned}$$

where

$$\begin{aligned} (\Delta, \mathcal{S}, t, t')^\nu = \{\ (\mathcal{R}, s, s') \mid\ & \mathcal{R} = \mathcal{S} \cup \{(\overline{\ell}, \overline{\ell}', \overline{\tau}\ \texttt{ref})\}, \quad (\overline{V}, \overline{V}', \overline{\tau}) \in (\Delta, \mathcal{R})^\star, \\ & s = t \uplus \{\overline{\ell} \mapsto \overline{V}\}, \quad s' = t' \uplus \{\overline{\ell}' \mapsto \overline{V}'\}\ \}. \end{aligned}$$

Informally, X^ν is an extention of X with extra locations $\overline{\ell}$ and $\overline{\ell}'$ allocated (and initialized with $\overline{V}$ and $\overline{V}'$) by the context. (This extention is limited only to elements of the form $(\Delta, \mathcal{R}, s, s')$ in the definition above. A similar extention is also possible for $(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau)$, but is less useful because M and M' often contain the extended locations $\overline{\ell}$ and $\overline{\ell}'$. See, e.g., Example 2 and 3.)

This time, the definition of up-to can *almost* be obtained by replacing positive X with $((X^\nu)^\star)^\rightarrow$ in Definition 6. However, Condition 2a, 2b and 2(f)ii require adjustments, as underlined below. Such adjustments were unnecessary in up-to reduction (and context) roughly because reduction (and context) closure does not essentially increase $(\Delta, \mathcal{R}, s, s') \in X$. This is not the case in allocation closure. Note also that the underlined conditions are still necessary for soundness, e.g., when the observed terms are functions that take n locations as arguments and return `true` if and only if these locations are pairwise different.

Definition 12. *X is an* environmental simulation up-to reduction, context, and allocation *(or just an* environmental simulation up-to *in short) if:*

1. *For any $(\Delta, \mathcal{R}, s \rhd M, s' \rhd M', \tau) \in X$,*
   (a) *If $s \rhd M \rightarrow t \rhd N$, then $s' \rhd M' \twoheadrightarrow t' \rhd N'$ for some t' and N' with $(\Delta, \mathcal{R}, t \rhd N, t' \rhd N', \tau) \in ((X^\nu)^\star)^\rightarrow$.*
   (b) *If $M = V$, then $s' \rhd M' \twoheadrightarrow t' \rhd V'$ for some t' and V' with $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, s, t') \in ((X^\nu)^\star)^\rightarrow$.*
2. *For any $(\Delta, \mathcal{R}, s, s') \in X$,*
   (a) *If $(\lambda x \colon \Delta^1(\tau_1).\, M,\ \lambda x \colon \Delta^2(\tau_1).\, M',\ \tau_1 \rightarrow \tau_2) \in \mathcal{R}$, then $(\Delta,\ \mathcal{S},\ t \rhd [W/x]M,\ t' \rhd [W'/x]M',\ \tau_2) \in ((X^\nu)^\star)^\rightarrow$ $\underline{\text{for any } (\mathcal{S}, t, t') \in (\Delta, \mathcal{R}, s, s')^\nu}$ and $(W, W', \tau_1) \in (\Delta, \mathcal{S})^\star$.*
   (b) *If $(\Lambda\alpha.\, M,\ \Lambda\alpha.\, M',\ \forall\alpha.\tau) \in \mathcal{R}$, then $(\Delta,\ \mathcal{S},\ t \rhd [\Delta^1(\sigma)/\alpha]M,\ t' \rhd [\Delta^2(\sigma)/\alpha]M',\ [\sigma/\alpha]\tau) \in ((X^\nu)^\star)^\rightarrow$ $\underline{\text{for any } (\mathcal{S}, t, t') \in (\Delta, \mathcal{R}, s, s')^\nu}$ and σ with $FTV(\sigma) \subseteq dom(\Delta)$.*

(c) If $(\texttt{pack}\ (\sigma, V)\ \texttt{as}\ \exists\alpha.\, \Delta^1(\tau),\ \texttt{pack}\ (\sigma', V')\ \texttt{as}\ \exists\alpha.\, \Delta^2(\tau),\ \exists\alpha.\,\tau) \in \mathcal{R}$*, then* $(\Delta \cup \{\alpha \mapsto (\sigma, \sigma')\}, \mathcal{R} \cup \{(V, V', \tau)\}, s, s') \in ((X^\nu)^\star)^\rightarrow$ *for some* α*, or else* $(\Delta, \mathcal{R} \cup \{(V, V', [\rho/\alpha]\tau)\}, s, s') \in ((X^\nu)^\star)^\rightarrow$ *for some* ρ *with* $FTV(\rho) \subseteq dom(\Delta)$*,* $\sigma = \Delta^1(\rho)$ *and* $\sigma' = \Delta^2(\rho)$*.*

(d) If $(\langle V_1, \ldots, V_n\rangle, \langle V_1', \ldots, V_n'\rangle, \tau_1 \times \cdots \times \tau_n) \in \mathcal{R}$*, then* $(\Delta, \mathcal{R} \cup \{(V_i, V_i')\}, s, s') \in ((X^\nu)^\star)^\rightarrow$ *for any* $i \in \{1, ..., n\}$*.*

(e) No condition required (item left only for the sake of consistent numbering).

(f) If $(\ell, \ell', \tau\ \texttt{ref}) \in \mathcal{R}$*, then*

i. $(\Delta, \mathcal{R} \cup \{(s(\ell), s'(\ell'), \tau)\}, s, s') \in ((X^\nu)^\star)^\rightarrow$*.*

ii. $(\Delta, \mathcal{S}, t\{\ell \mapsto W\}, t'\{\ell' \mapsto W'\}) \in ((X^\nu)^\star)^\rightarrow$ $\underline{\textit{for any } (\mathcal{S}, t, t') \in (\Delta, \mathcal{R}, s, s')^\nu}$ *and* $(W, W', \tau) \in (\Delta, \mathcal{S})^\star$*.*

(g) If $(\ell, \ell_1', \tau\ \texttt{ref}) \in \mathcal{R}$ *and* $(\ell, \ell_2', \tau\ \texttt{ref}) \in \mathcal{R}$*, then* $\ell_1' = \ell_2'$*.*

Lemma 5 (soundness of up-to reduction, context, and allocation). *Suppose* X *is an environmental simulation up-to reduction, context, and allocation. Then* X^ν *is an environmental simulation up-to reduction and context (so* $((X^\nu)^\star)^\rightarrow$ *is an environmental simulation).*

Proof. We check each condition of Definition 10 for each element of X^ν by expanding Definition 11. Details in Appendix C.

6 The characterization theorem

We now prove that environmental bisimilarity coincides with a form of observational equivalence. Let $\equiv$ be the largest environmental relation such that $\equiv^\star \subseteq \equiv$ and for any $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in \equiv$, $s \triangleright M$ converges if and only if $s' \triangleright M'$ does. It exists because the union of all such environmental relations trivially satisfies the same property (X appears only once in a negative position in each clause of Definition 9).

The relation $\equiv$ corresponds to the conventional definition of contextual equivalence in the following way. Take two closed values V and V' of type τ. If $(\emptyset, \{(V, V', \tau)\}, \emptyset, \emptyset) \in \equiv$, then $(\emptyset, \emptyset, \emptyset \triangleright [V/x]C, \emptyset \triangleright [V'/x]C) \in \equiv^\star$ by Definition 9 for any well-typed C. Therefore, $[V/x]C$ and $[V'/x]C$ coterminate by the definition above. Conversely, if V and V' coterminate under arbitrary (well-typed) contexts, then $\{(\emptyset, \{(V, V', \tau)\}, \emptyset, \emptyset)\}^\star$ satisfies the above property. Hence $(\emptyset, \{(V, V', \tau)\}, \emptyset, \emptyset) \in \equiv$.

Although the argument above assumed closed values, it is also straightforward to treat open terms N and N', by taking $V = \lambda\overline{x}.\, N$ and $V' = \lambda\overline{x}.\, N'$ for $\{\overline{x}\} \supseteq FV(N) \cup FV(N')$ as in previous work [26, Section 6][11, Appendix A.2].

Lemma 6 (soundness of environmental bisimulation). *Environmental bisimilarity* $\sim$ *is included in* $\equiv$*.*

Proof. Let $\preceq$ be the environmental similarity. By Lemma 4, $(\preceq^\star)^\rightarrow$ is an environmental simulation and therefore $\preceq^\star \subseteq (\preceq^\star)^\rightarrow \subseteq \preceq$. By symmetry, $\succeq^\star \supseteq \succeq$

(where $\succeq$ denotes $\preceq^{-1}$). Hence $\sim^\star = (\preceq \cap \succeq)^\star \subseteq (\preceq^\star \cap \succeq^\star) \subseteq (\preceq \cap \succeq) = \sim$. Also, by Condition 1a and 1b of Definition 6, for any $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in \sim$, $s \triangleright M$ converges if and only if $s' \triangleright M'$ does. Hence $\sim \subseteq \equiv$.

Lemma 7 (completeness). $\equiv$ *is an environmental bisimulation.*

Proof. By checking each condition of Definition 6 for each element of $\equiv$, expanding Definition 9. Again, details are found in Appendix D.

Theorem 1 (characterization). *Environmental bisimilarity* $\sim$ *equals observational equivalence* $\equiv$*.*

7 Examples

Below, we present examples of equivalence proofs by our environmental bisimulation. In each example, we prove the equivalence of the first two terms by constructing an environmental bisimulation X up-to reduction, context, and allocation. More examples are given in Appendix E.

Example 1 (abstract counters with and without bounds checking [3, Section 5.1]). Let

$$\begin{array}{rlrl} \mathit{mkCnt} &= (\texttt{let}\ x = \texttt{ref}\ 0\ \texttt{in}\ \mathit{cnt}_x) & \mathit{mkCnt}' &= (\texttt{let}\ x = \texttt{ref}\ 0\ \texttt{in}\ \mathit{cnt}'_x) \\ \mathit{cnt}_x &= \texttt{pack}\ (\texttt{int}, \langle \mathit{incr}_x, \mathit{chk}_x \rangle)\ \texttt{as}\ \tau & \mathit{cnt}'_x &= \texttt{pack}\ (\texttt{int}, \langle \mathit{incr}_x, \mathit{chk}' \rangle)\ \texttt{as}\ \tau \\ \mathit{chk}_x &= \lambda z.\, z \le !\, x & \mathit{chk}' &= \lambda z.\, \texttt{true} \\ \mathit{incr}_x &= \lambda \_.\ {+}{+}\, x & \tau &= \exists \alpha.\, (\mathbf{1} \to \alpha) \times (\alpha \to \texttt{bool}) \end{array}$$

with the standard syntactic sugar below.

$$\begin{array}{rlrl} {+}{+}\, x &= (x := !\, x + 1; !\, x) & M; N &= \texttt{let}\ \_ = M\ \texttt{in}\ N \\ \texttt{let}\ x = M\ \texttt{in}\ N &= (\lambda x.\, N) M & \lambda \_.\, M &= \lambda x.\, M \quad x \notin FV(M) \end{array}$$

Then mkCnt and mkCnt' are equivalent because of the following X.

$$\begin{array}{l} X = \{(\emptyset, \emptyset, \emptyset \triangleright \mathit{mkCnt}, \emptyset \triangleright \mathit{mkCnt}', \tau)\} \\ \quad \cup \{(\Delta, \mathcal{R}, s, s') \mid \\ \qquad \mathcal{R} = \{(\mathit{incr}_\ell, \mathit{incr}_{\ell'}, \mathbf{1} \to \alpha), (\mathit{chk}_\ell, \mathit{chk}', \alpha \to \texttt{bool}), (1, 1, \alpha), \dots, (n, n, \alpha)\}, \\ \qquad \Delta = \{\alpha \mapsto (\texttt{int}, \texttt{int})\},\ s = \{\ell \mapsto n\},\ s' = \{\ell' \mapsto n\},\ n \in \{0, 1, 2, \dots\}\} \end{array}$$

Let us check that X is indeed an environmental bisimulation up-to reduction, context, and allocation. Most conditions hold just by construction. The only cases that need to be checked are Condition 2a for the first and second elements of the above $\mathcal{R}$. Both of them are straightforward, given the fact that $(W, W', \mathbf{1}) \in (\Delta, \mathcal{R})^\star$ implies $W = W' = \langle\rangle$ and $(W, W', \alpha) \in (\Delta, \mathcal{R})^\star$ implies $W = W' \in \{1, 2, \dots, n\}$ (immediate from Definition 5).

Example 2 (irreversible state change [3, Section 5.5], or the "awkward" example [19, Example 5.9]). Below is an example that can be proved by recent work [3] but cannot by classic one [19]. This example uses no existential types, but is still interesting because of local state. (It can easily be turned into an equivalence involving packages, like Example 3.)

$$M = (\texttt{let}\ x = \texttt{ref}\ 0\ \texttt{in}\ V_x) \qquad M' = V'$$

$$V_x = \lambda f.\, x := 1; f\langle\rangle; !\,x \qquad\qquad V' = \lambda f.\, f\langle\rangle; 1 \qquad \tau = (\mathbf{1} \to \mathbf{1}) \to \mathtt{int}$$

for which we take

$$\begin{aligned}
X = \{&(\emptyset, \emptyset, \emptyset \rhd M, \emptyset \rhd M', \tau)\} \\
&\cup \{(\emptyset, \mathcal{R}, s, \emptyset) \mid \mathcal{R} = \{(V_\ell, V', \tau)\},\ \ s = \{\ell \mapsto i\},\ \ i \in \{0,1\}\} \\
&\cup \{(\emptyset, \mathcal{R}, s \uplus t \rhd N, t' \rhd N', \mathtt{int}) \mid \\
&\qquad \mathcal{R} = \{(V_\ell, V', \tau), (\overline{k}, \overline{k}', \overline{\rho}\ \mathtt{ref})\},\ \ s = \{\ell \mapsto 1\}, \\
&\qquad dom(t) = \{\overline{k}\},\ \ dom(t') = \{\overline{k}'\},\ \ (t(\overline{k}), t'(\overline{k}'), \overline{\rho}) \in (\emptyset, \mathcal{R})^\star, \\
&\qquad (N, N', \mathtt{int}) \in \{(E_1[...[E_n[C; !\,\ell]; !\,\ell]...]; !\,\ell,\ E_1[...[E_n[C; 1]; 1]...]; 1)\}^{(\emptyset,\mathcal{R})}\}.
\end{aligned}$$

Here, $\mathcal{T}^{(\Delta,\mathcal{R})}$ denotes closure under contexts in $\mathcal{T}$, i.e.,

$$\begin{aligned}
\mathcal{T}^{(\Delta,\mathcal{R})} = \{([\overline{V}/\overline{x}]\Delta^1(C),\ [\overline{V}'/\overline{x}]\Delta^2(C'),\ \tau) \mid (C, C') \in \mathcal{T},\ \ (\overline{V}, \overline{V}', \overline{\tau}) \in \mathcal{R}, \\
dom(\Delta), \overline{x} : \overline{\tau} \vdash C : \tau,\ \ dom(\Delta), \overline{x} : \overline{\tau} \vdash C' : \tau\}.
\end{aligned}$$

The irreversible change of state is represented by the requirement $s = \{\ell \mapsto 1\}$ in the third subset of X above. Stores t and t' account for locations $\overline{k}$ and $\overline{k}'$ (and their contents) created by the contexts. The most important technique here is the inclusion of all contexts of the form $E_1[\ldots[E_n[C; !\,\ell]; !\,\ell]\ldots]; !\,\ell$ and $E_1[\ldots[E_n[C; 1]; 1]\ldots]; 1$, representing nested calls to V_ℓ and V'. Then, the only non-trivial case to prove is Condition 1a for N and N', which follows from Lemma 2.

Example 3 (well-bracketed state change [3, Section 5.7], credited to Jacob Thamsborg). This is an existential variant of a negative example in [3] (i.e., they could not prove it).

$$\begin{aligned}
M &= \mathtt{pack}\ (\mathtt{int\ ref}, \langle \mathtt{ref}\ 1, \lambda x.\, V_x\rangle)\ \mathtt{as}\ \sigma & M' &= \mathtt{pack}\ (\mathbf{1}, \langle\langle\rangle, \lambda\_.\, V'\rangle)\ \mathtt{as}\ \sigma \\
V_x &= \lambda f.\, (x := 0; f\langle\rangle; x := 1; f\langle\rangle; !\,x) & V' &= \lambda f.\, (f\langle\rangle; f\langle\rangle; 1) \\
\sigma &= \exists\alpha.\, \alpha \times (\alpha \to \tau) & \tau &= (\mathbf{1} \to \mathbf{1}) \to \mathtt{int}
\end{aligned}$$

The difficulty of this example lies in how to represent the fact that the mutations $x := 0$ and $x := 1$ are "well-bracketed," i.e., whenever x is mutated to 0, it will eventually be restored to 1.

Consider first the context's observations on M. By opening and projection, the context learns some location ℓ under an abstract type α (with store $\{\ell \mapsto 1\}$) and the function $\lambda x.\, V_x$ of type $\alpha \to \tau$. By applying the latter to the former, it then learns function V_ℓ. This function can be applied to some $f = \lambda\_.\, [\overline{V}/\overline{x}]C$ (where $\overline{V}$ are taken from the context's knowledge), yielding a term N_1 of the form $[\overline{V}/\overline{x}]C_1; \ell := 1; [\overline{V}/\overline{x}]D_1; !\,\ell$ with store $\{\ell \mapsto 0\}$. By Lemma 2, any non-value of the form $[\overline{V}/\overline{x}]C_1$ either reduces to another term of the same form, or else "uses" some V_i. In the former case, the form of N_1 does not change. In the latter case, suppose $V_i = V_\ell$ and $N_1 = E[V_\ell W]$ for some E and W, that is, N_1 makes a nested call to V_ℓ (otherwise, the form of N_1 does not change, either). Then N_1 reduces to a term of the form $E[[\overline{V}/\overline{x}]C_2; \ell := 1; [\overline{V}/\overline{x}]D_2; !\,\ell]$ and the above arguments can be repeated for the subterm $[\overline{V}/\overline{x}]C_2; \ell := 1; [\overline{V}/\overline{x}]D_2; !\,\ell$. (Similar arguments apply to V' as well.) To enumerate all such terms that are possible under the store $\{\ell \mapsto 0\}$, we define a binary relation $\mathcal{T}_\ell^0$ on contexts by induction, using free variable v as a hole to substitute V_ℓ (or V').

- $(C; \ell := 1; D; !\,\ell)\ \mathcal{T}^0_\ell\ (C; D; 1)$
- If $E[vW]\ \mathcal{T}^0_\ell\ E'[vW]$, then $E[C; \ell := 1; D; !\,\ell]\ \mathcal{T}^0_\ell\ E'[C; D; 1]$

On the other hand, if $[\overline{V}/\overline{x}]C$ converges to value $\langle\rangle$, then $[\overline{V}/\overline{x}]C; \ell{:=}1; [\overline{V}/\overline{x}]D; !\,\ell$ reduces to $[\overline{V}/\overline{x}]D; !\,\ell$ with store $\{\ell \mapsto 1\}$. Similarly, $E[[\overline{V}/\overline{x}]C; \ell{:=}1; [\overline{V}/\overline{x}]D; !\,\ell]$ reduces to $E[[\overline{V}/\overline{x}]D; !\,\ell]$. We therefore define another binary relation $\mathcal{T}^1_\ell$ on contexts to enumerate possible terms under the store $\{\ell \mapsto 1\}$.

- $(D; !\,\ell)\ \mathcal{T}^1_\ell\ (D; 1)$
- If $E[vW]\ \mathcal{T}^0_\ell\ E'[vW]$, then $E[D; !\,\ell]\ \mathcal{T}^1_\ell\ E'[D; 1]$

Again by Lemma 2, any non-value of the form $[\overline{V}/\overline{x}]D$ either reduces to the same form or makes a nested call to V_ℓ. Hence the additional rules:

- If $E[vW]\ \mathcal{T}^1_\ell\ E'[vW]$, then $E[C; \ell := 1; D; !\,\ell]\ \mathcal{T}^0_\ell\ E'[C; D; 1]$
- If $E[vW]\ \mathcal{T}^1_\ell\ E'[vW]$, then $E[D; !\,\ell]\ \mathcal{T}^1_\ell\ E'[D; 1]$

This concludes the definition of $\mathcal{T}^0_\ell$ and $\mathcal{T}^1_\ell$. The following lemmas—proved by simple case analysis on the derivations of $E[vW]\ \mathcal{T}^0_\ell\ E'[vW]$ and $E[vW]\ \mathcal{T}^1_\ell\ E'[vW]$—are used when $[\overline{V}/\overline{x}]D$ converges to value $\langle\rangle$ and therefore $E[[\overline{V}/\overline{x}]D; !\,\ell]$ reduces to $E[1]$.

- If $E[vW]\ \mathcal{T}^1_\ell\ E'[vW]$, then $E[1]\ \mathcal{T}^1_\ell\ E'[1]$.
- If $E[vW]\ \mathcal{T}^0_\ell\ E'[vW]$, then $E[1]\ \mathcal{T}^1_\ell\ E'[1]$.

Then, take:

$$\begin{aligned}
X = {} & \{(\emptyset, \emptyset, \emptyset \rhd M, \emptyset \rhd M', \sigma)\} \\
& \cup \{(\Delta, \mathcal{R}, s, \emptyset) \mid \\
& \qquad s = \{\ell \mapsto 1\},\quad \Delta = \{\alpha \mapsto (\texttt{int ref}, \mathbf{1})\}, \\
& \qquad \mathcal{R} = \{(\ell, \langle\rangle, \alpha),\ (\lambda x.\, V_x, \lambda\_.\, V', \alpha \to \tau),\ (V_\ell, V', \tau)\}\} \\
& \cup \{(\Delta,\ \mathcal{R},\ s \uplus t \rhd N,\ t' \rhd N',\ \texttt{int}) \mid \\
& \qquad (N, N', \texttt{int}) \in \mathcal{T}^{0(\Delta,\mathcal{R})}_\ell,\quad s = \{\ell \mapsto 0\},\quad \Delta = \{\alpha \mapsto (\texttt{int ref}, \mathbf{1})\}, \\
& \qquad \mathcal{R} = \{(\ell, \langle\rangle, \alpha),\ (\lambda x.\, V_x, \lambda\_.\, V', \alpha \to \tau),\ (V_\ell, V', \tau),\ (\overline{k}, \overline{k}', \overline{\rho}\ \texttt{ref})\} \\
& \qquad dom(t) = \{\overline{k}\},\quad dom(t') = \{\overline{k}'\},\quad (t(\overline{k}), t'(\overline{k}'), \overline{\rho}) \in (\Delta, \mathcal{R})^\star\} \\
& \cup \{(\Delta,\ \mathcal{R},\ s \uplus t \rhd N,\ t' \rhd N',\ \texttt{int}) \mid \\
& \qquad (N, N', \texttt{int}) \in \mathcal{T}^{1(\Delta,\mathcal{R})}_\ell,\quad s = \{\ell \mapsto 1\},\quad \Delta = \{\alpha \mapsto (\texttt{int ref}, \mathbf{1})\}, \\
& \qquad \mathcal{R} = \{(\ell, \langle\rangle, \alpha),\ (\lambda x.\, V_x, \lambda\_.\, V', \alpha \to \tau),\ (V_\ell, V', \tau),\ (\overline{k}, \overline{k}', \overline{\rho}\ \texttt{ref})\} \\
& \qquad dom(t) = \{\overline{k}\},\quad dom(t') = \{\overline{k}'\},\quad (t(\overline{k}), t'(\overline{k}'), \overline{\rho}) \in (\Delta, \mathcal{R})^\star\}
\end{aligned}$$

Thanks to the construction of $\mathcal{T}^0_\ell$ and $\mathcal{T}^1_\ell$ with the lemmas above, the proof that X is a bisimulation up-to is routine, using Lemma 2 for reduction of the terms N and N'.

As one can see in some of the examples, our method has the ability to prove equivalences that depend on sequentiality. In fact, part of our reasoning—namely, the proof of Condition 1a by Lemma 2—resembles a classic syntactic disproof against the definability of parallel-or in PCF [16, p. 117]. Note, however, that the standard equivalence between

$$\begin{aligned} \mathit{isPor} = \lambda f.\, & \texttt{if}\ \neg f\langle \lambda\_.\,\texttt{true}, \lambda\_.\,\bot\rangle\ \texttt{then}\ \bot\ \texttt{else} \\ & \texttt{if}\ \neg f\langle \lambda\_.\,\bot, \lambda\_.\,\texttt{true}\rangle\ \texttt{then}\ \bot\ \texttt{else} \\ & \texttt{if}\ f\langle \lambda\_.\,\texttt{false}, \lambda\_.\,\texttt{false}\rangle\ \texttt{then}\ \bot\ \texttt{else}\ \langle\rangle \end{aligned}$$

and $\lambda f.\,\bot$ does *not* hold in our language because of state (for instance, f can count the number of calls to itself). This becomes apparent if one tries to construct a bisimulation proof, where the difference between states obstructs the use of Lemma 2. More specifically, for $x : \mathbf{1} \to \texttt{bool}, y : \mathbf{1} \to \texttt{bool} \vdash C : \texttt{bool}$,

$$s_0 \triangleright [\lambda\_.\,\texttt{true}, \lambda\_.\,\bot / x, y]C \twoheadrightarrow s_1 \triangleright \texttt{true}$$

and

$$s_1 \triangleright [\lambda\_.\,\bot, \lambda\_.\,\texttt{true} / x, y]C \twoheadrightarrow s_2 \triangleright \texttt{true}$$

do *not* imply

$$s_2 \triangleright [\lambda\_.\,\texttt{false}, \lambda\_.\,\texttt{false} / x, y]C \twoheadrightarrow s_3 \triangleright \texttt{true}$$

because of the differences among s_0, s_1 and s_2. Without state, our method could prove the equivalence between *isPor* and $\lambda f.\,\bot$.

8 Conclusion

We have presented the first complete, purely syntactic ("semantics-free," as opposed to syntax-free) proof technique for observational equivalences in polymorphic λ-calculus with full references, with non-trivial examples that could not be proved previously. Although we omitted explicit recursion, recursive functions can be encoded [17, Exercise 13.5.8]. Treatment of recursive types (either equi-recursive or iso-recursive) is also straightforward (see, e.g., [26]). Deallocation and pointer arithmetic (by defining stores as partial maps from locations to *arrays* of values) can also be added without essential difficulty, though deallocation introduces non-determinism [24] and invalidates the up-to reduction technique in general (but "up-to *deterministic* reduction" is still possible).

Of course, the above facts do *not* mean other approaches are useless. On the contrary, the inclusion of an infinite number of contexts in examples with callbacks suggests that, at least for some special cases, more convenient techniques (like [3]) can be devised to reduce the "size" of the set to be constructed by the user. (On the other hand, those examples have also shown that, with the help of Lemma 2, our "brute-force" method is often simple enough.) Logical relations are also better at giving a compositional model of universal types, as in [3] and [6, 7].

As we have shown in recent work [24], our approach is applicable to more general properties other than observational equivalence, such as memory safety and space improvement. It would also be possible to adapt them to our typed setting. Contrary to the previous (too negative) conjecture [26, Section 8], it *is* possible as well to use our method to prove free theorems à la Wadler [28] based on parametricity [20]. Work is ongoing on this topic.

Acknowledgements. Derek Dreyer first referred me to his work with Ahmed and Rossberg [3], from which I took examples in the present paper. Some of the anonymous reviewers found (and even corrected) a number of serious typos in a previous version of this paper.

References

[1] S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 334–344, 1998.

[2] A. Ahmed, A. W. Appel, and R. Virga. An indexed model of impredicative polymorphism and mutable references. `http://www.cs.princeton.edu/~amal/papers/impred.pdf`, 2003.

[3] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 340–353, 2009.

[4] A. W. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Transactions on Programming Languages and Systems*, 23(5):657–683, 2001.

[5] A. Banerjee and D. A. Naumann. State based ownership, reentrance, and encapsulation. In *ECOOP 2005 – Object-Oriented Programming*, volume 3586 of *Lecture Notes in Computer Science*, pages 387–411. Springer-Verlag, 2005.

[6] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, references, and recursive types. In *Foundations of Software Science and Computation Structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 456–470. Springer-Verlag, 2009.

[7] L. Birkedal, K. Støvring, and J. Thamsborg. Relational parametricity for references and recursive types. In *Types in Language Design and Implementation*, pages 91–104, 2009.

[8] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *Programming Languages and Systems*, volume 4279 of *Lecture Notes in Computer Science*, pages 79–96. Springer-Verlag, 2006.

[9] M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *14th Annual IEEE Symposium on Logic in Computer Science*, pages 214–224, 1999.

[10] A. D. Gordon. *Functional Programming and Input/Output.* PhD thesis, University of Cambridge, 1993.

[11] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 141–152, 2006.

[12] J. Laird. A fully abstract trace semantics for general references. In *Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 667–679. Springer-Verlag, 2007.

[13] S. B. Lassen and P. B. Levy. Typed normal form bisimulation for parametric polymorphism. In *23rd Annual IEEE Symposium on Logic in Computer Science*, pages 341–352, 2008.

[14] A. R. Meyer and K. Sieber. Towards fully abstract semantics for local variables: Preliminary report. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 191–203, 1988.

[15] J. C. Mitchell. On the equivalence of data representations. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 305–330. Academic Press, 1991.

[16] J. C. Mitchell. *Foundations for Programming Languages.* MIT Press, 1996.

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

[18] B. C. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *Journal of the ACM*, 47(3):531–586, 2000. Extended abstract appeared in *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1997, pp. 531–584.

[19] A. M. Pitts and I. Stark. Operational reasoning for functions with local state. In *Higher Order Operational Techniques in Semantics*, pages 227–273. Cambridge University Press, 1998.

[20] J. C. Reynolds. Types, abstraction and parametric polymorphism. In *Information Processing 83, Proceedings of the IFIP 9th World Computer Congres*, pages 513–523, 1983.

[21] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *Twenty-Second Annual IEEE Symposium on Logic in Computer Science*, pages 293–302, 2007.

[22] D. Sangiorgi and R. Milner. The problem of "weak bisimulation up to". In *CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1992.

[23] K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 161–172, 2007.

[24] E. Sumii. A theory of non-monotone memory (or: Contexts for `free`). In *18th European Symposium on Programming*, volume 5502 of *Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 2009.

[25] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1–3):169–192, 2007. Extended abstract appeared in *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 161–172, 2004.

[26] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM*, 54(5-26):1–43, 2007. Extended abstract appeared in *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 63–74, 2005.

[27] N. Tzevelekos. Full abstraction for nominal general references. In *Twenty-Second Annual IEEE Symposium on Logic in Computer Science*, pages 399–410, 2007.

[28] P. Wadler. Theorems for free! In *Proceedings of the Fourth ACM SIGPLAN International Conference on Functional Programming Languages and Computer Architecture*, pages 347–359. ACM, 1989.

Appendices

A Proof of Lemma 3

Case $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X^{\rightarrow}$ where $s \triangleright M \twoheadrightarrow t \triangleright N$ and $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$ with $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in \pi^1(X)$.

Condition 1a. Suppose $s \triangleright M \rightarrow t_0 \triangleright N_0$. We prove $(\Delta, \mathcal{R}, t_0 \triangleright N_0, t' \triangleright N', \tau) \in X^{\rightarrow}$. Since $s \triangleright M \twoheadrightarrow t \triangleright N$, we have $t_0 \triangleright N_0 \twoheadrightarrow \xi(t \triangleright N)$ for some permutation ξ on $Loc \setminus s$ by Lemma 1. Since $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in \pi^1(X)$, we have $(\Delta, \mathcal{R}, \xi(t \triangleright N), t' \triangleright N', \tau) \in (\xi \circ \pi)^1(X))$. Hence $(\Delta, \mathcal{R}, t_0 \triangleright N_0, t' \triangleright N', \tau) \in X^{\rightarrow}$ by Definition 7.

Condition 1b. Suppose $M = V$ and therefore $s \triangleright M = t \triangleright N$. We prove $s' \triangleright M' \twoheadrightarrow t'_0 \triangleright V'$ with $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, t, t'_0) \in X^{\rightarrow}$. This is immediate from $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$ and Condition 1b of Definition 8 for $(\Delta, (\pi^{-1})^1(\mathcal{R}), \pi^{-1}(t \triangleright N), t' \triangleright N', \tau) \in X$.

Case $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X^{\rightarrow}$ where $s \triangleright M$ diverges. We prove Condition 1a of Definition 6, i.e., $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in X^{\rightarrow}$ if $s \triangleright M \rightarrow t \triangleright N$. This is immediate from Definition 7 because $t \triangleright N$ also diverges by Lemma 1.

Case $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X^{\rightarrow}$ where $s \triangleright M \twoheadrightarrow t \triangleright V$ and $s' \triangleright M' \twoheadrightarrow t' \triangleright V'$ with $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, t, t') \in \pi^1(X)$.

Condition 1a. Suppose $s \triangleright M \rightarrow t_0 \triangleright N$. We prove $(\Delta, \mathcal{R}, t_0 \triangleright N, t' \triangleright V', \tau) \in X^{\rightarrow}$. Since $s \triangleright M \twoheadrightarrow t \triangleright V$, we have $t_0 \triangleright N \twoheadrightarrow \xi(t \triangleright V)$ for some permutation ξ on $Loc \setminus s$ by Lemma 1. Since $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, t, t') \in \pi^1(X)$, we have $(\Delta, \xi^1(\mathcal{R} \cup \{(V, V', \tau)\}), \xi(t), t') \in (\xi \circ \pi)^1(X)$. Hence $(\Delta, \mathcal{R}, t_0 \triangleright N, t' \triangleright V', \tau) \in X^{\rightarrow}$ by Definition 7.

Condition 1b. Suppose $M = V$ and therefore $s \triangleright M = t \triangleright V$. Trivial because $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, t, t') \in \pi^1(X) \subseteq X^{\rightarrow}$.

Case $(\Delta, \mathcal{R}, s, s') \in X^{\rightarrow}$ where $(\Delta, \mathcal{R}, s, s') \in \pi^1(X)$. We have only to check Conditions 2a–2g of Definition 6, which are immediate from the corresponding conditions of Definition 8 for $(\Delta, (\pi^{-1})^1(\mathcal{R}), \pi^{-1}(s), s') \in X$.

B Proof of Lemma 4

Case $(\Delta, \mathcal{R}, s \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)[M], s' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[M'], \tau) \in X^\star$ where $\Delta \subseteq \Delta_0$, $\mathcal{R} \subseteq (\Delta_0, \mathcal{S})^\star$ with $FTV(\mathcal{R}) \subseteq dom(\Delta)$, and $(\overline{V}, \overline{V}', \overline{\tau}) \in \mathcal{S}$ for $(\Delta_0, \mathcal{S}, s \triangleright M, s' \triangleright M', \tau_0) \in X$ and $dom(\Delta_0), \overline{y} : \overline{\tau} \vdash E[\tau_0] : \tau$ with $FTV(\tau) \subseteq dom(\Delta)$. We proceed by case analysis on whether M reduces or is a value.

Sub-case $s \triangleright M \rightarrow t \triangleright N$ and therefore $s \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)[M] \rightarrow t \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)[N]$. We prove Condition 1a of Definition 8, i.e., $s' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[M'] \twoheadrightarrow t' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[N']$ for $(\Delta, \mathcal{R}, t \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)[N], t' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[N'], \tau) \in (X^\star)^{\rightarrow}$.

Since $(\Delta_0, \mathcal{S}, s \triangleright M, s' \triangleright M', \tau_0) \in X$, we have $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$ with $(\Delta_0, \mathcal{S}, t \triangleright N, t' \triangleright N', \tau_0) \in (X^\star)^{\rightarrow}$ by Condition 1a of Definition 10. Since $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$, we have $s' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[M'] \twoheadrightarrow s' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[N']$. Since $(\Delta_0, \mathcal{S}, t \triangleright N, t' \triangleright N', \tau_0) \in (X^\star)^{\rightarrow}$, we also have $(\Delta, \mathcal{R}, t \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)N, t' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)N', \tau) \in (X^\star)^{\rightarrow}$ (simple calculation with Definition 7 and 9).

Sub-case $M = V$. Since $(\Delta_0, \mathcal{S}, s \triangleright M, s' \triangleright M', \tau_0) \in X$, we have $s' \triangleright M' \twoheadrightarrow t' \triangleright V'$ with $(\Delta_0, \mathcal{S} \cup \{(V, V', \tau_0)\}, s, t') \in (X^\star)^\rightarrow$ by Condition 1b of Definition 10. Since $s' \triangleright M' \twoheadrightarrow t' \triangleright V'$, we have $s' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[M'] \twoheadrightarrow t' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[V']$. Since $(\Delta_0, \mathcal{S} \cup \{(V, V', \tau_0)\}, s, t') \in (X^\star)^\rightarrow$, we have $(\Delta, (\pi^{-1})^1(\mathcal{R}), \pi^{-1}(s \triangleright [\overline{V}/\overline{y}]\Delta_0^1(E)[V]), t' \triangleright [\overline{V}'/\overline{y}]\Delta_0^2(E)[V'], \tau) \in X^\star$ (simple calculation with Definition 7 and 9). The rest of the proof amounts to the next case.

Case $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X^\star$ where $\Delta \subseteq \Delta_0$, $\mathcal{R} \subseteq (\Delta_0, \mathcal{S})^\star$ with $FTV(\mathcal{R}) \subseteq dom(\Delta)$, and $(M, M', \tau) \in (\Delta_0, \mathcal{S})^\star$ with $FTV(\tau) \subseteq dom(\Delta)$ for $(\Delta_0, \mathcal{S}, s, s') \in X$.

If $M = V$, then $M' = V'$ (immediate from Definition 5 for $(M, M', \tau) \in (\Delta_0, \mathcal{S})^\star$). We have only to check Condition 1b of Definition 8, i.e., $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, s, s') \in (X^\star)^\rightarrow$, which is immediate from Definition 9 because $(V, V', \tau) \in (\Delta_0, \mathcal{S})^\star$ and $FTV(\tau) \subseteq dom(\Delta)$.

Otherwise, $s \triangleright M \rightarrow t \triangleright N$. We prove Condition 1a of Definition 8, i.e., $s' \triangleright M' \twoheadrightarrow t' \triangleright N'$ and $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in (X^\star)^\rightarrow$. Expanding Definition 5 for $(M, M', \tau) \in (\Delta_0, \mathcal{S})^\star$, let $M = [\overline{V}/\overline{x}]\Delta_0^1(C)$ and $M' = [\overline{V}'/\overline{x}]\Delta_0^2(C)$ with $dom(\Delta_0), \overline{x} : \overline{\tau} \vdash C : \tau$ and $(\overline{V}, \overline{V}', \overline{\tau}) \in \mathcal{S}$. We proceed by case analysis on the form of C.

Sub-case C is not of the form $E[\texttt{ref}\ V]$ and no $x_i \in \{\overline{x}\}$ is at the destruction position in C. Since $s \triangleright M \rightarrow t \triangleright N$, we have $N = [\overline{V}/\overline{x}]\Delta_0^1(D)$ with $t = s$, and $s' \triangleright M' \rightarrow t' \triangleright N'$ for $N' = [\overline{V}'/\overline{x}]\Delta_0^1(D)$ with $t' = s'$ by Lemma 2. Hence $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in X^\star \subseteq (X^\star)^\rightarrow$ by Definition 9.

Sub-case $C = E[\texttt{ref}\ V]$ with $dom(\Delta_0), \overline{x} : \overline{\tau} \vdash V : \sigma$ and $dom(\Delta_0), \overline{x} : \overline{\tau} \vdash E[\sigma\ \texttt{ref}] : \tau$. Then $N = [\overline{V}/\overline{x}]\Delta_0^1(E)[\ell]$ and $t = s \uplus \{\ell \mapsto W\}$ for $W = [\overline{V}/\overline{x}]\Delta_0^1(V)$, and $s' \triangleright M' \rightarrow t' \triangleright N'$ with $N' = [\overline{V}'/\overline{x}]\Delta_0^2(E)[\ell']$ and $t' = s' \uplus \{\ell' \mapsto W'\}$ for $W' = [\overline{V}'/\overline{x}]\Delta_0^1(V)$. Since $(\Delta_0, \mathcal{S}, s, s') \in X$, we have $(\Delta_0, \mathcal{S} \cup \{(\ell, \ell', \sigma)\}, t, t') \in (X^\star)^\rightarrow$ by Condition 2e of Definition 10. Hence $(\Delta, \mathcal{R}, t \triangleright N, t' \triangleright N', \tau) \in (X^\star)^\rightarrow$ (simple calculation with Definition 7 and 9).

Sub-case x_i is at the destruction position in C. We expand Definition 1. (For brevity, we show two important cases only. The other cases are similar and simpler.)

Sub-sub-case $C = E[x_i V]$ where $\tau_i = \tau_{i1} \rightarrow \tau_{i2}$ with $dom(\Delta_0), \overline{x} : \overline{\tau} \vdash V : \tau_{i1}$ and $dom(\Delta_0), \overline{x} : \overline{\tau} \vdash E[\tau_{i2}] : \tau$. Let $V_i = \lambda x : \Delta_0^1(\tau_{i1}).\, M_0$ and $V_i' = \lambda x : \Delta_0^2(\tau_{i1}).\, M_0'$. Since $(\Delta_0, \mathcal{S}, s, s') \in X$, we have $(\Delta_0, \mathcal{S}, s \triangleright [W/x]M_0, s' \triangleright [W'/x]M_0', \tau_{i2}) \in (X^\star)^\rightarrow$ for $W = [\overline{V}/\overline{x}]\Delta_0^1(V)$ and $W' = [\overline{V}'/\overline{x}]\Delta_0^2(V)$ by Condition 2a of Definition 10. Hence $(\Delta, \mathcal{R}, s \triangleright [\overline{V}/\overline{x}]\Delta_0^1(E)[[W/x]M_0], s' \triangleright [\overline{V}'/\overline{x}]\Delta_0^2(E)[[W'/x]M_0'], \tau) \in (X^\star)^\rightarrow$ (simple calculation with Definition 7 and 9).

Sub-sub-case $C = E[\texttt{open}\ x_i\ \texttt{as}\ (\alpha, x)\ \texttt{in}\ D]$ where $\tau_i = \exists\alpha.\, \tau_{i0}$ with $dom(\Delta_0), \alpha, \overline{x} : \overline{\tau}, x : \tau_{i0} \vdash D : \tau_0$ and $dom(\Delta_0), \overline{x} : \overline{\tau} \vdash E[\tau_0] : \tau$. Let $V_i = \texttt{pack}\ (\sigma, V_{i0})\ \texttt{as}\ \tau_i$ and $V_i' = \texttt{pack}\ (\sigma', V_{i0}')\ \texttt{as}\ \tau_i$. Since $(\Delta_0, \mathcal{S}, s, s') \in X$, we have

$(\varDelta_0 \cup \{\alpha \mapsto (\sigma, \sigma')\}, \mathcal{S} \cup \{(V_{i0}, V'_{i0}, \tau_{i0})\}, s, s') \in (X^\star)^\rightarrow$ by Condition 2c of Definition 10. Hence $(\varDelta, \mathcal{R}, s \triangleright E[[\sigma, V_{i0}/\alpha, x]D], s' \triangleright E[[\sigma', V'_{i0}/\alpha, x]D], \tau) \in (X^\star)^\rightarrow$ (simple calculation with Definition 7 and 9).

C Proof of Lemma 5

Case $(\varDelta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X^\nu$ where $(\varDelta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in X$. Trivial.

Case $(\varDelta, \mathcal{R}, s, s') \in X^\nu$ where $(\mathcal{R}, s, s') \in (\varDelta, \mathcal{S}, t, t')^\nu$ with $(\varDelta, \mathcal{S}, t, t') \in X$. We check Condition 2a–2g of Definition 10. Again, we only show important cases.

Condition 2a. Let $(\lambda x : \varDelta^1(\tau_1).\, M, \lambda x : \varDelta^2(\tau_1).\, M', \tau_1 \rightarrow \tau_2) \in \mathcal{R}$ and $(W, W', \tau_1) \in (\varDelta, \mathcal{R})^\star$. We prove $(\varDelta, \mathcal{R}, s \triangleright [W/x]M, s' \triangleright [W'/x]M', \tau_2) \in ((X^\nu)^\star)^\rightarrow$. This is identical to Condition 2a of Definition 12 for $(\varDelta, \mathcal{S}, t, t') \in X$.

Condition 2e. Let $\ell \notin \mathit{dom}(s)$ and $\ell' \notin \mathit{dom}(s')$ with $(W, W', \tau) \in (\varDelta, \mathcal{R})^\star$. We prove $(\varDelta, \mathcal{R} \cup \{(\ell, \ell', \tau\ \mathtt{ref})\}, s \uplus \{\ell \mapsto W\}, s' \uplus \{\ell' \mapsto W'\}) \in ((X^\nu)^\star)^\rightarrow$. This is immediate from Definition 11 because $(\varDelta, \mathcal{S}, t, t') \in X$.

Now, expanding Definition 11 for $(\mathcal{R}, s, s') \in (\varDelta, \mathcal{S}, t, t')^\nu$, let $s = t \uplus \{\overline{\ell} \mapsto \overline{V}\}$ and $s' = t' \uplus \{\overline{\ell}' \mapsto \overline{V}'\}$ with $\mathcal{R} = \mathcal{S} \cup \{(\overline{\ell}, \overline{\ell}', \overline{\tau}\ \mathtt{ref})\}$ for $(\overline{V}, \overline{V}', \overline{\tau}) \in (\varDelta, \mathcal{R})^\star$.

Condition 2(f)i. Let $(\ell, \ell', \tau\ \mathtt{ref}) \in \mathcal{R}$. We prove $(\varDelta, \mathcal{R} \cup \{(s(\ell), s'(\ell'), \tau)\}, s, s') \in ((X^\nu)^\star)^\rightarrow$.

Sub-case $(\ell, \ell', \tau\ \mathtt{ref}) \in \mathcal{S}$. Since $(\varDelta, \mathcal{S}, t, t') \in X$, we have $(\varDelta, \mathcal{S} \cup \{(t(\ell), t'(\ell'), \tau)\}, t, t') \in ((X^\nu)^\star)^\rightarrow$ by Condition 2(f)i of Definition 12. Hence $(\varDelta, \mathcal{R} \cup \{(s(\ell), s'(\ell'), \tau)\}, s, s') \in ((X^\nu)^\star)^\rightarrow$ (simple calculation with Definition 7, 9 and 11).

Sub-case $\ell = \ell_i$ and $\ell' = \ell'_i$ with $\tau = \tau_i$. Since $(\varDelta, \mathcal{R}, s, s') \in X^\nu$ and $(V_i, V'_i, \tau_i) \in (\varDelta, \mathcal{R})^\star$, we have $(\varDelta, \mathcal{R} \cup \{(V_i, V'_i, \tau_i)\}, s, s') \in (X^\nu)^\star$ by Definition 9.

Condition 2(f)ii. Let $(\ell, \ell', \tau\ \mathtt{ref}) \in \mathcal{R}$ and $(W, W', \tau) \in (\varDelta, \mathcal{R})^\star$. We prove $(\varDelta, \mathcal{R}, s\{\ell \mapsto W\}, s'\{\ell' \mapsto W'\}) \in ((X^\nu)^\star)^\rightarrow$.

Sub-case $(\ell, \ell', \tau\ \mathtt{ref}) \in \mathcal{S}$. Since $(\varDelta, \mathcal{S}, t, t') \in X$, we have $(\varDelta, \mathcal{R}, s\{\ell \mapsto W\}, s'\{\ell' \mapsto W'\}) \in ((X^\nu)^\star)^\rightarrow$ by Condition 2(f)ii of Definition 12.

Sub-case $\ell = \ell_i$ and $\ell' = \ell'_i$ with $\tau = \tau_i$. Since $(W, W', \tau) \in (\varDelta, \mathcal{R})^\star$, we have $(\mathcal{R}, s\{\ell_i \mapsto W\}, s'\{\ell'_i \mapsto W'\}) \in (\varDelta, \mathcal{S}, t, t')^\nu$ by (the latter half of) Definition 11. Then, since $(\varDelta, \mathcal{S}, t, t') \in X$, we have $(\varDelta, \mathcal{R}, s\{\ell_i \mapsto W\}, s'\{\ell'_i \mapsto W'\}) \in (X^\nu)^\star$ by (the former half of) Definition 11.

Condition 2g. Let $(\ell, \ell'_1, \tau\ \mathtt{ref}) \in \mathcal{R}$ and $(\ell, \ell'_2, \tau\ \mathtt{ref}) \in \mathcal{R}$. We prove $\ell'_1 = \ell'_2$.

Sub-case $\ell \in \mathit{dom}(t)$. Since $\ell \notin \{\overline{\ell}\}$, we have $(\ell, \ell'_1, \tau\ \mathtt{ref}) \in \mathcal{S}$ and $(\ell, \ell'_2, \tau\ \mathtt{ref}) \in \mathcal{S}$. Since $(\varDelta, \mathcal{S}, t, t') \in X$, we have $\ell'_1 = \ell'_2$ by Condition 2g of Definition 12.

Sub-case $\ell = \ell_i$. Trivial.

D Proof of Lemma 7

Case $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in \equiv$.

Condition 1a. Suppose $s \triangleright M \to t \triangleright N$. We prove $(\Delta, \mathcal{R}, t \triangleright N, s' \triangleright M', \tau) \in \equiv$, i.e., $t \triangleright [\overline{V}/\overline{y}]\Delta^1_0(E)[N]$ converges if and only if $s' \triangleright [\overline{V}'/\overline{y}]\Delta^2_0(E)[M']$ does for any E etc. (with the same assumptions as in the first part of Definition 9). Suppose $t \triangleright [\overline{V}/\overline{y}]\Delta^1_0(E)[N]$ converges. Then so does $s \triangleright [\overline{V}/\overline{y}]\Delta^1_0(E)[M]$ and therefore $s' \triangleright [\overline{V}'/\overline{y}]\Delta^2_0(E)[M']$ also converges. Conversely, suppose $s' \triangleright [\overline{V}'/\overline{y}]\Delta^2_0(E)[M']$ converges. Then so does $s \triangleright [\overline{V}/\overline{y}]\Delta^1_0(E)[M]$ and therefore $t \triangleright [\overline{V}/\overline{y}]\Delta^1_0(E)[N]$ also converges by Lemma 1.

Condition 1b. Suppose $M = V$. We prove $s' \triangleright M' \twoheadrightarrow t' \triangleright V'$ and $(\Delta, \mathcal{R} \cup \{(V, V', \tau)\}, s, t') \in \equiv$, i.e., $s \triangleright [\overline{V}, V/\overline{y}, y]\Delta^1_0(C)$ converges if and only if $t' \triangleright [\overline{V}', V'/\overline{y}, y]\Delta^2_0(C)$ does (again with due assumptions). The former is immediate because $s \triangleright M$ converges and therefore so does $s' \triangleright M'$. The latter follows from the cotermination for $(\Delta, \mathcal{R}, s \triangleright M, s' \triangleright M', \tau) \in \equiv$ under evaluation context $\texttt{let } y = [\,] \texttt{ in } C$.

Case $(\Delta, \mathcal{R}, s, s') \in \equiv$. We spell out important cases only.

Condition 2a. Let $(\lambda x : \Delta^1(\tau_1).\, M, \lambda x : \Delta^2(\tau_1).\, M', \tau_1 \to \tau_2) \in \mathcal{R}$. We prove $(\Delta, \mathcal{R}, s \triangleright [W/x]M, s' \triangleright [W'/x]M', \tau_2) \in \equiv$ for any $(W, W', \tau_1) \in (\Delta, \mathcal{R})^\star$. That is, $s \triangleright [\overline{V}/\overline{y}]\Delta^1_0(E)[[C/x]M]$ converges if and only if $s' \triangleright [\overline{V}'/\overline{y}]\Delta^2_0(E)[[C/x]M']$ does. This follows from the cotermination for $(\Delta, \mathcal{R}, s, s') \in \equiv$ under context $E[zC]$.

Condition 2c. Let $(\texttt{pack } (\sigma, V) \texttt{ as } \exists\alpha.\, \Delta^1(\tau), \texttt{pack } (\sigma', V') \texttt{ as } \exists\alpha.\, \Delta^2(\tau), \exists\alpha.\, \tau) \in \mathcal{R}$. We prove $(\Delta \cup \{\alpha \mapsto (\sigma, \sigma')\}, \mathcal{R} \cup \{(V, V', \tau)\}, s, s') \in \equiv$. That is, $s \triangleright [\overline{V}, V, \sigma/\overline{y}, y, \alpha]\Delta^1_0(C)$ converges if and only if $s' \triangleright [\overline{V}', V', \sigma'/\overline{y}, y, \alpha]\Delta^2_0(C)$ does. This follows from the cotermination for $(\Delta, \mathcal{R}, s, s') \in \equiv$ under context $\texttt{open } z \texttt{ as } (\alpha, y) \texttt{ in } C$.

Condition 2e. We prove $(\Delta, \mathcal{R} \cup \{(\ell, \ell', \tau \texttt{ ref})\}, s \uplus \{\ell \mapsto W\}, s' \uplus \{\ell' \mapsto W'\}) \in \equiv$ for any $\ell \notin \mathit{dom}(s)$, $\ell' \notin \mathit{dom}(s')$ and $(W, W', \tau) \in (\Delta, \mathcal{R})^\star$. That is, $s \uplus \{\ell \mapsto [\overline{V}/\overline{y}]\Delta^1_0(D)\} \triangleright [\overline{V}, \ell/\overline{y}, y]\Delta^1_0(C)$ converges if and only if $s' \uplus \{\ell' \mapsto [\overline{V}'/\overline{y}]\Delta^2_0(D)\} \triangleright [\overline{V}', \ell'/\overline{y}, y]\Delta^2_0(C)$ does. This follows from the cotermination for $(\Delta, \mathcal{R}, s, s') \in \equiv$ under context $\texttt{let } y = \texttt{ref } D \texttt{ in } C$.

E More examples

Example 4 (twin abstraction with integers and references as generative names [3, Section 5.2 and 5.3]). Let

$$
\begin{aligned}
\mathit{mkGenPkg} &= \mathit{genPkg} \\
\mathit{mkGenPkg}' &= (\texttt{let } x = \texttt{ref } 0 \texttt{ in } \mathit{genPkg}'_x) \\
\mathit{genPkg} &= \texttt{pack } (\mathbf{1}\ \texttt{ref}, \mathbf{1}\ \texttt{ref}, \langle \mathit{gen}, \mathit{gen}, \mathit{cmp} \rangle) \texttt{ as } \tau \\
\mathit{genPkg}'_x &= \texttt{pack } (\texttt{int}, \texttt{int}, \langle \mathit{gen}'_x, \mathit{gen}'_x, \mathit{cmp}' \rangle) \texttt{ as } \tau \\
\mathit{gen} &= \lambda\_.\, \texttt{ref } \langle\rangle
\end{aligned}
$$

$$\begin{aligned}
\mathit{gen}'_x &= \lambda\_.\ {+}{+}\,x \\
\mathit{cmp} &= \lambda p.\ \#_1(p) \overset{\mathit{ptr}}{=} \#_2(p)\ ?\ \texttt{true} : \texttt{false} \\
\mathit{cmp}' &= \lambda p.\ \#_1(p) \overset{\mathit{int}}{=} \#_2(p) \\
\tau &= \exists\alpha,\beta.\ (\mathbf{1} \to \alpha) \times (\mathbf{1} \to \beta) \times (\alpha \times \beta \to \texttt{bool})
\end{aligned}$$

with

$$\begin{aligned}
X = {} & \{(\emptyset, \emptyset, \emptyset \rhd \mathit{mkGenPkg}, \emptyset \rhd \mathit{mkGenPkg}', \tau)\} \\
& \cup \{(\Delta, \mathcal{R}, s, s') \mid \\
& \quad \Delta = \{\alpha \mapsto (\mathbf{1}\ \texttt{ref}, \texttt{int})\} \uplus \{\beta \mapsto (\mathbf{1}\ \texttt{ref}, \texttt{int})\}, \\
& \quad \mathcal{R} = \{(\mathit{genPkg}, \mathit{genPkg}', \tau), \\
& \qquad\quad (\mathit{gen}, \mathit{gen}'_k, \mathbf{1} \to \alpha), \\
& \qquad\quad (\mathit{gen}, \mathit{gen}'_k, \mathbf{1} \to \beta), \\
& \qquad\quad (\mathit{cmp}, \mathit{cmp}', \alpha \times \beta \to \texttt{bool}), \\
& \qquad\quad (\overline{\ell}, \overline{i}, \alpha), (\overline{m}, \overline{j}, \beta)\}, \\
& \quad s = \{\overline{\ell} \mapsto \langle\rangle\} \uplus \{\overline{m} \mapsto \langle\rangle\}, \quad s' = \{k \mapsto n\}, \\
& \quad \{\overline{i}\} \cap \{\overline{j}\} = \emptyset, \quad \overline{i}, \overline{j} \le n\}.
\end{aligned}$$

Then X is an environmental bisimulation up-to reduction, context, and allocation. Checking this is just as easy as in Example 1. Note in particular that the comparison functions *cmp* and *cmp'* always give `false`. That is, $t \rhd \Delta^1(\mathit{cmp})(W) \twoheadrightarrow t \rhd \texttt{false}$ and $t' \rhd \Delta^2(\mathit{cmp}')(W') \twoheadrightarrow t' \rhd \texttt{false}$ for any $(W, W', \alpha \times \beta) \in (\Delta, \mathcal{S})^\star$ with $(t, t', \mathcal{S}) \in (\Delta, \mathcal{R}, s, s')^\nu$.

Example 5 (generic cell classes [3, Section 5.4] [11, Section 6.3]). Let

$$\begin{aligned}
\mathit{genCell} &= \Lambda\alpha.\ \texttt{pack}\ (\mathit{Cell}_\alpha, \langle \mathit{new}_\alpha, \mathit{read}_\alpha, \mathit{write}_\alpha\rangle)\ \texttt{as}\ \tau_\alpha \\
\mathit{genCell}' &= \Lambda\alpha.\ \texttt{pack}\ (\mathit{Cell}'_\alpha, \langle \mathit{new}'_\alpha, \mathit{read}'_\alpha, \mathit{write}'_\alpha\rangle)\ \texttt{as}\ \tau_\alpha \\
\mathit{Cell}_\alpha &= \alpha\ \texttt{ref} \\
\mathit{Cell}'_\alpha &= \texttt{bool ref} \times \alpha\ \texttt{ref} \times \alpha\ \texttt{ref} \\
\mathit{new}_\alpha &= \lambda x.\ \texttt{ref}\ x \\
\mathit{new}'_\alpha &= \lambda x.\ \langle \texttt{ref true}, \texttt{ref}\ x, \texttt{ref}\ x\rangle \\
\mathit{read}_\alpha &= \lambda \mathit{cell}.\ !\,\mathit{cell} \\
\mathit{read}'_\alpha &= \lambda\langle \mathit{switch}, \mathit{cell1}, \mathit{cell2}\rangle. \\
&\qquad \texttt{if}\ !\,\mathit{switch}\ \texttt{then}\ !\,\mathit{cell1}\ \texttt{else}\ !\,\mathit{cell2} \\
\mathit{write}_\alpha &= \lambda\langle \mathit{cell}, x\rangle.\ \mathit{cell} := x \\
\mathit{write}'_\alpha &= \lambda\langle\langle \mathit{switch}, \mathit{cell1}, \mathit{cell2}\rangle, x\rangle. \\
&\qquad \mathit{switch} := \neg\,!\,\mathit{switch}; \\
&\qquad \texttt{if}\ !\,\mathit{switch}\ \texttt{then}\ \mathit{cell1} := x\ \texttt{else}\ \mathit{cell2} := x \\
\tau_\alpha &= \exists\beta.\ (\alpha \to \beta) \times (\beta \to \alpha) \times (\beta \times \alpha \to \mathbf{1})
\end{aligned}$$

and then the following X is a bisimulation up-to.

$$\begin{aligned}
X = \{(\Delta, \mathcal{R}, s, s') \mid & \\
& \Delta = \{\overline{\beta} \mapsto (\mathit{Cell}_{\Delta^1(\overline{\rho})}, \mathit{Cell}'_{\Delta^2(\overline{\rho})})\}, \\
& \mathit{FTV}(\mathit{Cell}_i) \subseteq \{\beta_1, \ldots, \beta_{i-1}\} \text{ for all } i \in \{1, \ldots, n\}, \\
& \mathcal{R} = \{(\mathit{genCell}, \mathit{genCell}', \forall\alpha.\ \tau_\alpha), \\
& \qquad (\mathit{new}_{\mathit{Cell}_i}, \mathit{new}'_{\mathit{Cell}_i}, \mathit{Cell}_i \to \beta_i), \\
& \qquad (\mathit{read}_{\mathit{Cell}_i}, \mathit{read}'_{\mathit{Cell}_i}, \beta_i \to \rho_i),
\end{aligned}$$

$$
\begin{aligned}
&\quad\quad (\mathit{write}_{\mathit{Cell}_i}, \mathit{write}'_{\mathit{Cell}_i}, \beta_i \times \rho_i \to \mathbf{1}),\\
&\quad\quad (\overline{\ell}_i, \langle \overline{b}_i, \overline{\ell}'_i, \overline{m}'_i \rangle, \beta_i) \ \mid\\
&\quad\quad i \in \{1, ..., n\}\}\\
&\{(s(\ell_{ij}), s'(\ell'_{ij}), \mathit{Cell}_i) \ \mid \ b_{ij} = \mathtt{true}\} \subseteq (\varDelta, \mathcal{R})^\star,\\
&\{(s(\ell_{ij}), s'(m'_{ij}), \mathit{Cell}_i) \ \mid \ b_{ij} = \mathtt{false}\} \subseteq (\varDelta, \mathcal{R})^\star\}
\end{aligned}
$$

The only news here (with respect to previous examples) is the first element of $\mathcal{R}$, for which Condition 2b is checked.

Example 6 (callback with lock [3, Section 5.6][5]). We show the equivalence of two terms L_M and $L_{M'}$ at type $\sigma \times \tau$, where:

$$
\begin{aligned}
L_z &= (\mathtt{let}\ b, x = \mathtt{ref\ true}, \mathtt{ref}\ 0\ \mathtt{in}\ \langle U_x, W_{b,x,z} \rangle)\\
U_x &= \lambda\_.\ !x\\
W_{b,x,z} &= \lambda f.\ \mathtt{if}\ \ !b\ \mathtt{then}\ (b := \mathtt{false}; z; b := \mathtt{true})\\
M &= f\langle\rangle; x := !x + 1\\
M' &= \mathtt{let}\ y = !x\ \mathtt{in}\ f\langle\rangle; x := y + 1\\
\tau &= (\mathbf{1} \to \mathbf{1}) \to \mathbf{1}\\
\sigma &= \mathbf{1} \to \mathtt{int}
\end{aligned}
$$

Take:

$$
\begin{aligned}
X = \{&(\emptyset, \mathcal{R}, s, s') \ \mid\\
&\quad \mathcal{R} = \{(U_m, U_{m'}, \sigma), (W_{\ell,m,M}, W_{\ell',m',M'}, \tau),\\
&\quad s = \{\ell \mapsto \mathtt{true}\} \uplus \{m \mapsto n\},\\
&\quad s' = \{\ell' \mapsto \mathtt{true}\} \uplus \{m' \mapsto n\}\}\\
\cup\, \{&(\emptyset, \mathcal{R}, s \uplus t \rhd N, s' \uplus t' \rhd N', \mathbf{1}) \ \mid\\
&\quad \mathcal{R} = \{(U_m, U_{m'}, \sigma), (W_{\ell,m,M}, W_{\ell',m',M'}, \tau), (\overline{k}, \overline{k}', \overline{\rho}\ \mathtt{ref})\},\\
&\quad s = \{\ell \mapsto \mathtt{false}\} \uplus \{m \mapsto n\},\\
&\quad s' = \{\ell' \mapsto \mathtt{false}\} \uplus \{m' \mapsto n\},\\
&\quad \mathit{dom}(t) = \{\overline{k}\}, \quad \mathit{dom}(t') = \{\overline{k}'\},\\
&\quad (t(\overline{k}), t'(\overline{k}'), \overline{\rho}) \in (\emptyset, \mathcal{R})^\star,\\
&\quad (N, N', \mathbf{1}) \in \{(C; m := !m + 1; \ell := \mathtt{true},\\
&\quad\quad\quad\quad\quad\quad C; m' := n + 1; \ell' := \mathtt{true})\}^{(\emptyset,\mathcal{R})}\\
&\quad v : \sigma, w : \tau, \overline{z} : \overline{\rho}\ \mathtt{ref} \vdash C : \mathbf{1}\}
\end{aligned}
$$

As in Example 2, the inclusion of contexts C accounts for the body of the callback function f, with the help of Lemma 2 for proving Condition 1a for the reduction of N and N'. Note that (unlike in Example 2) nested calls to $W_{b,x,z}$ have no effect, thanks to the "lock" b. Other than these, everything is straightforward again. Note that step-indexed logical relations [3] require non-trivial use of step indices to prove this equivalence.

Example 7 (deferred divergence [3, Section 5.8], credited to Hongseok Yang). Like Example 3, this is an existential version of an example for which [3] could not give a proof. For brevity, we use syntactic sugar for pattern matching on tuples (with the obvious meaning).

$$
V = \lambda f.\ f(\mathtt{pack}\ (\mathbf{1}, \langle\langle\rangle, \lambda\_.\ W\rangle)\ \mathtt{as}\ \sigma)
$$

$$\begin{aligned}
V' &= \lambda f.\, \texttt{let}\ \langle x, y\rangle = \langle \texttt{ref}\ 0, \texttt{ref}\ 0\rangle\ \texttt{in} \\
&\qquad\qquad f(\texttt{pack}\ (\texttt{int ref} \times \texttt{int ref}, \langle\langle x, y\rangle, \lambda\langle x, y\rangle.\, W'_{x,y}\rangle)\ \texttt{as}\ \sigma); N'_{x,y} \\
N'_{x,y} &= \texttt{if}\ \,!y = 0\ \texttt{then}\ x := 1\ \texttt{else}\ \bot \\
W &= \lambda\_.\, \bot \qquad\qquad\qquad\qquad W'_{x,y} = \lambda\_.\, \texttt{if}\ \,!x = 0\ \texttt{then}\ y := 1\ \texttt{else}\ \bot \\
\sigma &= \exists\alpha.\, \alpha \times (\alpha \to \mathbf{1} \to \mathbf{1}) \qquad \tau = (\sigma \to \mathbf{1}) \to \mathbf{1}
\end{aligned}$$

The difficulty here is how to keep track of the fact that $N'_{x,y}$ is waiting to be executed after every call to f by V', even if f may make nested calls to V'. The following X represents this simply by nested contexts D'. (For readability, typings of the contexts are omitted.) The first half of X corresponds to states where W and $W'_{x,y}$ have not been called, while the second to states where they have already been called (and therefore the reductions diverge).

$$\begin{aligned}
X = \{(&\Delta,\ \mathcal{R},\ t \triangleright M,\ s' \uplus t' \triangleright M',\ \mathbf{1})\ \mid \\
&(M, M', \mathbf{1}) \in \{(D, D')\}^{(\Delta,\mathcal{R})}, \\
&D = E_1[E_2[\ldots[E_{p-1}[C]]]], \\
&D' = E_1[E_2[\ldots[E_{p-1}[C; N'_{\ell_p,m_p}]; N'_{\ell_{p-1},m_{p-1}}]\ldots]; N'_{\ell_1,m_1}], \\
&s'(\ell_1) = \cdots = s'(\ell_p) = 0,\ \ s'(\ell_{p+1}) = \cdots = s'(\ell_{p+q}) = 1, \\
&s'(m_1) = \cdots = s'(m_{p+q}) = 0, \\
&\Delta = \{\overline{\alpha} \mapsto (\mathbf{1}, \texttt{int ref} \times \texttt{int ref})\}, \\
&\mathcal{R} = \{(V, V', \tau),\ (\langle\rangle, \langle\overline{\ell}, \overline{m}\rangle, \overline{\alpha}),\ (\lambda\_.\, W, \lambda\langle x, y\rangle.\, W'_{x,y}, \overline{\alpha} \to \mathbf{1} \to \mathbf{1}), \\
&\qquad (W, W'_{\overline{\ell},\overline{m}}, \mathbf{1} \to \mathbf{1}),\ (\overline{k}, \overline{k}', \overline{\rho}\ \texttt{ref})\}, \\
&dom(t) = \{\overline{k}\},\ \ dom(t') = \{\overline{k}'\},\ \ (t(\overline{k}), t'(\overline{k}'), \overline{\rho}) \in (\Delta, \mathcal{R})^\star\} \\
\cup \{(&\Delta,\ \mathcal{R},\ t \triangleright M,\ s' \uplus t' \triangleright M',\ \mathbf{1})\ \mid \\
&(M, M', \mathbf{1}) \in \{(D, D')\}^{(\Delta,\mathcal{R})}, \\
&D = E[\bot], \\
&D' = E_1[E_2[\ldots[E_{p-1}[C; N'_{\ell_p,m_p}]; N'_{\ell_{p-1},m_{p-1}}]\ldots]; N'_{\ell_1,m_1}], \\
&s'(\ell_1) = \cdots = s'(\ell_p) = 0,\ \ s'(\ell_{p+1}) = \cdots = s'(\ell_{p+q}) = 1, \\
&s'(m_j) = 1 \text{ for some } j \in \{1, \ldots, p\}, \\
&\Delta = \{\overline{\alpha} \mapsto (\mathbf{1}, \texttt{int ref} \times \texttt{int ref})\}, \\
&\mathcal{R} = \{(V, V', \tau),\ (\langle\rangle, \langle\overline{\ell}, \overline{m}\rangle, \overline{\alpha}),\ (\lambda\_.\, W, \lambda\langle x, y\rangle.\, W'_{x,y}, \overline{\alpha} \to \mathbf{1} \to \mathbf{1}), \\
&\qquad (W, W'_{\overline{\ell},\overline{m}}, \mathbf{1} \to \mathbf{1}),\ (\overline{k}, \overline{k}', \overline{\rho}\ \texttt{ref})\}, \\
&dom(t) = \{\overline{k}\},\ \ dom(t') = \{\overline{k}'\},\ \ (t(\overline{k}), t'(\overline{k}'), \overline{\rho}) \in (\Delta, \mathcal{R})^\star\}
\end{aligned}$$

Once more, it is routine to check X to be a bisimulation up-to, because most of the conditions hold just by the construction of X, again using Lemma 2 for reduction of the terms M and M'.

$$\frac{S \uplus \{\alpha\}, \varGamma, \varSigma \vdash M : \tau}{S, \varGamma, \varSigma \vdash \varLambda\alpha.\, M : \forall\alpha.\, \tau} \qquad \frac{S, \varGamma, \varSigma \vdash M : \forall\alpha.\, \sigma \quad FTV(\tau) \subseteq S}{S, \varGamma, \varSigma \vdash M[\tau] : [\tau/\alpha]\sigma}$$

$$\frac{FTV(\tau) \subseteq S \quad S, \varGamma, \varSigma \vdash M : [\tau/\alpha]\sigma}{S, \varGamma, \varSigma \vdash \texttt{pack}\ (\tau, M)\ \texttt{as}\ \exists\alpha.\, \sigma : \exists\alpha.\, \sigma}$$

$$\frac{S, \varGamma, \varSigma \vdash M : \exists\alpha.\, \sigma \quad S \uplus \{\alpha\}, \varGamma \uplus \{x \mapsto \sigma\}, \varSigma \vdash N : \tau \quad \alpha \notin FTV(\tau)}{S, \varGamma, \varSigma \vdash \texttt{open}\ M\ \texttt{as}\ (\alpha, x)\ \texttt{in}\ N : \tau}$$

$$\frac{}{S, \varGamma, \varSigma \vdash \ell : \varSigma(\ell)} \qquad \frac{S, \varGamma, \varSigma \vdash M : \tau}{S, \varGamma, \varSigma \vdash \texttt{ref}\ M : \tau\ \texttt{ref}} \qquad \frac{S, \varGamma, \varSigma \vdash M : \tau\ \texttt{ref}}{S, \varGamma, \varSigma \vdash\ !\, M : \tau}$$

$$\frac{S, \varGamma, \varSigma \vdash M : \tau\ \texttt{ref} \quad S, \varGamma, \varSigma \vdash N : \tau}{S, \varGamma, \varSigma \vdash M := N : \mathbf{1}} \qquad \frac{\begin{array}{c} S, \varGamma, \varSigma \vdash M_1 : \sigma\ \texttt{ref} \quad S, \varGamma, \varSigma \vdash M_2 : \sigma\ \texttt{ref} \\ S, \varGamma, \varSigma \vdash N_1 : \tau \quad S, \varGamma, \varSigma \vdash M_2 : \tau \end{array}}{S, \varGamma, \varSigma \vdash M_1 \overset{ptr}{=} M_2\ ?\ N_1 : N_2 : \tau}$$

$$s \rhd (\varLambda\alpha.\, M)[\tau] \to s \rhd [\tau/\alpha]M$$

$$s \rhd \texttt{open}\ (\texttt{pack}\ (\tau, V)\ \texttt{as}\ \exists\alpha.\, \sigma)\ \texttt{as}\ (\alpha, x)\ \texttt{in}\ N \to s \rhd [\tau, V/\alpha, x]N$$

$$s \rhd \texttt{ref}\ V \to s \uplus \{\ell \mapsto V\} \rhd \ell \qquad s \rhd\ !\, \ell \to s \rhd s(\ell)$$

$$s \uplus \{\ell \mapsto V\} \rhd \ell := W \to s \uplus \{\ell \mapsto W\} \rhd \langle\rangle$$

$$s \rhd \ell \overset{ptr}{=} \ell\ ?\ N_1 : N_2 \to s \rhd N_1 \qquad s \rhd \ell_1 \overset{ptr}{=} \ell_2\ ?\ N_1 : N_2 \to s \rhd N_2 \quad (\ell_1 \neq \ell_2)$$

Fig. 2. Typing and Reduction Rules (Excerpt)