Compositional relational reasoning via operational game semantics

Guilhem Jaber
Université de Nantes, LS2N, Inria, France

Andrzej S. Murawski
University of Oxford, UK

***Abstract*—We show how to use operational game semantics as a guide to develop relational techniques for establishing contextual equivalences with respect to contexts drawn from a hierarchy of four call-by-value higher-order languages: with either general or ground-type references and with either call/cc or no control operator. In game semantics, differences between the contexts can be captured by the absence or presence of the O-visibility and O-bracketing conditions.**

The proposed technique, which we call Kripke normal-form bisimulations, combines insights from normal-form bisimulation and Kripke logical relations with game semantics. In particular, the role of the heap and the name history is abstracted away using Kripke-style world transition systems. The differences between the four kinds of contexts manifest themselves through simple local conditions that can be shown to correspond to O-visibility and O-bracketing, as applicable.

The technique is sound and complete by virtue of correspondence with operational game semantics. Moreover, it sheds a new light on other related developments, such as backtracking and private transitions in Kripke logical relations, which can be related to specific phenomena in game models.

I. Introduction

Contextual equivalence is a fundamental concept in programming language theory with wide applicability to program specification and verification. Attempting to characterise it through mathematical models has been a major theme in denotational semantics, known as the quest for full abstraction [1]. Game semantics [2], [3] has proved especially fruitful in that quest, providing fully abstract models for a broad spectrum of programming paradigms.

Game semantics is a mathematical theory that views computation as an exchange of moves between two players, called O and P, representing the context/environment (O) and the program (P) respectively. Accordingly, it models a program as a strategy for P. Such strategies can be assigned to programs in a compositional way by building categories of games and strategies. Alternatively, one can define them in an operational way, as traces of a carefully crafted labelled transition system (LTS) [4], [5], [6]. In this paper we rely on the latter approach, referred to as operational game semantics. In particular, we explain how to extract reasoning principles for contextual equivalence using the structure of the underlying transition system.

Our language of study will be HOSC, which is a call-by-value higher-order language with first-class continuations, the control operator $\mathrm{call/cc}$ and general references. We will also consider its sublanguages GOSC, HOS and GOS, obtained respectively by restricting storage to ground values, by removing $\mathrm{call/cc}$, and by imposing both restrictions. In the paper, we study contextual testing of a class of HOSC terms using contexts from each of the languages $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$. Our working notion of testing will be convergence to error, represented by a free variable. Accordingly, at the technical level, we will work with four equivalence relations, written $\cong^{\mathbf{x}}_{err}$.

The differences between $\mathbf{x}$ correspond to varying access of contexts to general references and control. Game semantics can capture the resultant variations in the discriminating power by imposing restrictions on what moves O is allowed to make during play: the absence of general references in contexts ($\mathbf{x} = \mathrm{GOSC}, \mathrm{GOS}$) corresponds to a condition called O-visibility, while the absence of control operators ($\mathbf{x} = \mathrm{HOS}, \mathrm{GOS}$) corresponds to O-bracketing. In recent work [7], it was shown how to capture $\cong^{\mathbf{x}}_{err}$ in the spirit of operational game semantics: for each $\mathbf{x}$, one can define an LTS, called $\mathbf{x}[\mathrm{HOSC}]$, such that the induced notion of trace equivalence (strategy equality) coincides with $\cong^{\mathbf{x}}_{err}$ for terms of $\mathbf{x}$. Indeed, the traces of $\mathbf{x}[\mathrm{HOSC}]$ are those of $\mathrm{HOSC}[\mathrm{HOSC}]$ that conform to restrictions relevant to $\mathbf{x}$. Our starting point will be to present the four LTSs in a unified way by observing that the conditions boil down to different levels of access to function and continuation names introduced by P. The unifying LTS, called $\mathcal{L}_{\mathbf{x}}$, will thus feature a notion of history of available names, used to enforce restrictions on O-behavior. The differences between $\mathbf{x}$s can then be addressed purely locally by adjusting the way in which the two components are updated, the HOSC case being unrestricted. We then use $\mathcal{L}_{\mathbf{x}}$ as a guide to develop a relational technique for establishing trace equivalence. Because $\mathcal{L}_{\mathbf{x}}$ is deterministic, trace equivalence (and so contextual equivalence) corresponds to bisimilarity.

A key point of $\mathcal{L}_{\mathbf{x}}$ is the way higher-order values and evaluation contexts provided by the program to contexts are represented via function and continuation names respectively. In order to enable computation with such names, configurations contain an environment γ, which is a mapping from the relevant names to the corresponding higher-order values or evaluation contexts. We find similar notions of such environments in eager normal-form bisimulations [8], [9] and in environmental bisimulations [10], [11], [12]. Their presence is an obstacle to naive compositional reasoning, as shown by the

978-1-6654-4895-6/21/$31.00

following two ML-like programs:

`let x = ref 0 in`	`let y = ref 0 in`
`let inc f = f(); x++ in`	`let dec f = f(); y-- in`
`let get() =!x`	`let nget() = −!y`
`in (inc, get)`	`in (dec, nget)`

In order to prove the terms equivalent, we might want to prove the equivalence of `inc` with `dec` and that of `get` with `nget` separately. Unfortunately, as pointed out in [13], this would not be sound, since they share a common reference. That is why access to the environment γ is crucial: after `inc`/`dec` call f or return, the context may continue to use elements of γ. However, keeping it means that proving the equivalence in a componentwise fashion requires non-trivial measures.

The solution we propose to this problem is to abstract away the part that is shared by the components of γ, namely the heap and the name history, using Kripke-style world transition systems (WTSs) equipped with two accessibility relations ($\sqsubseteq_{\mathsf{OQ}}$, $\sqsubseteq_{\mathsf{OA}}$), which correspond to tracking the availability of function and continuation names respectively. Kripke-style reasoning about heaps has been explored in the setting of Kripke logical relations (KLR), where a notion of worlds as heap invariants was introduced [14], and later refined [15], [16] to take into account the possibility of invariant evolution.

Our proof principle will be presented as a family of parameterized operators $\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}$, $\mathcal{V}^{\mathbf{x}}_{\mathcal{A}}$, $\mathcal{K}^{\mathbf{x}}_{\mathcal{A}}$ (for expressions, values and continuations respectively) over a WTS $\mathcal{A}$. Kripke Normal-Form Bisimulations (KNFB) are then defined as post-fixpoints of such operators, in the style of work on normal-form bisimulations [8]. Equivalence proofs based on KNFBs make it possible to decompose the proof of the target equivalence into other equivalences according to the normal form of the underlying pure λ-term. The soundness of the approach relies on being able to compose the subproofs subject to compatibility rules expressed via the WTS. Completeness is obtained by lifting $\mathcal{L}_{\mathbf{x}}$ into a WTS.

Interestingly, we can develop the results simultaneously for all four variants of $\mathbf{x}$ and the differences between the cases correspond to minor adjustments of the way the $\sqsubseteq_{\mathsf{OQ}}$ and $\sqsubseteq_{\mathsf{OA}}$ relationships have to be maintained after reduction steps.

We call the technique Kripke Normal-Form Bisimulations (KNFB), because it combines the flavour of open/normal-form bisimulations [17], [8] with Kripke-style relational reasoning [16]. An important feature of KNFBs is that universal quantifications range over relatively simple first-order entities, which opens up the way to automated reasoning in each of the four cases, following [18]. By virtue of the correspondence with operational game semantics, the technique is sound and complete. In contrast to [16], we need not use step-indexing to model general references and we need not rely on bi-orthogonality to model call/cc. Also, our completeness results are proved without bi-orthogonality. This is a non-trivial difference, because bi-orthogonality is a generic completion technique, which may lead to completeness even if the corresponding "direct-style" reasoning principles are incomplete (e.g. the principle of local invariants from [14] is incomplete).

The authors of [16] discussed the impact of higher-order state and control effects on relational reasoning and proposed *backtracking* and *private transitions* as sound reasoning principles in the absence of general references and control respectively. Our work makes it possible to relate them to the O-visibility and O-bracketing conditions by observing that backtracking corresponds to the justification relation between moves and private transitions correspond to tracking continuation names. Overall, the connections indicate that it should be possible to state direct-style reasoning principles for KLR, removing the need to rely on bi-orthogonality for completeness. However, some additional book-keeping would have to be added to the structure of the proof, in line with how we track the history of worlds. We expand on this in Section VII.

Earlier work on marrying game semantics with operational techniques in the spirit of this paper concerned contextual testing of HOS terms inside HOS contexts [19]. Our framework covers all four languages in a uniform way. The associated soundness and completeness arguments are structured and provide abstract ways of translating KNFBs into bisimulations over $\mathcal{L}_{\mathbf{x}}$, and vice versa. In the future, we hope to capitalise on the generality of the approach and adapt it to further programming paradigms.

II. HOSC AND ITS FRAGMENTS

HOSC is a higher-order programming language equipped with general references and continuations. Its syntax is given in Figure 1. We will study HOSC along with its three sublanguages: GOSC, HOS and GOS.

Assuming countably infinite sets $\mathbf{Loc}$ (locations) and $\mathbf{Var}$ (variables), HOSC typing judgments take the form $\Sigma; \Gamma \vdash M : \tau$, where Σ and Γ are finite partial functions that assign types to locations and variables respectively. All the typing rules are standard and can be found in the full version of [7]. In typing judgements, we often write Σ as shorthand for $\Sigma; \emptyset$ (closed) and Γ as shorthand for $\emptyset; \Gamma$ (location-free). Similarly, $\vdash M : \tau$ means $\emptyset; \emptyset \vdash M : \tau$.

A heap h is a finite type-respecting map from $\mathbf{Loc}$ to values. We write $h : (\Sigma; \Gamma)$, if $\mathrm{dom}(\Sigma) \subseteq \mathrm{dom}(h)$ and $\Sigma; \Gamma \vdash h(\ell) : \sigma$ for $(\ell, \sigma) \in \Sigma$, The operational semantics of HOSC reduces pairs (M, h), where $\Sigma; \Gamma \vdash M : \tau$ and $h : (\Sigma; \Gamma)$. The rules are given in Figure 2, where $\{\cdot\}$ denotes (capture-avoiding) substitution. We write $(M, h) \Downarrow_{ter}$ if there exist V, h' such that $(M, h) \to^* (V, h')$ and V is a value.

We consider three fragments of HOSC. In GOSC, reference types are restricted to $\mathrm{ref}\,\iota$, where ι is given by $\iota \triangleq \mathrm{Unit} \mid \mathrm{Int} \mid \mathrm{Bool} \mid \mathrm{ref}\,\iota$. GOSC terms are HOSC terms whose typing derivations (i.e. not only the final typing judgments) rely on GOSC types only. GOSC is a superset of FOSC [16][1]. HOS types are HOSC types that do not contain the cont constructor. HOS terms are HOSC terms whose typing derivations rely on HOS types only. Consequently, HOS terms never have subterms of the form $\mathrm{call/cc}_{\tau}(x.M)$,

[1] GOSC also features references to references - the $\mathrm{ref}\,\iota$ case.

$$
\begin{array}{lcl}
\sigma,\tau & \triangleq & \mathrm{Unit} \mid \mathrm{Int} \mid \mathrm{Bool} \mid \mathrm{ref}\tau \mid \tau \times \sigma \mid \tau \to \sigma \mid \mathrm{cont}\ \tau \\
U,V & \triangleq & () \mid \mathbf{tt} \mid \mathbf{ff} \mid \widehat{n} \mid x \mid \ell \mid \langle U, V\rangle \mid \lambda x^{\tau}.M \mid \mathbf{rec}\ y(x^{\tau}).M \mid \mathrm{cont}_{\tau} K \\
M,N & \triangleq & V \mid \langle M, N\rangle \mid \pi_i M \mid MN \mid \mathrm{ref}_{\tau} M \mid !M \mid M := N \mid \mathrm{if}\, M_1 \,\mathrm{then}\, M_2 \,\mathrm{else}\, M_3 \mid M \oplus N \mid M \boxdot N \\
 & & \mid M = N \mid \mathrm{call/cc}_{\tau}(x.M) \mid \mathrm{throw}_{\tau}\ \mathrm{M}\ \mathrm{to}\ N \\
K & \triangleq & \bullet \mid \langle V, K\rangle \mid \langle K, M\rangle \mid \pi_i K \mid VK \mid KM \mid \mathrm{ref}_{\tau} K \mid !K \mid V := K \mid K := M \mid \mathrm{if}\, K \,\mathrm{then}\, M \,\mathrm{else}\, N \\
 & & \mid K \oplus M \mid V \oplus K \mid K \boxdot M \mid V \boxdot K \mid K = M \mid V = K \mid \mathrm{throw}_{\tau}\ \mathrm{V}\ \mathrm{to}\ K \mid \mathrm{throw}_{\tau}\ \mathrm{K}\ \mathrm{to}\ M \\
C & \triangleq & \bullet \mid \langle M, C\rangle \mid \langle C, M\rangle \mid \pi_i C \mid \lambda x^{\tau}.C \mid \mathbf{rec}\ y(x^{\tau}).C \mid MC \mid CM \mid \mathrm{ref}_{\tau} C \mid !C \\
 & & \mid C := M \mid M := C \mid \mathrm{if}\, C \,\mathrm{then}\, M \,\mathrm{else}\, N \mid \mathrm{if}\, M \,\mathrm{then}\, C \,\mathrm{else}\, N \mid \mathrm{if}\, M \,\mathrm{then}\, N \,\mathrm{else}\, C \mid C \oplus M \mid M \oplus C \\
 & & \mid C \boxdot M \mid M \boxdot C \mid C = M \mid M = C \mid \mathrm{call/cc}_{\tau}(x.C) \mid \mathrm{throw}_{\tau}\ \mathrm{C}\ \mathrm{to}\ M \mid \mathrm{throw}_{\tau}\ \mathrm{M}\ \mathrm{to}\ C
\end{array}
$$

Notational conventions: $x, y \in \mathbf{Var},\ \ell \in \mathbf{Loc},\ n \in \mathbb{Z},\ i \in \{1,2\},\ \oplus \in \{+,-,*\},\ \boxdot \in \{=,<\}$
Syntactic sugar: $\mathrm{let}\, x = M \,\mathrm{in}\, N$ stands for $(\lambda x.N)M$ (if x does not occur in N we also write $M; N$)

Fig. 1. HOSC syntax

$$
\begin{array}{lcl}
(K[(\lambda x^{\sigma}.M)V], h) & \to & (K[M\{V/x\}], h) \\
(K[\pi_i\langle V_1, V_2\rangle], h) & \to & (K[V_i], h) \\
(K[\mathrm{if}\ \mathbf{tt}\ \mathrm{then}\ M_1\ \mathrm{else}\ M_2], h) & \to & (K[M_1], h) \\
(K[\mathrm{if}\ \mathbf{ff}\ \mathrm{then}\ M_1\ \mathrm{else}\ M_2], h) & \to & (K[M_2], h) \\
(K[\widehat{n} \oplus \widehat{m}], h) & \to & (K[\widehat{n \oplus m}], h) \\
(K[\hat{n} \boxdot \hat{m}], h) & \to & (K[b], h) \\
\multicolumn{3}{l}{\text{with } b = \mathbf{tt} \text{ if } n \boxdot m, \text{ otherwise } b = \mathbf{ff}} \\
(K[\mathrm{call/cc}(x^{\tau}.M)], h) & \to & (K[M\{\mathrm{cont}_{\tau} K/x\}], h)
\end{array}
\qquad
\begin{array}{lcl}
(K[!\ell], h) & \to & (K[h(\ell)], h) \\
(K[\mathrm{ref}\, V], h) & \to & (K[\ell], h \cdot [\ell \mapsto V]) \\
(K[\ell := V], h) & \to & (K[()], h[\ell \mapsto V]) \\
(K[\ell = \ell'], h) & \to & (K[b], h) \\
\multicolumn{3}{l}{\text{with } b = \mathbf{tt} \text{ if } \ell = \ell', \text{otherwise } b = \mathbf{ff}} \\
(K[(\underbrace{\mathbf{rec}\ y(x^{\sigma}).M}_{U})V], h) & \to & (K[M\{V/x, U/y\}], h) \\
(K[\mathrm{throw}_{\tau}\ \mathrm{V}\ \mathrm{to}\ \mathrm{cont}_{\tau} K'], h) & \to & (K'[V], h)
\end{array}
$$

Fig. 2. Operational reduction for HOSC

$\mathrm{throw}_{\tau}\ \mathrm{M}\ \mathrm{to}\ N$ or $\mathrm{cont}_{\tau} K$. GOS is the intersection of HOS and GOSC, both for types and terms, i.e. there are no continuations and storage is restricted to values of type ι, defined above. The following definition allows us to study HOSC terms interacting with contexts in weaker fragments.

Definition 1. Given a HOSC term $\Gamma \vdash M : \tau$, we refer to types in Γ and τ as ***boundary types***. Let $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$. A HOSC term $\Gamma \vdash M : \tau$ has an $\mathbf{x}$ boundary if all of its boundary types are from $\mathbf{x}$.

Note that typing derivations of HOSC terms with an $\mathbf{x}$ boundary may contain arbitrary HOSC types as long as the final typing judgment uses types from $\mathbf{x}$ only.

Next we introduce several notions of contextual equivalence for HOSC-terms parameterised by $\mathbf{x}$. We start with the classic notion based on observing termination. We write $\Gamma \vdash C : \tau \to \tau'$ if $\Gamma, x : \tau \vdash C[x] : \tau'$.

Definition 2 (Contextual Equivalence)**.** Let $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$. Given $\mathbf{x}$-terms $\Gamma \vdash M_1, M_2 : \tau$, we define $\Gamma \vdash M_1 \cong^{\mathbf{x}}_{ter} M_2$ to hold, when for all $\mathbf{x}$-contexts $\vdash C : \tau \to \tau'$, we have $(C[M_1], \emptyset) \Downarrow_{ter}$ iff $(C[M_2], \emptyset) \Downarrow_{ter}$.

We also consider another way of testing, based on observing whether a program can reach a breakpoint (error point) inside a context. Technically, the breakpoints are represented as occurrences of a special free error variable $err : \mathrm{Unit} \to \mathrm{Unit}$. Reaching a breakpoint then corresponds to convergence to a stuck configuration of the form $(K[err()], h)$: we write $(M, h) \Downarrow_{err}$ if there exist K, h' such that $(M, h) \to^* (K[err()], h')$.

Definition 3 (Contextual Equivalence via Error)**.** Let $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$. Given $\mathbf{x}$-terms $\Gamma \vdash M_1, M_2 : \tau$ with $err \notin \mathrm{dom}(\Gamma)$, we define $\Gamma \vdash M_1 \cong^{\mathbf{x}}_{err} M_2$ to hold, when for all $\mathbf{x}$-contexts $err : \mathrm{Unit} \to \mathrm{Unit} \vdash C : \tau \to \tau'$, we have $(C[M_1], \emptyset) \Downarrow_{err}$ iff $(C[M_2], \emptyset) \Downarrow_{err}$.

For the languages we study, it is known that $\cong^{\mathbf{x}}_{err}$ is always at least as discriminating as $\cong^{\mathbf{x}}_{ter}$ (i.e. $\cong^{\mathbf{x}}_{err}$ implies $\cong^{\mathbf{x}}_{ter}$) and, for contexts with control, they coincide: $\cong^{\mathbf{x}}_{err} = \cong^{\mathbf{x}}_{ter}$ for $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}\}$. Intuitively, for $\mathbf{x} \in \{\mathrm{HOS}, \mathrm{GOS}\}$, $\cong^{\mathbf{x}}_{err}$ is stricter because it can detect differences in behaviour regardless of whether or not they lead to termination later, whereas $\cong^{\mathbf{x}}_{ter}$ only picks up differences in terminating computations. With control, this difference disappears, because a context can trigger termination at any point.

For higher-order languages with state and control, it is well known that contextual testing can be restricted to evaluation contexts after instantiating the free variables of terms to closed values (the so-called *closed instances of use*, CIU). Let us write $\Sigma, \Gamma' \vdash \gamma : \Gamma$ for substitutions γ such that, for any $(x, \sigma_x) \in \Gamma$, the term $\gamma(x)$ is a value satisfying $\Sigma; \Gamma' \vdash \gamma(x) : \sigma_x$. Then $M\{\gamma\}$ stands for the outcome of applying γ to M.

Definition 4 (CIU Equivalence)**.** Let $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$ and let $\Gamma \vdash M_1, M_2 : \tau$ be HOSC terms with an $\mathbf{x}$ boundary. We write $\Gamma \vdash M_1 \cong^{\mathbf{x}(ciu)}_{err} M_2 : \tau$, when for all Σ, h, K, γ, all built from $\mathbf{x}$ syntax, such that $h : \Sigma; e\hat{\imath}r,\ \Sigma; e\hat{\imath}r \vdash K : \tau \to \tau'$,

and $\Sigma; e\hat{r}r \vdash \gamma : \Gamma$, we have $(K[M_1\{\gamma\}], h) \Downarrow_{err}$ iff $(K[M_2\{\gamma\}], h) \Downarrow_{err}$, where $err \notin \mathrm{dom}(\Gamma)$ and $e\hat{r}r$ stands for $err : \mathrm{Unit} \to \mathrm{Unit}$.

Note that we consider an asymmetric version of CIU equivalence here: while contexts are taken from $\mathbf{x}$, programs are in HOSC. In the symmetric setting, one can prove that CIU and contextual equivalence coincide.

Lemma 5 (CIU Lemma)**.** *Let* $\mathbf{x} \in \{$ HOSC, GOSC, HOS, GOS$\}$ *and* M_1, M_2 *be two* $\mathbf{x}$*-terms. Then we have* $\Gamma \vdash M_1 \cong^{\mathbf{x}}_{err} M_2$ *iff* $\Gamma \vdash M_1 \cong^{\mathbf{x}(ciu)}_{err} M_2$.

The equivalences $\cong^{\mathbf{x}(ciu)}_{err}$ will play an important role in the paper. As shown in [7], they correspond to trace equivalence in operational game semantics. We will review the connection in the next section and, using it as a foundation, develop a sound and complete technique for establishing $\cong^{\mathbf{x}(ciu)}_{err}$ (and, thus, $\cong^{\mathbf{x}}_{err}$ for $\mathbf{x}$-terms). The techniques will be applicable to a class of terms that we call *cr-free*.

Definition 6. A HOSC term $\Gamma \vdash M : \tau$ is ***cr-free*** if it does not contain occurrences of $\mathrm{cont}_\sigma K$ and locations, and its boundary types are cont- and ref-free.

We stress that the above boundary restriction applies to Γ and τ only, and subterms of M may well contain arbitrary HOSC types and occurrences of ref_σ, $\mathrm{call/cc}_\sigma$, throw_σ for any σ. The large majority of examples from the literature, e.g. [14], [15], [16], concern cr-free terms. The fact that cr-free terms may not contain subterms $\mathrm{cont}_\tau K$ or ℓ is not really a restriction, as $\mathrm{cont}_\tau K$ and ℓ are run-time constructs and not really meant to be used directly by programmers. Note that the boundary of a cr-free term is an $\mathbf{x}$ boundary for any $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$. Thus, they are an ideal common ground for comparing the discriminating power of HOSC, GOSC, HOS and GOS contexts. We will discuss the scope for extending our results outside of the cr-free fragment, and for richer type systems, in Section VII.

Example 7. Let $\Gamma = \{f : ((\mathrm{Unit} \to \mathrm{Unit}) \to \mathrm{Unit}) \to \mathrm{Unit}, h : \mathrm{Unit} \to \mathrm{Unit}\}$. In Figure 3, we present three terms $\Gamma \vdash M_i : \mathrm{Bool}$ $(i = 1, 2, 3)$, which will be used as running examples. Using the methodology of the paper, we will be able to establish the following relationships between the terms: $M_1 \cong^{\mathrm{GOSC}}_{err} M_2$ (but $M_1 \ncong^{\mathrm{HOS}}_{err} M_2$), $M_2 \cong^{\mathrm{HOS}}_{err} M_3$ (but $M_2 \ncong^{\mathrm{GOSC}}_{err} M_3$), $M_1 \cong^{\mathrm{GOS}}_{err} M_3$ (but $M_1 \ncong^{\mathrm{GOSC}}_{err} M_3$ and $M_1 \ncong^{\mathrm{HOS}}_{err} M_3$). For inequivalences, we will rely on trace equivalence (Theorem 17) and, for equivalence, we will take advantage of Kripke Normal-Form Bisimulations (Theorem 30).

III. Operational game semantics

Game semantics models interactions between terms and contexts as a dialogue between two players, called O (context) and P (term). Accordingly, by imposing restrictions on O-moves, one can try to capture constraints imposed on contexts. The absence of control constructs in contexts turns out to correspond to a *bracketing* condition [20], while the absence of higher-order references corresponds to *visibility* [21]. An operational account of the correspondences has been presented in [7] through a series of labelled transition systems $\mathbf{x}[\mathrm{HOSC}]$, which generate traces corresponding to interactions of HOSC terms with $\mathbf{x}$-contexts. In this section, we give a unifying presentation of the results as a single LTS, called $\mathcal{L}_{\mathbf{x}}$, which can be specialised to each case $\mathbf{x}$ through simple local adjustments.

1) Names: Operational game semantics relies on countably infinite sets of names that both players will use in their moves.

Definition 8. Let $\mathrm{FNames} = \biguplus_{\sigma,\sigma'} \mathrm{FNames}_{\sigma\to\sigma'}$ be the set of ***function names***, partitioned into mutually disjoint countably infinite sets $\mathrm{FNames}_{\sigma\to\sigma'}$. We use f, g to range over FNames , and write $f : \sigma \to \sigma'$ for $f \in \mathrm{FNames}_{\sigma\to\sigma'}$. Analogously, let $\mathrm{CNames} = \biguplus_\sigma \mathrm{CNames}_\sigma$ be the set of ***continuation names***. We use c, d to range over CNames, and write $c : \sigma$ for $c \in \mathrm{CNames}_\sigma$. Note that the constants represent continuations, so the "real" type of c is $\mathrm{cont}\ \sigma$, but we write $c : \sigma$ for the sake of brevity. We assume $\mathrm{CNames}, \mathrm{FNames}$ are disjoint and let $\mathrm{Names} = \mathrm{FNames} \uplus \mathrm{CNames}$. Elements of Names will be woven into various constructions in the paper, e.g. terms, heaps, etc. Then we write $\nu(X)$ to refer to the set of names used in some entity X.

2) Values: When players use names during play, they will be required to specify values to which the names are applied. Because of the shape of boundary types in cr-free terms (in particular, the presence of product types), the relevant values are tuples consisting of $()$, integers, booleans and functions. To capture this shape, we introduce a notion of ***abstract values***, which are simply patterns that match such values: they are generated by the grammar:

$$A, B \triangleq () \mid \mathbf{tt} \mid \mathbf{ff} \mid \widehat{n} \mid f \mid \langle A, B\rangle$$

with the proviso that no name may occur more than once. They can be seen as a name-based representation of ultimate patterns introduced in [22]. As function names are intrinsically typed, we assign types to abstract values in the obvious way, writing $A : \sigma$. Observe that any closed value V of a cont- and ref-free type σ can be decomposed into an abstract value A (pattern) and the corresponding substitution γ (matching). Given a value V of a (cr-free) type σ, we write $\mathbf{AVal}_\sigma(V)$ for the set of all pairs (A, γ) such that A is an abstract value and $\gamma : \nu(A) \to \mathrm{Vals}$ is a substitution such that $A\{\gamma\} = V$ (an inductive definition of $\mathbf{AVal}_\sigma(V)$ is given in Figure 4).

Remark 9. Note that, by writing $\cdot$, we mean to require implicitly that the function domains be disjoint. Similarly, when writing $\uplus$, we stipulate that the argument sets be disjoint.

Example 10. Let $\sigma = ((\mathrm{Int} \to \mathrm{Bool}) \times \mathrm{Int}) \times (\mathrm{Int} \to \mathrm{Bool})$ and $V \equiv \langle\langle \lambda x^{\mathrm{Int}}.x = 1, 2\rangle, \lambda x^{\mathrm{Int}}.x \neq 3\rangle$. Then $\mathbf{AVal}_\sigma(V)$ consists of $(\langle\langle f, 2\rangle, g\rangle, [f \mapsto (\lambda x^{\mathrm{Int}}.x = 1), g \mapsto (\lambda x^{\mathrm{Int}}.x \neq 3)])$, where $f, g \in \mathrm{FNames}_{\mathrm{Int}\to\mathrm{Bool}}$ and $f \neq g$.

3) Play: During play, each name will have its owner, who will be the player that introduced the name into the game. Accordingly, we will refer to names as O-names or P-

$$
\begin{array}{lcl}
M_1 & \triangleq & \mathrm{let}\, b = \mathrm{ref}\,(\mathtt{tt})\,\mathrm{in}\, f(\lambda g^{\mathrm{Unit}\to\mathrm{Unit}}.\,\mathrm{if}\,(!b)\,\mathrm{then}\,(b := \mathtt{ff};\ h();\ g();\ b := \mathtt{tt})\,\mathrm{else}\,());\, !b \\
M_2 & \triangleq & \mathrm{let}\, b = \mathrm{ref}\,(\mathtt{tt})\,\mathrm{in}\, f(\lambda g^{\mathrm{Unit}\to\mathrm{Unit}}.\,\mathrm{if}\,(!b)\,\mathrm{then}\,(h();\ b := \mathtt{ff};\ g();\ b := \mathtt{tt})\,\mathrm{else}\,());\, !b \\
M_3 & \triangleq & \mathrm{let}\, b = \mathrm{ref}\,(\mathtt{tt})\,\mathrm{in}\, f(\lambda g^{\mathrm{Unit}\to\mathrm{Unit}}.\,\mathrm{if}\,(!b)\,\mathrm{then}\,(h();\ b := \mathtt{ff};\ g();\ b := \mathtt{tt})\,\mathrm{else}\,());\, \mathtt{tt}
\end{array}
$$

Fig. 3. Terms M_1, M_2, M_3

$$
\begin{array}{lcl}
\mathbf{AVal}_\sigma(V) & \triangleq & \{(V, \emptyset)\} \quad \text{for } \sigma \in \{\mathrm{Unit}, \mathrm{Bool}, \mathrm{Int}\} \\
\mathbf{AVal}_{\sigma\to\sigma'}(V) & \triangleq & \{(f, [f \mapsto V]) \mid f \in \mathrm{FNames}_{\sigma\to\sigma'}\} \\
\mathbf{AVal}_{\sigma\times\sigma'}(\langle U, V\rangle) & \triangleq & \{(\langle A_1, A_2\rangle, \gamma_1 \cdot \gamma_2) \mid (A, \gamma_1) \in \mathbf{AVal}_\sigma(U),\ (A_2, \gamma_2) \in \mathbf{AVal}_{\sigma'}(V)\}
\end{array}
$$

Fig. 4. Value decompositions into abstract values and substitutions

names. The starting point for play will be a set N_O of names corresponding to the associated typing judgment. These names are assumed to have been introduced by O, i.e. they are O-names. After that, P and O will take turns making moves (P goes first) and their moves may introduce new names into play. Moves must take one of the four shapes specified below so that the resultant sequence forms an N_O-trace, defined below.

Definition 11 (N_O-trace)**.** Let $N_O \subseteq \mathrm{Names}$. The empty sequence ϵ is an N_O-trace. If t is an N_O-trace then tm is an N_O-trace as long as it satisfies one of the conditions below.

- t is of even length, $m = \bar{f}(A, c')$ (***P-Question***) or $m = \bar{c}(A)$ (***P-Answer***), where $f : \sigma \to \sigma'$, $A : \sigma$, $c' : \sigma'$ and $c : \sigma$. Here f, c must be O-names (introduced by O in t or from N_O), while all names in A and c' must be fresh (must not occur in t). These fresh names are considered to be introduced by P in m and become P-names in tm.
- t is of odd length, $m = f(A, c')$ (***O-Question***) or $m = c(A)$ (***O-Answer***), where $f : \sigma \to \sigma'$, $A : \sigma$, $c' : \sigma'$ and $c : \sigma$. Here f, c must be P-names introduced by P in t, while all names in A and c' must be fresh (cannot occur in t). These fresh names are considered to be introduced by O in m and become O-names in tm.

We will refer to f and c respectively as the *head names* of m. Note that the head name of a move by one player always belongs to the other player.

Example 12. Let $N_O = \{f : ((\mathrm{Unit} \to \mathrm{Unit}) \to \mathrm{Unit}) \to \mathrm{Unit}, h : \mathrm{Unit} \to \mathrm{Unit}, c : \mathrm{Bool}\}$. The sequences $\mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_3$ in Figure 5 are N_O-traces.

4) O-visibility and O-bracketing: To spell out the O-visibility and O-bracketing constraints, for any odd-length N_O-trace t, we define the set $\mathsf{Vis}_O(t)$ of ***O-visible names*** and the ***top continuation name*** $\mathsf{Top}_O(t)$. The definitions are given in Figure 6. Note that the definition of $\mathsf{Vis}_O(t)$ is based on tracing the connection between a move and the point of introduction of its head name. In game semantics, such links are referred to as *justification pointers*.

Definition 13. An N_O-trace t is ***O-visible*** if, for any even-length prefix $t'\, f'(A', c')$ of t, we have $f' \in \mathsf{Vis}_O(t')$ and, for any even-length prefix $t'\, c'(A')$ of t, we have $c' \in \mathsf{Vis}_O(t')$.

t is ***O-bracketed*** if, for any even-length prefix $t'\, c'(A)$ of t (i.e. any prefix ending with an O-answer), we have $c' \in \mathsf{Top}_O(t')$.

Remark 14. Observe that, whenever $\mathsf{Top}_O(t)$ is defined, it contains a single continuation name which is also in $\mathsf{Vis}_O(t)$. Consequently, O-bracketing implies O-visibility for answers.

Example 15. We revisit the traces from Figure 5. The fourth move $x(g_2, c_4)$ in $\mathtt{t}_1$ breaks O-visibility, because $\mathsf{Vis}_O(\bar{f}(x, c_1)\ x(g_1, c_2)\ \bar{h}((), c_3)) = \{c_3\}$. $\mathtt{t}_2$ is O-visible, e.g. the sixth move $c_1(())$ does not violate O-visibility, because $c_1 \in \mathsf{Vis}_O(\bar{f}(x, c_1)\ x(g_1, c_2)\ \bar{h}((), c_3)\ c_3(())\ \bar{g_1}((), c_4)) = \{x, c_1, c_4\}$. However, $c_1(())$ in $\mathtt{t}_2$ breaks O-bracketing, because $\mathsf{Top}_O(\bar{f}(x, c_1)\ x(g_1, c_2)\ \bar{h}((), c_3)\ c_3(())\ \bar{g_1}((), c_4)) = \{c_4\}$. $\mathtt{t}_3$ satisfies both O-visibility and O-bracketing.

5) The LTS $\mathcal{L}_{\mathbf{x}}$: To derive the set of traces corresponding to a given term, we use the LTS $\mathcal{L}_{\mathbf{x}}$ given in Figure 8. Next we explain how it works. First of all, the LTS is based on extended HOSC syntax, which incorporates function names as constants and values. In contrast, continuation names are not terms on their own. Instead, they are built into the syntax via a new construct $\mathrm{cont}_\sigma\,(K, c)$, which is a staged continuation that first evaluates terms inside K and, if this produces a value, the value is passed to c. $\mathrm{cont}_\sigma\,(K, c)$ is also regarded as a syntactic value. Note that we remove the old construct $\mathrm{cont}_\sigma\, K$ from the extended syntax. The typing and reduction rules that are added to the definition of HOSC are summarised in Figure 7. The operational semantics $\to$ underpinning $\mathcal{L}_{\mathbf{x}}$ is based on triples (M, c, h) such that $\Sigma; \Gamma \vdash M : \sigma$, $c \in \mathrm{CNames}_\sigma$ and $h : \Sigma$. The continuation name c is used to represent the surrounding context, which is left abstract. The previous operational rules $\to$ are embedded into the new reduction $\to$ using the rule:

$$\frac{(M, h) \to (M', h')}{(M, c, h) \to (M', c, h')}$$

The two reduction rules for handling continuations in HOSC (last line of Figure 2) are replaced with two analogous rules, shown in Figure 7.

The $\mathcal{L}_{\mathbf{x}}$ LTS features two kinds of configurations: $\langle \gamma, \phi, h, H_F, H_C, Fn, Cn\rangle$ (*passive*, O to play) and $\langle M, c, \gamma, \phi, h, H_F, H_C\rangle$ (*active*, internal or P to play). In both, ϕ contains all names introduced so far by both players and h is the current heap. γ is an *environment* mapping function names introduced by P to function values, and continuation names introduced by P to pairs (K, c). Fn

$$
\begin{array}{lcl}
\mathtt{t}_1 & = & \bar{f}(x,c_1)\quad x(g_1,c_2)\quad \bar{h}((),c_3)\quad x(g_2,c_4)\quad \bar{h}((),c_5)\\
\mathtt{t}_2 & = & \bar{f}(x,c_1)\quad x(g_1,c_2)\quad \bar{h}((),c_3)\quad c_3(())\quad \bar{g_1}((),c_4)\quad c_1(())\quad \bar{c}(\mathbf{ff})\\
\mathtt{t}_3 & = & \bar{f}(x,c_1)\quad x(g_1,c_2)\quad \bar{h}((),c_3)\quad c_3(())\quad \bar{g_1}((),c_4)\quad c_4(())\quad \bar{c_2}(())\quad c_1(())\quad \bar{c}(\mathbf{tt})
\end{array}
$$

Fig. 5. N_O-traces for $N_O = \{f : ((\mathrm{Unit} \to \mathrm{Unit}) \to \mathrm{Unit}) \to \mathrm{Unit},\ h : \mathrm{Unit} \to \mathrm{Unit},\ c : \mathrm{Bool}\}$

$$
\begin{array}{rcll}
\mathsf{Vis}_O(t\ \bar{c}(A)) & = & \nu(A) & c \in N_O\\
\mathsf{Vis}_O(t\ \bar{f}(A',c)\ t'\ \bar{c}(A)) & = & \mathsf{Vis}_O(t) \cup \nu(A) & \\
\mathsf{Vis}_O(t\ \bar{f}(A,c)) & = & \nu(A) \cup \{c\} & f \in N_O\\
\mathsf{Vis}_O(t\ f'(A',c')\ t'\ \bar{f}(A,c)) & = & \mathsf{Vis}_O(t) \cup \nu(A) \cup \{c\} & f \in \nu(A')\\
\mathsf{Vis}_O(t\ c'(A')\ t'\ \bar{f}(A,c)) & = & \mathsf{Vis}_O(t) \cup \nu(A) \cup \{c\} & f \in \nu(A')
\end{array}
$$

$$
\begin{array}{rcll}
\mathsf{Top}_O(t\ \bar{c}(A)) & = & \emptyset & c \in N_O\\
\mathsf{Top}_O(t\ f(A',c)\ t'\ \bar{c}(A)) & = & \mathsf{Top}_O(t) & \\
\mathsf{Top}_O(t\ \bar{f}(A,c)) & = & \{c\} &
\end{array}
$$

Fig. 6. O-visible names $\mathsf{Vis}_O(t)$ and top continuation name $\mathsf{Top}_O(t)$.

$$
\frac{f \in \mathrm{FNames}_{\sigma\to\sigma'}}{\Sigma;\Gamma \vdash f : \sigma \to \sigma'} \qquad \frac{\Sigma;\Gamma \vdash K : \sigma \to \sigma' \quad c \in \mathrm{CNames}_{\sigma'}}{\Sigma;\Gamma \vdash \mathrm{cont}_\sigma\,(K,c) : \mathrm{cont}\ \sigma}
$$

$$
\frac{(M,h) \to (M',h')}{(M,c,h) \to (M',c,h')}
$$

$$
\begin{array}{rcl}
(K[\mathrm{call/cc}_\tau(x.M)],c,h) & \to & (K[M\{\mathrm{cont}_\tau\,(K,c)/x\}],c,h)\\
(K[\mathrm{throw}_\tau\ \mathrm{V}\ \mathrm{to}\ \mathrm{cont}_\tau\,(K',c')],c,h) & \to & (K'[V],c',h)
\end{array}
$$

Fig. 7. Modifications of HOSC syntax for use in $\mathcal{L}_{\mathbf{x}}$

$$
\begin{array}{ll}
(P\tau) & \langle M,c,\gamma,\phi,h,H_F,H_C\rangle \xrightarrow{\tau} \langle N,c',\gamma,\phi,h',H_F,H_C\rangle\\
& \text{when } (M,c,h) \to (N,c',h')\\
(PA) & \langle V,c,\gamma,\phi,h,H_F,H_C\rangle \xrightarrow{\bar{c}(A)} \langle \gamma\cdot\gamma',\phi \uplus \nu(A),h,H_F,H_C,F^{\mathbf{x}}_{PA} \uplus \nu(A),C^{\mathbf{x}}_{PA}\rangle\\
& \text{when } c : \sigma,\ (A,\gamma') \in \mathbf{AVal}_\sigma(V)\\
(PQ) & \langle K[fV],c,\gamma,\phi,h,H_F,H_C\rangle \xrightarrow{\bar{f}(A,c')} \langle \gamma\cdot\gamma'\cdot[c' \mapsto (K,c)],\phi \uplus \phi',h,H_F,H_C,F^{\mathbf{x}}_{PQ} \uplus \nu(A),C^{\mathbf{x}}_{PQ} \uplus \{c'\})\rangle\\
& \text{when } f : \sigma \to \sigma',\ (A,\gamma') \in \mathbf{AVal}_\sigma(V),\ c' : \sigma' \text{ and } \phi' = \nu(A) \uplus \{c'\}\\
(OA) & \langle \gamma,\phi,h,H_F,H_C,Fn,Cn\rangle \xrightarrow{c(A)} \langle K[A],c',\gamma,\phi \uplus \nu(A),h,H_F\cdot[\nu(A) \mapsto Fn],H_C\cdot[\nu(A) \mapsto Cn]\rangle\\
& \text{when } c \in Cn,\ c : \sigma,\ A : \sigma,\ \gamma(c) = (K,c')\\
(OQ) & \langle \gamma,\phi,h,H_F,H_C,Fn,Cn\rangle \xrightarrow{f(A,c)} \langle VA,c,\gamma,\phi \uplus \phi',h,H_F\cdot[\phi' \mapsto Fn],\ H_C\cdot[\phi' \mapsto Cn]\rangle\\
& \text{when } f \in Fn,\ f : \sigma \to \sigma',\ A : \sigma,\ c : \sigma',\ \gamma(f) = V \text{ and } \phi' = \nu(A) \uplus \{c\}
\end{array}
$$

Given $N \subseteq \mathrm{Names}$, $[N \mapsto \mathcal{V}]$ stands for the map $[n \mapsto \mathcal{V} \mid n \in N]$.

Fig. 8. $\mathcal{L}_{\mathbf{x}}$ transition rules

represents the set of function P-names currently available to O, while H_F contains historical information about availability. H_F is a function from all O-names encountered so far to sets of function P-names. The LTS will maintain the invariant that $\mathrm{dom}(H_F)$ is the set of all O-names played so far and, for each such O-name o, $H_F(o)$ consists of function P-names that were available to O when o was first used. Similarly, Cn represents the set of currently available continuation P-names and H_C plays a role analogous to H_F, recording historical information about availability of continuation P-names.

Because of the $c \in Cn$ and $f \in Fn$ constraints in rules $(OA), (OQ)$ respectively, passive configurations may progress only if O uses one of the currently available names as the head name. Note how the information stored in H_C and H_F is updated at this point to take new O-names into account.

In rules $(PA), (PQ)$, the LTS calculates the Fn, Cn components of the successor configuration, deciding which P-names should be made available to O. Names that are introduced by P in the current label become immediately available in each case (as $\nu(A)$ and $\{c'\}$ respectively). Other names to be made available are given by the $F^{\mathbf{x}}_{PA}, C^{\mathbf{x}}_{PA}, F^{\mathbf{x}}_{PQ}, C^{\mathbf{x}}_{PQ}$ components according to the table below, where ϕ_{PF} (resp. ϕ_{PC} stands for all function (resp. continuation) P-names, i.e. $\phi_{PF} = \mathrm{dom}(\gamma) \cap \mathrm{FNames}$ and $\phi_{PC} = \mathrm{dom}(\gamma) \cap \mathrm{CNames}$. The table is designed in such a way that the components enforce the game-semantic conditions corresponding to $\mathbf{x}$, as listed below. This is the only part of $\mathcal{L}_{\mathbf{x}}$ that really depends on $\mathbf{x}$. For $\mathbf{x} = \mathrm{HOSC}$, all P-names are being made available to the next configuration. In other cases, the updates follow the definition of $\mathsf{Vis}_O(t)$ and $\mathsf{Top}_O(t)$, as applicable. Note that occurrences in the table of H_F, H_C correspond to following justification pointers. The $\mathcal{L}_{\mathbf{x}}$ LTS amounts to a uniform

$\mathbf{x}$	$F^{\mathbf{x}}_{PA}$	$C^{\mathbf{x}}_{PA}$	$F^{\mathbf{x}}_{PQ}$	$C^{\mathbf{x}}_{PQ}$
HOSC	ϕ_{PF}	ϕ_{PC}	ϕ_{PF}	ϕ_{PC}
GOSC	$H_F(c)$	$H_C(c)$	$H_F(f)$	$H_C(f)$
HOS	ϕ_{PF}	$H_C(c)$	ϕ_{PF}	$\emptyset$
GOS	$H_F(c)$	$H_C(c)$	$H_F(f)$	$\emptyset$

Fig. 9. Specification of components $F^{\mathbf{x}}_{PA}, C^{\mathbf{x}}_{PA}, F^{\mathbf{x}}_{PQ}, C^{\mathbf{x}}_{PQ}$

$\mathbf{x}$ (context)	O-questions	O-answers
HOSC	unrestricted	unrestricted
GOSC	O-visibility	O-visibility
HOS	unrestricted	O-bracketing
GOS	O-visibility	O-bracketing

Fig. 10. $\mathbf{x}$-contexts and corresponding constraints

presentation of the series of LTSs, called $\mathbf{x}[\mathrm{HOSC}]$, from [7]. By design, $\mathcal{L}_{\mathrm{HOSC}}$ produces the same traces as the LTS HOSC[HOSC] from [7]. For other $\mathbf{x}$, it produces only those traces from HOSC[HOSC] that satisfy the restrictions relevant to $\mathbf{x}$. Consequently, in each case, the traces produced by $\mathcal{L}_{\mathbf{x}}$ are the same as those of $\mathbf{x}[\mathrm{HOSC}]$.

Let us write $\mathbf{Tr}_{\mathbf{x}}(\mathbf{C})$ for the set of traces produced by $\mathcal{L}_{\mathbf{x}}$ started in configuration $\mathbf{C}$. To state the full abstraction result for $\mathcal{L}_{\mathbf{x}}$, we need to specify initial configurations. Let $\Gamma \vdash M : \tau$ be a cr-free HOSC term such that $\Gamma = \{x_1 : \sigma_1, \cdots, x_k : \sigma_k\}$. A Γ-***assignment*** ρ is a map from $\{x_1, \cdots, x_k\}$ to the set of abstract values such that, for all $1 \leq i \neq j \leq k$, we have $\rho(x_i) : \sigma_i$ and $\nu(\rho(x_i)) \cap \nu(\rho(x_j)) = \emptyset$. ρ simply creates a supply of names corresponding to the context. Let $c : \tau$ and $N_O = \nu(\rho) \cup \{c\}$. Then the active initial configuration $\mathsf{C}^{\rho,c}_M$ is defined to be $\langle M\{\rho\}, c, \emptyset, N_O, \emptyset, [N_O \mapsto \emptyset], [N_O \mapsto \emptyset]\rangle$.

Definition 16. Let $\mathbf{x} \in \{\mathrm{HOSC}, \mathrm{GOSC}, \mathrm{HOS}, \mathrm{GOS}\}$. The $\mathbf{x}$ ***trace semantics*** of a cr-free HOSC term $\Gamma \vdash M : \tau$ is defined to be $\mathbf{Tr}_{\mathbf{x}}(\Gamma \vdash M : \tau) \triangleq \{((\rho, c), t) \mid \rho \text{ is a } \Gamma\text{-assignment}, c : \tau, t \in \mathbf{Tr}_{\mathbf{x}}(\mathsf{C}^{\rho,c}_M)\}$.

We can then restate the full abstraction results from [7] as follows. They establish an exact correspondence between ciu-equivalence and trace equivalence.

Theorem 17. *For any cr-free HOSC terms* $\Gamma \vdash M_1, M_2 : \tau$, $\Gamma \vdash M_1 \cong^{\mathbf{x}(ciu)}_{err} M_2$ *iff* $\mathbf{Tr}_{\mathbf{x}}(\Gamma \vdash M_1 : \tau) = \mathbf{Tr}_{\mathbf{x}}(\Gamma \vdash M_2 : \tau)$.

From Lemma 5, we deduce an exact correspondence between contextual equivalence and trace equivalence, in the symmetric setting.

Corollary 18. *For any cr-free* $\mathbf{x}$*-terms* $\Gamma \vdash M_1, M_2 : \tau$, $\Gamma \vdash M_1 \cong^{\mathbf{x}}_{err} M_2$ *iff* $\mathbf{Tr}_{\mathbf{x}}(\Gamma \vdash M_1 : \tau) = \mathbf{Tr}_{\mathbf{x}}(\Gamma \vdash M_2 : \tau)$.

Example 19. We revisit the terms $\Gamma \vdash M_i$ $(i = 1, 2, 3)$ from Figure 3 and traces $\mathtt{t}_i$ from Figure 5. Let $\rho = [f \mapsto f, h \mapsto h]$ (for simplicity, we conflate variable names with function names) and c : Bool. Then we have $\mathtt{t}_1 \notin \mathbf{Tr}_{\mathrm{HOS}}(\mathsf{C}^{\rho,c}_{M_1})$ but $\mathtt{t}_1 \in \mathbf{Tr}_{\mathrm{HOS}}(\mathsf{C}^{\rho,c}_{M_2})$ and $\mathtt{t}_1 \in \mathbf{Tr}_{\mathrm{HOS}}(\mathsf{C}^{\rho,c}_{M_3})$. Hence, by Theorem 17, $\Gamma \vdash M_1 \not\cong^{\mathrm{HOS}}_{err} M_2$ and $\Gamma \vdash M_1 \not\cong^{\mathrm{HOS}}_{err} M_3$. For $\mathtt{t}_2$, we have $\mathtt{t}_2 \in \mathbf{Tr}_{\mathrm{GOSC}}(\mathsf{C}^{\rho,c}_{M_1}) \cap \mathbf{Tr}_{\mathrm{GOSC}}(\mathsf{C}^{\rho,c}_{M_2})$, and $\mathtt{t}_2 \notin \mathbf{Tr}_{\mathrm{GOSC}}(\mathsf{C}^{\rho,c}_{M_3})$. Thus, $\Gamma \vdash M_1 \not\cong^{\mathrm{GOSC}}_{err} M_3$ and $\Gamma \vdash M_2 \not\cong^{\mathrm{GOSC}}_{err} M_3$. Note that $\mathtt{t}_3 \in \mathbf{Tr}_{\mathrm{GOS}}(\mathsf{C}^{\rho,c}_{M_1}) \cap \mathbf{Tr}_{\mathrm{GOS}}(\mathsf{C}^{\rho,c}_{M_2}) \cap \mathbf{Tr}_{\mathrm{GOS}}(\mathsf{C}^{\rho,c}_{M_3})$. In the second half of the paper, we will establish $\Gamma \vdash M_1 \cong^{\mathrm{GOSC}}_{err} M_2$, $\Gamma \vdash M_2 \cong^{\mathrm{HOSC}}_{err} M_3$ and $\Gamma \vdash M_1 \cong^{\mathrm{GOS}}_{err} M_3$.

IV. From LTS to KNFB

Recall that Theorem 17 recasts $\cong^{\mathbf{x}(ciu)}_{err}$-equivalence as trace equivalence in the respective LTS $\mathcal{L}_{\mathbf{x}}$. Since $\mathcal{L}_{\mathbf{x}}$ is deterministic (up to the choice of reference names), this corresponds to bisimilarity. Based on this observation, we develop a relational framework for proving bisimilarity in all four cases $\mathbf{x}$. Unfortunately, bisimulations defined directly on configurations of $\mathcal{L}_{\mathbf{x}}$ would be quite complicated, not least due to the growth of the environment γ and evolution of the heap h. To address this complexity, we introduce ***Kripke Normal-Form Bisimulations*** (KNFB) as a friendlier technique.

- The associated bisimulations will not be defined on configurations, but directly on terms, evaluation contexts and values.
- To disentangle the reasoning about the heap from the reasoning about program evaluation, we will rely on a notion of transition systems of invariants, following the work on Kripke Logical Relations [16]. In our case, the transition system will be equipped with two transition relations, $\sqsubseteq_{\mathsf{OQ}}$ and $\sqsubseteq_{\mathsf{OA}}$, introduced to model the availability of function and continuation names respectively. Remarkably, the differences between the four fragments will merely boil down to local conditions controlling how the two relations have to be maintained during the proof. As a consequence, we will be able to develop our techniques simultaneously for all four cases.
- To address the growth of γ, whenever values or evaluation contexts would be added to $\mathcal{L}_{\mathbf{x}}$-configurations, we establish their equivalence upfront. However, as heaps are evolving, it would not be sound to perform these checks for the current heaps only. Similarly, it would be too strong to aim for equivalence with respect to arbitrary heaps. Thus, to account for all relevant uses (in an abstract fashion), we will rely on the transition system of invariants, using $\sqsubseteq_{\mathsf{OQ}}$ for function values and $\sqsubseteq_{\mathsf{OA}}$ for continuations.

We begin with a formal definition of world transition systems.

Definition 20. A ***world transition system*** (WTS) $\mathcal{A}$ is a triple $(\mathrm{Worlds}, \sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}, \mathcal{I})$, where Worlds is a set of states (*worlds*), $\sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}$ are binary reflexive relations on Worlds, and $\mathcal{I}$: Worlds $\to \mathcal{P}(\mathrm{Heap} \times \mathrm{Heap})$ is the *invariant assignment* that associates a set of pairs of heaps to any world.

Intuitively, a world can be seen as an abstraction of (a set of) LTS configurations, where only the heaps matter (via the function $\mathcal{I}$). $w \sqsubseteq_{\mathsf{OQ}} w'$ is meant to capture world evolution that protects the availability of function names introduced by

$$
\begin{array}{lllll}
\mathcal{V}^{\mathbf{x}}_{\mathcal{A}}: & (X_{\mathcal{V}}, X_{\mathcal{K}}, X_{\mathcal{E}}) & \mapsto & \{\ (\beta, V, V, w, \mathcal{H}) \mid V : \beta \ \wedge\ \nu(V) \subseteq \mathrm{dom}(\mathcal{H})\ \} \cup & \text{(with } \beta \in \{\mathrm{Unit}, \mathrm{Bool}, \mathrm{Int}\}\text{)} \\
& & & \{\ (\sigma_1 \times \sigma_2, \langle U_1, U_2\rangle, \langle V_1, V_2\rangle, w, \mathcal{H}) \mid (\sigma_i, U_i, V_i, w, \mathcal{H}) \in X_{\mathcal{V}} \text{ for } i \in \{1,2\}\ \} \cup & \\
& & & \{\ (\sigma \to \sigma', V_1, V_2, w, \mathcal{H}) \mid \forall w' \sqsupseteq^*_{\mathsf{OQ}} w.\ \forall A : \sigma.\ \forall c : \tau.\ (\mathrm{dom}(\mathcal{H}), \nu(A), \{c\} \text{ mutually disjoint}) & \\
& & & \qquad \Longrightarrow (\sigma', V_1 A, c, V_2 A, c, w', \mathcal{H} \cdot [\nu(A) \mapsto w'] \cdot [c \mapsto w']) \in X_{\mathcal{E}}\ \} &
\end{array}
$$

$$
\begin{array}{lllll}
\mathcal{K}^{\mathbf{x}}_{\mathcal{A}}: & (X_{\mathcal{V}}, X_{\mathcal{K}}, X_{\mathcal{E}}) & \mapsto & \{\ (\sigma, \sigma', K_1, c_1, K_2, c_2, w, \mathcal{H}) \mid \forall w' \sqsupseteq^*_{\mathsf{OA}} w.\ \forall A : \sigma.\ (\mathrm{dom}(\mathcal{H}), \nu(A) \text{ disjoint}) & \\
& & & \qquad \Longrightarrow (\sigma', K_1[A], c_1, K_2[A], c_2, w', \mathcal{H} \cdot [\nu(A) \mapsto w'])\ \} \in X_{\mathcal{E}} &
\end{array}
$$

$$
\begin{array}{lllll}
\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}: & (X_{\mathcal{V}}, X_{\mathcal{K}}, X_{\mathcal{E}}) & \mapsto & \{\ (\sigma, M_1, c_1, M_2, c_2, w, \mathcal{H}) \mid \forall (h_1, h_2) \in \mathcal{I}(w).\ P_{Div} \vee P_{PA} \vee P_{PQ}\ \} &
\end{array}
$$

$$
\begin{array}{rcl}
P_{Div} & \triangleq & (M_1, c_1, h_1) \Uparrow\ \wedge\ (M_2, c_2, h_2) \Uparrow \\
P_{PA} & \triangleq & \exists V_1, V_2, c.\ \exists w' \sqsupseteq^{\mathbf{x}}_c (w, \mathcal{H}).\ \exists (h_1', h_2') \in \mathcal{I}(w').\ (\sigma, V_1, V_2, w', \mathcal{H}) \in X_{\mathcal{V}} \wedge \\
& & \quad (M_1, c_1, h_1) \to^* (V_1, c, h_1')\ \wedge\ (M_2, c_2, h_2) \to^* (V_2, c, h_2') \\
P_{PQ} & \triangleq & \exists K_1, V_1, K_2, V_2.\ \exists c_1', c_2' : \tau.\ \exists \sigma_1, \sigma_2.\ \exists f : \sigma_1 \to \sigma_2.\ \exists w' \sqsupseteq^{\mathbf{x}}_f (w, \mathcal{H}).\ \exists (h_1', h_2') \in \mathcal{I}(w'). \\
& & \quad (\sigma_1, V_1, V_2, w', \mathcal{H}) \in X_{\mathcal{V}}\ \wedge\ (\sigma_2, \sigma, K_1, c_1', K_2, c_2', w', \mathcal{H}) \in X_{\mathcal{K}} \\
& & \quad \wedge\ (M_1, c_1, h_1) \to^* (K_1[fV_1], c_1', h_1')\ \wedge\ (M_2, c_2, h_2) \to^* (K_2[fV_2], c_2', h_2')
\end{array}
$$

Fig. 11. $\mathcal{A}^{\mathbf{x}}$-Kripke Normal-Form Bisimulation for $\mathcal{A} = (\mathrm{Worlds}, \sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}, \mathcal{I})$

terms (P) to the environment (O). Intuitively, if such a value was available in w and $w \sqsubseteq_{\mathsf{OQ}} w'$ then the environment can also access it in w'. The role of $w \sqsubseteq_{\mathsf{OA}} w'$ is analogous but it corresponds to continuation names instead. The two relations will allow us to represent the flow of information about P-name availability, depending on $\mathbf{x}$. Given a WTS $\mathcal{A}$, we define three operators $\mathcal{V}^{\mathbf{x}}_{\mathcal{A}}$, $\mathcal{K}^{\mathbf{x}}_{\mathcal{A}}$, $\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}$ on relations for handling values, continuations and expressions respectively. The relevant relations will contain tuples of the form $(\cdots, w, \mathcal{H})$, where w indicates the world in which the $\cdots$ entities are being related, and $\mathcal{H}$ is a partial map from Names to Worlds. $\mathcal{H}$ will be referred to as the *world history* - it records the names that were introduced by the context (O-names) along with the world in which each name was introduced.

Values: : $\mathcal{V}^{\mathbf{x}}_{\mathcal{A}}(X)$ relates identical values at base types and pairs of values that are already related in X. For function values V_1, V_2 to be related in w, we interrogate them in any world w' in which they are available ($w \sqsubseteq^*_{\mathsf{OQ}} w'$) by applying them to an abstract value A and continuation c. Fresh names are used to represent unspecified functional and continuation values in the spirit of open bisimulation [17], [8]. This corresponds to the (OQ) rule in $\mathcal{L}_{\mathbf{x}}$.

Evaluation contexts: : $\mathcal{K}^{\mathbf{x}}_{\mathcal{A}}$ corresponds to testing continuations by providing an abstract value A to two evaluation contexts K_1, K_2. In $\mathcal{L}_{\mathbf{x}}$, this corresponds to the (OA) rule and, similarly, we write $w \sqsubseteq^*_{\mathsf{OA}} w'$ to range over all the worlds w' in which such tests can be legitimately carried out.

Expressions: : The definition of $\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}$ is split into three cases: P_{Div} (both terms diverge), P_{PA} (both terms reduce to values; this corresponds to the (PA) rule), and P_{PQ} (both terms reduce to callbacks; this corresponds to (PQ)). In all three cases, in order for the terms to be related in world w, we execute them with heaps that satisfy the invariant of w. In both P_{PA} and P_{PQ}, we stipulate the existence of a world w' whose invariant captures the new heaps h_1', h_2'. Additionally, the world w' must be related to $(w, \mathcal{H})$ as follows: $(w, \mathcal{H}) \sqsubseteq^{\mathbf{x}}_c w'$ for P_{PA} and $(w, \mathcal{H}) \sqsubseteq^{\mathbf{x}}_f w'$ for P_{PQ}. These conditions are specified in Figure 12. This is the only place where the definition actually depends on $\mathbf{x}$.

Remark 21. The shapes of $\sqsubseteq^{\mathbf{x}}_c, \sqsubseteq^{\mathbf{x}}_f$ in Figure 12 may seem mysterious at first, but in fact they mirror the definitions of $F^{\mathbf{x}}_{PA}, C^{\mathbf{x}}_{PA}, F^{\mathbf{x}}_{PQ}, C^{\mathbf{x}}_{PQ}$ from Figure 9. For example, $F^{\mathbf{x}}_{PA} = \phi_{PF}$ (making all old function names available) corresponds to inheriting all function names from the original world, i.e. $w \sqsubseteq_{\mathsf{OQ}} w'$. The case $F^{\mathbf{x}}_{PA} = H_F(c)$ is captured by $\mathcal{H}(c) \sqsubseteq_{\mathsf{OQ}} w'$. Conditions regarding $C^{\mathbf{x}}_{PA}$ can be interpreted similarly, though $\sqsubseteq_{\mathsf{OA}}$ must be used instead of $\sqsubseteq_{\mathsf{OQ}}$. For $F^{\mathbf{x}}_{PQ}$ and $C^{\mathbf{x}}_{PQ}$, the correspondence is analogous except the case $C^{\mathbf{x}}_{PQ} = \emptyset$, which generates no condition, since no old names are being made available. That is why $\sqsubseteq^{\mathrm{HOS}}_f, \sqsubseteq^{\mathrm{GOS}}_f$ do not refer to $\sqsubseteq_{\mathsf{OA}}$.

Definition 22. Suppose $\mathcal{A} = (\mathrm{Worlds}, \sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}, \mathcal{I})$ is a WTS. Let τ, τ' range over types, M_i over HOSC terms, K_i over evaluation contexts, V_i over syntactic values, c_i over CNames, w over Worlds, and $\mathcal{H}$ over $\mathrm{Names} \rightharpoonup \mathrm{Worlds}$. A triple $(R_{\mathcal{V}}, R_{\mathcal{K}}, R_{\mathcal{E}})$ of relations is *admissible* if $R_{\mathcal{V}}, R_{\mathcal{K}}, R_{\mathcal{E}}$ respectively contain tuples of the form $(\tau, V_1, V_2, w, \mathcal{H})$, $(\tau, \tau', K_1, c_1, K_2, c_2, w, \mathcal{H})$, $(\tau, M_1, c_1, M_2, c_2, w, \mathcal{H})$, and each name in V_i, K_i, M_i, c_i (as appropriate) is in $\mathrm{dom}(\mathcal{H})$.

Definition 23 ($\mathcal{A}^{\mathbf{x}}$-Kripke Normal-Form Bisimulation)**.** Let $\mathcal{A}$ be a WTS. An admissible triple $R = (R_{\mathcal{V}}, R_{\mathcal{K}}, R_{\mathcal{E}})$ of relations is an ***$\mathcal{A}^{\mathbf{x}}$-KNFB*** if it is a post-fixpoint of $(\mathcal{V}^{\mathbf{x}}_{\mathcal{A}}, \mathcal{K}^{\mathbf{x}}_{\mathcal{A}}, \mathcal{E}^{\mathbf{x}}_{\mathcal{A}})$ defined in Figure 11, i.e. $R \subseteq (\mathcal{V}^{\mathbf{x}}_{\mathcal{A}}(R), \mathcal{K}^{\mathbf{x}}_{\mathcal{A}}(R), \mathcal{E}^{\mathbf{x}}_{\mathcal{A}}(R))$. Note that the definition of $\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}$ relies on conditions $(w, \mathcal{H}) \sqsubseteq^{\mathbf{x}}_c w'$ and $(w, \mathcal{H}) \sqsubseteq^{\mathbf{x}}_f w'$, which are specified in Figure 12. (We write $(w, \mathcal{H}) \sqsubseteq^{\mathbf{x}}_n w'$ and $w' \sqsupseteq^{\mathbf{x}}_n (w, \mathcal{H})$ interchangeably.)

Remark 24. The definition of $\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}$ is reminiscent of bisimulation games: the condition $P_{Div} \vee P_{PA} \vee P_{PQ}$ inside the definition of $\mathcal{E}^{\mathbf{x}}_{\mathcal{A}}$ stipulates identical behaviour from both parties and specifies, through $\sqsubseteq^{\mathbf{x}}_f$ and $\sqsubseteq^{\mathbf{x}}_c$, how the evolution of

$\mathbf{x}$	$(w,\mathcal{H}) \sqsubseteq_c^{\mathbf{x}} w'$	$(w,\mathcal{H}) \sqsubseteq_f^{\mathbf{x}} w'$
HOSC	$w \sqsubseteq_{\mathsf{OQ}} w' \wedge w \sqsubseteq_{\mathsf{OA}} w'$	$w \sqsubseteq_{\mathsf{OQ}} w' \wedge w \sqsubseteq_{\mathsf{OA}} w'$
GOSC	$\mathcal{H}(c) \sqsubseteq_{\mathsf{OQ}} w' \wedge \mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$	$\mathcal{H}(f) \sqsubseteq_{\mathsf{OQ}} w' \wedge \mathcal{H}(f) \sqsubseteq_{\mathsf{OA}} w'$
HOS	$w \sqsubseteq_{\mathsf{OQ}} w' \wedge \mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$	$w \sqsubseteq_{\mathsf{OQ}} w'$
GOS	$\mathcal{H}(c) \sqsubseteq_{\mathsf{OQ}} w' \wedge \mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$	$\mathcal{H}(f) \sqsubseteq_{\mathsf{OQ}} w'$

Fig. 12. $\sqsubseteq_c^{\mathbf{x}}$ and $\sqsubseteq_f^{\mathbf{x}}$

worlds is required to progress with respect to $\sqsubseteq_{\mathsf{OQ}}$ and $\sqsubseteq_{\mathsf{OA}}$. In this spirit, $\mathcal{V}_{\mathcal{A}}^{\mathbf{x}}$ and $\mathcal{K}_{\mathcal{A}}^{\mathbf{x}}$ advance the game into multiple futures calculated according to $\sqsubseteq_{\mathsf{OQ}}$ and $\sqsubseteq_{\mathsf{OA}}$ respectively.

Using KNFBs, we can define bisimilarity for HOSC terms.

Definition 25. Two cr-free HOSC terms $\Gamma \vdash M_1, M_2 : \tau$, are $\mathbf{x}$-bisimilar, written $\Gamma \vdash M_1 \equiv^{\mathbf{x}} M_2 : \tau$, if there exists a WTS $\mathcal{A} = (\text{Worlds}, \sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}, \mathcal{I})$, an initial world $w_0 \in \text{Worlds}$ such that $(\emptyset,\emptyset) \in \mathcal{I}(w_0)$, and $\mathcal{A}^{\mathbf{x}}$-KNFB $(R_{\mathcal{V}}, R_{\mathcal{K}}, R_{\mathcal{E}})$ such that, for any Γ-assignment ρ and $c : \tau$, we have $(M_1\{\rho\}, c\ M_2\{\rho\}, c, w_0, \mathcal{H}_0) \in R_{\mathcal{E}}$, where $\mathcal{H}_0 = [\nu(\rho) \mapsto w_0, c \mapsto w_0]$.

In Section VI, we will establish the following theorem.

Theorem (KNFB Full Abstraction)**.** *For any cr-free* HOSC *terms* $\Gamma \vdash M_1, M_2 : \tau$, $\Gamma \vdash M_1 \equiv^{\mathbf{x}} M_2 : \tau$ *iff* $\Gamma \vdash M_1 \cong_{err}^{\mathbf{x}(ciu)} M_2 : \tau$. *Hence, for any cr-free* $\mathbf{x}$*-terms* $\Gamma \vdash M_1, M_2 : \tau$, $\Gamma \vdash M_1 \equiv^{\mathbf{x}} M_2 : \tau$ *iff* $\Gamma \vdash M_1 \cong_{err}^{\mathbf{x}} M_2 : \tau$.

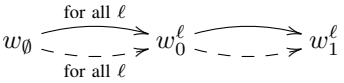
Equivalence proofs based on KNFBs have a compositional flavour in that they proceed by establishing equivalences for pairs of subterms in various worlds, and piecing them together in a way controlled by $\mathcal{A}$.

We shall write $(\mathcal{V}_{\mathcal{A}}^{\mathbf{x}}, \mathcal{K}_{\mathcal{A}}^{\mathbf{x}}, \mathcal{E}_{\mathcal{A}}^{\mathbf{x}})$ to refer to the greatest $\mathcal{A}^{\mathbf{x}}$-KNFB. Below, for simplicity, we ignore types in related tuples.

Example 26 ($\vdash M_1^{isc} \cong_{err}^{\text{HOSC}} M_2^{isc}$ [14])**.**

$$M_1^{isc} \triangleq \text{let x} = \text{ref}\,(0) \text{ in } \lambda \text{f}^{\text{Unit}\to\text{Unit}}.\text{x} := 1; \text{f}(); !\text{x}$$
$$M_2^{isc} \triangleq \lambda \text{f}^{\text{Unit}\to\text{Unit}}.\text{f}(); 1$$

We use the WTS $\mathcal{A}$ shown below, where $\mathcal{I}(w_\emptyset) = \{(\emptyset,\emptyset)\}$, and $\mathcal{I}(w_j^\ell) = \{([\ell \mapsto j], \emptyset)\}$ for $j \in \{0,1\}$. Solid lines indicate $\sqsubseteq_{\mathsf{OA}}$, dashed ones represent $\sqsubseteq_{\mathsf{OQ}}$.



Given c : Int and $\mathcal{H}_0 = [c \mapsto w_\emptyset]$, we aim to show $(M_1^{isc}, c, M_2^{isc}, c, w_\emptyset, \mathcal{H}_0) \in \mathcal{E}_{\mathcal{A}}^{\text{HOSC}}$.

Let $(h_1,h_2) \in \mathcal{I}(w_\emptyset)$. Then $(M_i^{isc}, c, h_i) \to^* (V_i, c, h_i')$, where $V_1 = \lambda f.\ell := 1; f(); !\ell$, $h_1' = [\ell \mapsto 0]$, $V_2 = \lambda f.f(); 1$ and $h_2' = \emptyset$. Noting that $(h_1', h_2') \in \mathcal{I}(w_0^\ell)$ and $(w_\emptyset, \mathcal{H}_0) \sqsubseteq_c^{\text{HOSC}} w_0^\ell$ (i.e. $w_\emptyset \sqsubseteq_{\mathsf{OQ}} w_0^\ell$ and $w_\emptyset \sqsubseteq_{\mathsf{OA}} w_0^\ell$), it suffices to show $(V_1, V_2, w_0^\ell, \mathcal{H}_0) \in \mathcal{V}_{\mathcal{A}}^{\text{HOSC}}$. For this, consider w' with $w_0^\ell \sqsubseteq_{\mathsf{OQ}}^* w'$, i.e. $w' = w_0^\ell$ or $w' = w_1^\ell$. Writing f for A and taking c' : Unit $\to$ Unit, we then need to show $(V_1 f, c', V_2 f, c', w', \mathcal{H}_1) \in \mathcal{E}_{\mathcal{A}}^{\text{HOSC}}$, where $\mathcal{H}_1 = \mathcal{H}_0 \cdot [f, c' \mapsto w']$.

Let $(h_1,h_2) \in \mathcal{I}(w')$. Observe that $(V_i f, c', h_i) \to^* (K_i[f()], c', h_i')$ for $i = 1,2$, where $K_1 = \bullet; !\ell$, $K_2 = \bullet; 1$ and $(h_1', h_2') \in \mathcal{I}(w_1^\ell)$. Noting $(w', \mathcal{H}_1) \sqsubseteq_f^{\text{HOSC}} w_1^\ell$, it suffices to show $((),(), w_1^\ell, \mathcal{H}_1) \in \mathcal{V}_{\mathcal{A}}^{\text{HOSC}}$ and $(K_1, c', K_2, c', w_1^\ell, \mathcal{H}_1) \in \mathcal{K}_{\mathcal{A}}^{\text{HOSC}}$. The former follows directly from the definition.

To show $(K_1, c', K_2, c', w_1^\ell, \mathcal{H}_1) \in \mathcal{K}_{\mathcal{A}}^{\text{HOSC}}$, consider w' with $w_1^\ell \sqsubseteq_{\mathsf{OA}}^* w'$, i.e. $w' = w_1^\ell$. Hence, it suffices to show $(K_1[()], c', K_2[()], c', w_1^\ell, \mathcal{H}_1) \in \mathcal{E}_{\mathcal{A}}^{\text{HOSC}}$. Taking $(h_1,h_2) \in \mathcal{I}(w_1^\ell)$, note that $(K_i[()], c', h_i) \to^* (1, c', h_i)$. Noting $(w_1^\ell, \mathcal{H}_1) \sqsubseteq_{c'}^{\text{HOSC}} w_1^\ell$, we only need to show $(1, 1, w_1^\ell, \mathcal{H}_1) \in \mathcal{V}_{\mathcal{A}}^{\text{HOSC}}$, which follows from the definition.

V. Simplifications and Further Examples

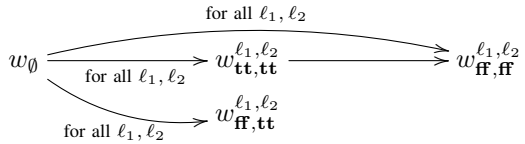
Our KBNF framework does not relate $\sqsubseteq_{\mathsf{OQ}}$ with $\sqsubseteq_{\mathsf{OA}}$ for the sake of maximum generality and with a view to applying the same methodology to other languages. However, for the languages we consider, it is possible to make some simplifying assumptions without losing completeness. For example, for $\mathbf{x} \in \{\text{HOSC}, \text{GOSC}\}$, function and continuation names are propagated in the same way, and $\sqsubseteq_{\mathsf{OQ}}$ and $\sqsubseteq_{\mathsf{OA}}$ can be assumed to coincide. Formally, this will be demonstrated in our completeness arguments. Consequently, in these cases we can restrict the search for world transition systems to those with a single reflexive relation, i.e. $\sqsubseteq_{\mathsf{OQ}} = \sqsubseteq_{\mathsf{OA}}$. In HOS, we will have $\sqsubseteq_{\mathsf{OA}}$ implies $\sqsubseteq_{\mathsf{OQ}}^*$, while in GOS, $\sqsubseteq_{\mathsf{OA}}$ implies $\sqsubseteq_{\mathsf{OQ}}$ (this is related to Remark 14). Under these extra assumptions, the shape of $(w,\mathcal{H}) \sqsubseteq_c^{\text{HOSC}} w'$ and $(w,\mathcal{H}) \sqsubseteq_f^{\text{HOSC}} w'$ from Figure 12 could be simplified as follows.

$\mathbf{x}$	$(w,\mathcal{H}) \sqsubseteq_c^{\mathbf{x}} w'$	$(w,\mathcal{H}) \sqsubseteq_f^{\mathbf{x}} w'$	assumption
HOSC	$w \sqsubseteq_{\mathsf{OA}} w'$	$w \sqsubseteq_{\mathsf{OA}} w'$	$\sqsubseteq_{\mathsf{OA}} = \sqsubseteq_{\mathsf{OQ}}$
GOSC	$\mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$	$\mathcal{H}(f) \sqsubseteq_{\mathsf{OA}} w'$	$\sqsubseteq_{\mathsf{OA}} = \sqsubseteq_{\mathsf{OQ}}$
HOS	$w \sqsubseteq_{\mathsf{OQ}} w', \mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$	$w \sqsubseteq_{\mathsf{OQ}} w'$	$\sqsubseteq_{\mathsf{OA}} \subseteq \sqsubseteq_{\mathsf{OQ}}^*$
GOS	$\mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$	$\mathcal{H}(f) \sqsubseteq_{\mathsf{OQ}} w'$	$\sqsubseteq_{\mathsf{OA}} \subseteq \sqsubseteq_{\mathsf{OQ}}$

We rely on the simplifications in Examples 27, 28, 29. Full proofs for the examples are available in the full version of the paper (along with other examples). Below we only give the associated WTSs and discuss a single representative step in each proof.

Example 27 ($\Gamma \vdash M_1 \cong_{err}^{\text{GOSC}} M_2$ (Example 7))**.** Let $\rho = [f \mapsto f, h \mapsto h]$ be a Γ-assignment and c : Bool. The relevant $\mathcal{A}$ is displayed below, where $\mathcal{I}(w_\emptyset) = \{(\emptyset,\emptyset)\}$ and $\mathcal{I}(w_{b_1,b_2}^{\ell_1,\ell_2}) = \{([\ell_1 \mapsto b_1], [\ell_2 \mapsto b_2])\}$ for $b_1, b_2 \in \{\mathbf{ff}, \mathbf{tt}\}$.

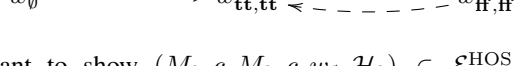
The solid lines represent both $\sqsubseteq_{\mathsf{OA}}$ and $\sqsubseteq_{\mathsf{OQ}}$, i.e. $\sqsubseteq_{\mathsf{OA}}$=$\sqsubseteq_{\mathsf{OQ}}$.



The ultimate objective is to show $(M_1, c, M_2, c, w_\emptyset, \mathcal{H}_0) \in \mathcal{E}^{\mathrm{GOSC}}_{\mathcal{A}}$, where $\mathcal{H}_0 = [f, h, c \mapsto w_\emptyset]$. To this end, among others, one needs to establish $(K'_1[()], c', K'_2[()], c', w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}}, \mathcal{H}_1) \in \mathcal{E}^{\mathrm{GOSC}}_{\mathcal{A}}$ for fresh g, c' and $\mathcal{H}_1 = \mathcal{H}_0 \cdot [g, c' \mapsto w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}]$, where $K'_1 = \bullet; g(); \ell_1 := \mathbf{tt}$, $K'_2 = \bullet; \ell_2 := \mathbf{ff}; g(); \ell_1 := \mathbf{tt}$. This necessitates $(w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}}, \mathcal{H}_1) \sqsubseteq^{\mathrm{GOSC}}_g w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$, which holds, since it boils down to $w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}} \sqsubseteq_{\mathsf{OA}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$, because $\mathcal{H}_1(g) = w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}$.

In contrast, $(w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}}, \mathcal{H}_1) \not\sqsubseteq^{\mathbf{x}}_g w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$ ($\mathbf{x} = \mathrm{HOS}, \mathrm{HOSC}$), since both require $w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}} \sqsubseteq_{\mathsf{OQ}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$, which does not hold in $\mathcal{A}$. Consequently, the argument will not carry over to $\cong^{\mathrm{HOS}}_{err}$ or $\cong^{\mathrm{HOSC}}_{err}$, which is consistent with $\Gamma \vdash M_1 \not\cong^{\mathrm{HOS}}_{err} M_2$.

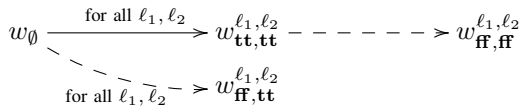
Example 28 ($\Gamma \vdash M_2 \cong^{\mathrm{HOS}}_{err} M_3$ (Example 7))**.** Let $\rho = [f \mapsto f, h \mapsto h]$ be a Γ-assignment and $c : \mathrm{Bool}$. The relevant $\mathcal{A}$ is displayed below, where solid lines represent both $\sqsubseteq_{\mathsf{OA}}$ and $\sqsubseteq_{\mathsf{OQ}}$, and dashed ones - $\sqsubseteq_{\mathsf{OQ}} \setminus \sqsubseteq_{\mathsf{OA}}$, i.e. $\sqsubseteq_{\mathsf{OA}} \subseteq \sqsubseteq_{\mathsf{OQ}}$. The invariants associated with worlds are defined as before.



We want to show $(M_2, c, M_3, c, w_\emptyset, \mathcal{H}_0) \in \mathcal{E}^{\mathrm{HOS}}_{\mathcal{A}}$, where $\mathcal{H}_0 = [f, h, c \mapsto w_\emptyset]$. This leads, among others, to the subgoal $(K'_1[()], c', K'_2[()], c', w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}, \mathcal{H}_1) \in \mathcal{E}^{\mathrm{HOS}}_{\mathcal{A}}$, where $\mathcal{H}_1 = \mathcal{H}_0 \cdot [g, c' \mapsto w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}]$ and $K'_i = \bullet; \ell_i := \mathbf{ff}; g(); \ell_i := \mathbf{tt}$. In particular, one needs to check $(w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}, \mathcal{H}_1) \sqsubseteq^{\mathrm{HOS}}_g w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$, which holds, as it stands for $w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}} \sqsubseteq_{\mathsf{OQ}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$.

However, $(w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}, \mathcal{H}_1) \not\sqsubseteq^{\mathbf{x}}_g w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$ (for $\mathbf{x} = \mathrm{GOSC}, \mathrm{HOSC}$), because both involve $w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}} \sqsubseteq_{\mathsf{OA}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$. Consequently, the argument does not carry over to $\cong^{\mathrm{HOSC}}_{err}$ or $\cong^{\mathrm{GOSC}}_{err}$, which is consistent with $\Gamma \vdash M_2 \not\cong^{\mathrm{GOSC}}_{err} M_3$.

Example 29 ($\Gamma \vdash M_1 \cong^{\mathrm{GOS}}_{err} M_3$ (Example 7))**.** Let $\rho = [f \mapsto f, h \mapsto h]$ be a Γ-assignment and $c : \mathrm{Bool}$. The relevant $\mathcal{A}$ is displayed below, where solid lines represent both $\sqsubseteq_{\mathsf{OA}}$ and $\sqsubseteq_{\mathsf{OQ}}$, and dashed ones - $\sqsubseteq_{\mathsf{OQ}} \setminus \sqsubseteq_{\mathsf{OA}}$, i.e. $\sqsubseteq_{\mathsf{OA}} \subseteq \sqsubseteq_{\mathsf{OQ}}$. The associated heap invariants are the same as before.



We show $(M_1, c, M_3, c, w_\emptyset, \mathcal{H}_0) \in \mathcal{E}^{\mathrm{GOS}}_{\mathcal{A}}$, where $\mathcal{H}_0 = [f, h, c \mapsto w_\emptyset]$. One of the subgoals will be $(K'_1[()], c', K'_2[()], c', w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}}, \mathcal{H}_1) \in \mathcal{E}^{\mathrm{GOS}}_{\mathcal{A}}$ for fresh g, c', where

$$K'_1 = \bullet; g(); \ell_1 := \mathbf{tt},$$
$$K'_2 = \bullet; \ell_2 := \mathbf{ff}; g(); \ell_1 := \mathbf{tt},$$

and $\mathcal{H}_1 = \mathcal{H}_0 \cdot [g, c' \mapsto w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}]$. To this end, one needs to argue that $(w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}}, \mathcal{H}_1) \sqsubseteq^{\mathrm{GOS}}_g w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$, which boils down to $w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}} \sqsubseteq_{\mathsf{OQ}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$, because $\mathcal{H}_1(g) = w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}$.

However, we would not have $(w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}}, \mathcal{H}_1) \sqsubseteq^{\mathbf{x}}_g w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$ ($\mathbf{x} = \mathrm{GOSC}, \mathrm{HOS}$), since they reduce to $w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}} \sqsubseteq_{\mathsf{OQ}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}} \wedge w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}} \not\sqsubseteq_{\mathsf{OA}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$ ($\mathbf{x} = \mathrm{GOSC}$) and $w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{tt}} \sqsubseteq_{\mathsf{OQ}} w^{\ell_1,\ell_2}_{\mathbf{ff},\mathbf{ff}}$ ($\mathbf{x} = \mathrm{HOS}$). Consequently, the argument does not imply $\cong^{\mathrm{GOSC}}_{err}$ or $\cong^{\mathrm{HOS}}_{err}$, which is consistent with $\Gamma \vdash M_1 \not\cong^{\mathbf{x}}_{err} M_3$ ($\mathbf{x} = \mathrm{GOSC}, \mathrm{HOS}$).

VI. Soundness and Completeness

We sketch the proof of our main result.

Theorem 30 (KNFB Full Abstraction)**.** *For any cr-free* HOSC *terms* $\Gamma \vdash M_1, M_2 : \tau$, $\Gamma \vdash M_1 \equiv^{\mathbf{x}} M_2 : \tau$ *iff* $\Gamma \vdash M_1 \cong^{\mathbf{x}(ciu)}_{err} M_2 : \tau$. *Hence, for any cr-free* $\mathbf{x}$*-terms* $\Gamma \vdash M_1, M_2 : \tau$, $\Gamma \vdash M_1 \equiv^{\mathbf{x}} M_2 : \tau$ *iff* $\Gamma \vdash M_1 \cong^{\mathbf{x}}_{err} M_2 : \tau$.

We will rely on Theorem 17 and the fact that trace equivalence coincides with bisimilarity in this case. For the left-to-right direction (soundness), we show how to lift an $\mathcal{A}^{\mathbf{x}}$-KNFB to a bisimulation over $\mathcal{L}_{\mathbf{x}}$. For the converse (completeness), we define an $\mathcal{A}^{\mathbf{x}}$-KNFB from a bisimulation over $\mathcal{L}_{\mathbf{x}}$, where $\mathcal{A}$ will be based on $\mathcal{L}_{\mathbf{x}}$.

Soundness

Let $\mathbf{R} = (R_{\mathcal{V}}, R_{\mathcal{K}}, R_{\mathcal{E}})$ be an $\mathcal{A}^{\mathbf{x}}$-KNFB such that $\mathcal{A} = (\mathrm{Worlds}, \sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}, \mathcal{I})$, $w_0 \in \mathrm{Worlds}$, ρ is a Γ-assignment, $c : \tau$ and $(M_1\{\rho\}, c\ M_2\{\rho\}, c, w_0, \mathcal{H}_0) \in R_{\mathcal{E}}$, where $\mathcal{H}_0 = [\nu(\rho) \mapsto w_0, c \mapsto w_0]$. We will show how to construct a bisimulation over $\mathcal{L}_{\mathbf{x}}$, using $\mathbf{R}$.

First we define bisimulations on $\mathcal{L}_{\mathbf{x}}$ in a more structured way that takes the bipartite nature of $\mathcal{L}_{\mathbf{x}}$ into account. A relation S over configurations is *well-formed* when, for all $(C_1, C_2) \in S$, both configurations are either active or passive, and have the same H_F, H_C, Fn, Cn components.

Definition 31. A (bipartite) bisimulation over $\mathcal{L}_{\mathbf{x}}$ is a pair $\mathbf{S} = (S_{pas}, S_{act})$ of well-formed relations (over passive and active configurations respectively) such that

- if $(C_1, C_2) \in S_{pas}$ then, for all O-moves $\mathbf{o}$, if $C_1 \xrightarrow{\mathbf{o}} C'_1$ and $C_2 \xrightarrow{\mathbf{o}} C'_2$ then $(C'_1, C'_2) \in S_{act}$;
- if $(C_1, C_2) \in S_{act}$ then either $C_1 \Uparrow$ and $C_2 \Uparrow$, or there exists a P-move $\mathbf{p}$ such that $C_1 \overset{\mathbf{p}}{\Longrightarrow} C'_1$ and $C_2 \overset{\mathbf{p}}{\Longrightarrow} C'_2$ and $(C'_1, C'_2) \in S_{pas}$. We write $\overset{\mathbf{a}}{\Longrightarrow}$ for $((\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}})$.

$\mathcal{L}_{\mathbf{x}}$ bisimilarity can be seen as a three-player game (O, P_1, P_2) where O chooses a challenge, represented by $\mathbf{o}$, that both Proponents must either refuse (by diverging) or satisfy (by producing the same P-action).

As a bridge between $\mathcal{A}$-KNFBs and $\mathcal{L}_{\mathbf{x}}$ bisimulations, we will introduce an intermediate notion of bisimilarity, based on incomplete configurations. In particular, it will disentangle the reasoning over the heap resources from other components of the configurations.

A *partial configuration* is a configuration without the heap and the availability record (H_F, H_C, Fn, Cn). It is called a

prime configuration if it is either active and γ is empty, or if it is passive and γ is either empty or singleton. Next we construct an LTS $\twoheadrightarrow$, called the *prime* $\mathcal{L}_{\mathbf{x}}$, in which pairs (P, h), where P is a prime configuration and h is a heap, are reduced using rules analogous to those of $\mathcal{L}_{\mathbf{x}}$, as long as the prime configuration contains enough information to fire the corresponding rule.

Because prime configurations do not record available names, for O-transitions, we simply regard the singleton name in γ as available. The name is then removed from γ after the transition fires, to make the successor (active) prime. This defines $(P, h) \xrightarrow{\mathbf{o}} (P', h)$.

Silent transitions ($P\tau$) are simply inherited: $(P, h) \xrightarrow{\tau} (P', h')$ implies $(P, h) \xrightarrow{\tau} (P', h')$.

As P-transitions may introduce multiple P-names into γ, we split them into several transitions as follows. Recall that $\mathbf{p} = \bar{f}(A, c')$ or $\mathbf{p} = \bar{c}(A)$, where A is an abstract value, i.e. essentially a tuple $\vec{A} = (A_1, \cdots, A_k)$ of atomic values that are not tuples. Abusing notation somewhat, we will refer to this shape generically as $\bar{n}(\vec{A})$, assuming that for $\mathbf{p} = \bar{f}(A, c')$, the name c' is also included in $\vec{A}$, i.e. $A_k = c'$. Then we split the $\xrightarrow{\mathbf{p}}$ transition into $(P, h) \xrightarrow{(j, \bar{n}(A_j))} (P_j, h)$ $(1 \leq j \leq k)$, where the γ component in P_j contains A_j (if it is a name) and is empty if A_j is a constant.

The $\twoheadrightarrow$ LTS corresponds to a Böhm-tree like representation of terms, namely, Lassen's trees introduced in the setting of eager normal-form bisimulations for pure call-by-value λ-calculus [23]. Note that it forces O to explore the names introduced by P in the very next step, like view functions in game semantics [3]. Standard bisimulations over the prime $\mathcal{L}_{\mathbf{x}}$ are *not* sound in the presence of heap resources.

In order to design sound bisimulations based on the prime LTS, we will reason about heaps and the availability components separately, with the help of the WTS $\mathcal{A}$. By a *$\mathcal{A}$-Kripke prime relation* R we mean a relation containing tuples $(P_1, P_2, w, \mathcal{H}_O)$, where P_1, P_2 are prime configurations of the same kind, the domains of γ in P_1, P_2 are the same, $w \in \mathrm{Worlds}$, and $\mathcal{H}_O$ is a world history such that $\mathrm{dom}(\mathcal{H}_O)$ contains all names occurring in the tuple except that in $\mathrm{dom}(\gamma)$.

Definition 32. An *$\mathcal{A}$-Kripke Prime Bisimulation* ($\mathcal{A}^{\mathbf{x}}$-KPB) over $\mathcal{L}_{\mathbf{x}}$ is a pair $\mathbf{R} = (R_{pas}, R_{act})$ of $\mathcal{A}$-Kripke prime relations (over respectively passive and active prime configurations) such that:

- if $(P_1, P_2, w, \mathcal{H}_O) \in R_{pas}$ then for all O-moves $\mathbf{o}$, for all worlds $w' \sqsupseteq^*_{\mathsf{OX}} w$ (with X either Q or A depending on $\mathbf{o}$), for all active prime configurations P_1', P_2', and heaps $(h_1, h_2) \in \mathcal{I}(w')$, if $(P_1, h_1) \xrightarrow{\mathbf{o}} (P_1', h_1)$ and $(P_2, h_2) \xrightarrow{\mathbf{o}} (P_2', h_2)$ then $(P_1', P_2', w', \mathcal{H}_O \cdot [\phi \mapsto w]) \in R_{act}$, where ϕ are the names introduced by $\mathbf{o}$;
- if $(P_1, P_2, w, \mathcal{H}_O) \in R_{act}$ then for all heaps $(h_1, h_2) \in \mathcal{I}(w)$:
  - Either $(P_1, h_1) \Uparrow$ and $(P_2, h_2) \Uparrow$,
  - Or there exists a Player action $\bar{n}(\vec{A})$, a world $w' \sqsupseteq^{\mathbf{x}}_n (w, \mathcal{H}_O)$ and $(h_1', h_2') \in \mathcal{I}(w')$ such that $\vec{A} = (A_1, \cdots, A_k)$ and, for all $1 \leq j \leq k$, $(P_i, h_i) \xRightarrow{(j, \bar{n}(A_j))} (P_i^j, h_i')$ $(i = 1, 2)$, where $(P_1^j, P_2^j, w', \mathcal{H}_O) \in R_{pas}$.

  We write $\xRightarrow{\mathbf{a}}$ for $((\xrightarrow{\tau}\!\!\twoheadrightarrow)^* \xrightarrow{\mathbf{a}}\!\!\twoheadrightarrow)$.

The definition of $\mathcal{A}^{\mathbf{x}}$-KPBs is very similar to $\mathcal{A}^{\mathbf{x}}$-KNFBs. In fact, we can move between them easily, as shown in Figure 13. However, $\mathcal{A}^{\mathbf{x}}$-KPBs are already based on rudimentary configurations, which makes them a better starting point for developing a bisimulation over $\mathcal{L}_{\mathbf{x}}$.

The construction will proceed in two steps, given in Figure 13. First we lift an $\mathcal{A}^{\mathbf{x}}$-KPB R to a relation $R^\dagger$ featuring partial configurations (i.e. still with no heap or name-availability information). In the Figure, we write $\otimes$ to represent, as in [24], a product of configurations corresponding to configuration merging, which is allowed if the argument configurations do not contain the same names in their γ components. So prime configurations are indeed the prime elements of this product. $\bigotimes_j$ corresponds to an arbitrary number of mergers so that we can generate arbitrarily large environments. In R_{pas}, passive prime configurations are being merged. In R_{act}, P and P' are active. Note that the construction creates a new component $\mathcal{H}_P$ at the end of the tuple, to keep track of worlds in which names from the environments γ (i.e. P-names) have been introduced (recall that $\mathcal{H}_O$ tracks O-names).

The second construction $\widehat{R}$ concretises the outcome of the previous construction by providing heaps and availability information in line with the information drawn from $\mathcal{A}$ via $\mathcal{I}$, $\sqsubseteq_{\mathsf{OQ}}$ and $\sqsubseteq_{\mathsf{OA}}$. We write (D, h, H_F, H_C) and (D, h, H_F, H_C, Fn, Cn) to inject the heap h and the availability record into the partial configuration D.

Given $w, \mathcal{H}_O, \mathcal{H}_P$, the corresponding availability components $\mathbb{H}(\mathcal{H}_O, \mathcal{H}_P)$, $\mathbb{A}(w, \mathcal{H}_P)$ are defined to be respectively (H_F, H_C) and (Fn, Cn), as specified below.

$$\begin{array}{rcl} H_F(n) & \triangleq & \{f \mid \mathcal{H}_P(f) \sqsubseteq^*_{\mathsf{OQ}} \mathcal{H}_O(n)\} \\ H_C(n) & \triangleq & \{c \mid \mathcal{H}_P(c) \sqsubseteq^*_{\mathsf{OA}} \mathcal{H}_O(n)\} \\ Fn & \triangleq & \{f \mid \mathcal{H}_P(f) \sqsubseteq^*_{\mathsf{OQ}} w\} \\ Cn & \triangleq & \{c \mid \mathcal{H}_P(c) \sqsubseteq^*_{\mathsf{OA}} w\} \end{array}$$

We then have:

Lemma 33. *If $\mathbf{R}$ is an $\mathcal{A}$-Kripke prime bisimulation over $\mathcal{L}_{\mathbf{x}}$ then $\widehat{\mathbf{R}^\dagger}$ is a bisimulation over $\mathcal{L}_{\mathbf{x}}$.*

Note that the tuple corresponding to $(M_1\{\rho\}, c, M_2\{\rho\}, c, w_0, \mathcal{H}_0) \in R_{\mathcal{E}}$ in the associated $\mathcal{A}^{\mathbf{x}}$-KPB (Figure 13) consists of the initial configurations $\mathsf{C}^{\rho,c}_{M_1}$, $\mathsf{C}^{\rho,c}_{M_2}$ (without the empty heap and information about empty availability). Consequently, by the Lemma above, the configurations will be bisimilar. By Theorem 17, $\Gamma \vdash M_1 \cong^{\mathbf{x}(ciu)}_{err} M_2$.

Completeness

Assuming $\Gamma \vdash M_1 \cong^{\mathbf{x}(ciu)}_{err} M_2$, for any Γ-assignment ρ and suitably typed c, we need to construct an $\mathcal{A}^{\mathbf{x}}$-KNFB, along with an initial world w_0, such that

From $\mathcal{A}^{\mathbf{x}}$-KNFB to $\mathcal{A}^{\mathbf{x}}$-KPB, and back.

$$\begin{array}{lcl}(\sigma, M_1, c_1, M_2, c_2, w, \mathcal{H}_O) & \leftrightarrow & (\langle M_1, c_1, \cdot, \mathrm{dom}(\mathcal{H}_O)\rangle, \langle M_2, c_2, \cdot, \mathrm{dom}(\mathcal{H}_O)\rangle, (w, \mathcal{H}_O)) \\ (\sigma, \sigma', K_1, c_1, K_2, c_2, w, \mathcal{H}_O) & \leftrightarrow & (\langle [c \mapsto (K_1, c_1)], \mathrm{dom}(\mathcal{H}_O) \uplus \{c\}\rangle, \langle [c \mapsto (K_2, c_2)], \mathrm{dom}(\mathcal{H}_O) \uplus \{c\}\rangle, (w, \mathcal{H}_O)) \\ (\sigma \to \sigma', V_1, V_2, w, \mathcal{H}_O) & \leftrightarrow & (\langle [f \mapsto V_1], \mathrm{dom}(\mathcal{H}_O) \uplus \{f\}\rangle, \langle [f \mapsto V_2], \mathrm{dom}(\mathcal{H}_O) \uplus \{f\}\rangle, (w, \mathcal{H}_O))\end{array}$$

Step 1: Given $\mathbf{R} = (R_{pas}, R_{act})$, we set $\mathbf{R}^\dagger = (R^\dagger_{pas}, R^\dagger_{act})$. Below n_j is the unique name in the γ component of P^j_1 and P^j_2.

$$\begin{array}{rcl}R^\dagger_{pas} & \triangleq & \{(\bigotimes_j P^j_1, \bigotimes_j P^j_2, \mathcal{H}_O, \bigcup_j [n_j \mapsto w_j]) \mid (P^j_1, P^j_2, w_j, \mathcal{H}_O) \in R_{pas},\ j \in J,\ J \text{ finite}\} \\ R^\dagger_{act} & \triangleq & \{(P \otimes D, P' \otimes D', w, \mathcal{H}_O, \mathcal{H}_P) \mid (P, P', w, \mathcal{H}_O) \in R_{act} \wedge (D, D', \mathcal{H}_O, \mathcal{H}_P) \in R^\dagger_{pas}\}\end{array}$$

Step 2: Given $\mathbf{R} = (R_{pas}, R_{act})$, we set $\widehat{\mathbf{R}} = (\widehat{R_{pas}}, \widehat{R_{act}})$.

$$\begin{array}{rcl}\widehat{R_{pas}} & \triangleq & \{((D_1, h_1, H_F, H_C, Fn, Cn), (D_2, h_2, H_F, H_C, Fn, Cn)) \mid \exists (w, \mathcal{H}_O, \mathcal{H}_P). \\ & & \quad (h_1, h_2) \in \mathcal{I}(w) \wedge (H_F, H_C) = \mathbb{H}(\mathcal{H}_O, \mathcal{H}_P) \wedge (Fn, Cn) = \mathbb{A}(w, \mathcal{H}_P) \wedge (D_1, D_2, \mathcal{H}_O, \mathcal{H}_P) \in R_{pas}\} \\ \widehat{R_{act}} & \triangleq & \{((D_1, h_1, H_F, H_C), (D_2, h_2, H_F, H_C)) \mid \exists (w, \mathcal{H}_P, \mathcal{H}_O). \\ & & \quad (H_F, H_C) = \mathbb{H}(\mathcal{H}_O, \mathcal{H}_P) \wedge (h_1, h_2) \in \mathcal{I}(w) \wedge (D_1, D_2, w, \mathcal{H}_O, \mathcal{H}_P) \in R_{act}\}\end{array}$$

Fig. 13. Lifting steps

$(M_1\{\rho\}, c, M_2\{\rho\}, c, w_0, \mathcal{H}_0) \in R_{\mathcal{E}}$, where $\mathcal{H}_0 = [\nu(\rho), c \mapsto w_0]$. Using the correspondence from Figure 13, it suffices to construct the corresponding $\mathcal{A}^{\mathbf{x}}$-KPB. Next we sketch several crucial steps in the argument.

The first step extracts a bisimulation $\mathbf{S} = (S_{pas}, S_{act})$ over $\mathcal{L}_{\mathbf{x}}$ via Theorem 17. We use the bisimulation to specify the WTS $\mathcal{A}$ as $(\mathrm{Worlds}, \sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}, \mathcal{I})$, where $\mathrm{Worlds} = S_{pas}$ (pairs of passive configurations) and $\mathcal{I}$ maps a pair of configurations to the pair of their heaps. To define $\sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}$, we rely on several new relations.

- $C_1 \sqsubseteq_{op} C_2$ if $C_1 = C_2$ or $C_1 \stackrel{\mathbf{op}}{\Longrightarrow} C_2$ for some $\mathbf{o}, \mathbf{p}$.
- $C_1 \sqsubseteq_{j} C_2$ if $C_1 = C_2$ or $C_1 \stackrel{\mathbf{o}t\mathbf{p}}{\Longrightarrow} C_2$, where $\mathbf{p}$ is justified by $\mathbf{o}$ (i.e. $\mathbf{p}$'s head name is introduced in $\mathbf{o}$).
- $C_1 \sqsubseteq_{jQA} C_2$ if $C_1 = C_2$ or $C_1 \stackrel{\mathbf{o}t\mathbf{p}}{\Longrightarrow} C_2$, where $\mathbf{p}$ is an answer justified by $\mathbf{o}$, i.e. $\sqsubseteq_{jQA}$ is a subset of $\sqsubseteq_{j}$.

Note that $\sqsubseteq_j$ is like calculating $\mathsf{Vis}_O(t)$, while $\sqsubseteq_{jQA}$ mimics $\mathsf{Top}_O(t)$. An important point to note is that the definitions given below will be an exact match with how name availability is calculated in each case (Figure 10). We define $\sqsubseteq_{\mathsf{OQ}}, \sqsubseteq_{\mathsf{OA}}$ as follows.

$\mathbf{x}$	$\sqsubseteq_{\mathsf{OQ}}$	$\sqsubseteq_{\mathsf{OA}}$
HOSC	$\sqsubseteq_{op}$	$\sqsubseteq_{op}$
GOSC	$\sqsubseteq_{j}$	$\sqsubseteq_{j}$
HOS	$\sqsubseteq_{op}$	$\sqsubseteq_{jQA}$
GOS	$\sqsubseteq_{j}$	$\sqsubseteq_{jQA}$

Finally, we construct an $\mathcal{A}^{\mathbf{x}}$-KPB $\mathbf{S}^\uparrow = (S^\uparrow_{pas}, S^\uparrow_{act})$ using tuples of the form $(P_1, P_2, ((C_1, C_2), \mathcal{H}_O))$, which must be "consistent" with $\mathbf{S}$, e.g. $(C_1, C_2) \in S_{pas}$, P_i is a prime component of C_i (for $S^\uparrow_{pas}$) and P_i is a prime component of C_i's successor (for $S^\uparrow_{act}$).

VII. Related work and conclusions

Kripke logical relational frameworks [15], [16] typically operate with a single notion of future. However, to capture scenarios without control, [16] proposed to make a distinction between $\sqsubseteq$ (private) and $\sqsubseteq_{\mathrm{pub}}$ (public), with the intuition that contexts make public transitions only. This is similar to our $\sqsubseteq_{\mathsf{OQ}}$ vs $\sqsubseteq_{\mathsf{OA}}$ distinction, though our perspective is motivated by distinguishing questions and answers, and the two relations cover (sequences of) pairs of OP moves. In our setting, the requirement that the "end-to-end" behaviour must appear public corresponds to the $\mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$ condition.

When proving equivalences without higher-order state (i.e. GOSC and GOS), the authors of [16] observed that, in the course of their proof, it was still sound not to follow the future relation, and to go back in time before moving into the future again. The technique, called *backtracking*, has been described somewhat informally. The authors explain how far one can backtrack as follows: when proving that functions are related at a starting state s, "we can transition from any state accessible from s to any other state accessible from s". This backtracking policy is sufficient to prove the examples handled in [16], but our analysis (namely, the $\mathcal{H}(n) \sqsubseteq_{\mathsf{OA}} w'$, $\mathcal{H}(n) \sqsubseteq_{\mathsf{OQ}} w'$ conditions in various cases) indicates that, to make the technique complete on its own, it is necessary to allow jumps to the point of introduction of the relevant continuation name (for values) or function names (for callbacks). This point of introduction may well precede the point at which we are showing functions equivalent, e.g. in Example 27, the equivalence argument for the $\lambda g.\cdots$ terms at $w^{\ell_1,\ell_2}_{\mathbf{tt},\mathbf{tt}}$ needs to jump to $w_\emptyset$ in connection with the callback $h()$.

We believe that a general notion of backtracking in KLRs [16] can be developed by unfolding the bi-orthogonal definitions, akin to the *Principle of Local Invariants* of [14]. However, its formal statement would most likely require the introduction of a notion of history, similar to the one we have introduced, i.e. the origin of each function or continuation provided by the context has to be tracked. It is interesting to note that the $\mathcal{H}(c) \sqsubseteq_{\mathsf{OA}} w'$ condition for HOS looks like backtracking, even though it concerns higher-order state.

In [25], Relational Transition Systems were introduced as a synthesis of bisimulations and Kripke-style reasoning

(for a programming language with references and polymorphism), and later refined to Parametric Inter-Language Bisimulations [26]. They are based on a notion of global knowledge that seeks to generalise notions such as the environment γ, e.g. in the setting of inter-language reasoning. Our approach has a rather different technical objective: we seek to remove the need for such global knowledge in order to obtain a modular and complete technique for languages with both references and continuations.

The worlds used in [16], are divided into *islands* that carry a transition system of invariants on disjoint parts of the heap. A future world is then able to add new islands, i.e. also new transition systems, to the current world. Our setting is somewhat different: in a WTS, the future relation corresponds to navigating a transition system that is fixed upfront and not modified later. While we have not designed reasoning principles for combining reasoning on components that handle disjoint parts of the heap, it should be possible to do so by extending the tensor product of partial configurations presented in Section VI to configurations with disjoints heaps.

As we briefly mentioned, for contexts without control, $\cong^{\mathbf{x}}_{err}$ is stronger than $\cong^{\mathbf{x}}_{ter}$. In [7], it was shown how to capture $\cong^{\mathbf{x}}_{ter}$ using complete traces. Consequently, equivalence then boils down to bisimilarity testing over complete traces. To adapt our methodology to such traces, it is necessary to detect configurations that will never run to completion (termination). This is largely an orthogonal concern to capturing the shape of potential interactions in each case. It can be handled by introducing a special class of "inconsistent" worlds that guarantee non-termination in any future, as in [16], [19], [9].

Our methodology is general and we would like to extend it to further paradigms. To go beyond cr-free HOSC terms, i.e. to allow reference- and continuation-types at the term interface, it should be sufficient to maintain bijections of names inside worlds, as introduced in [27]. For polymorphic types, a natural starting point would be the operational game semantics provided in [28] for a language with references and parametric polymorphism.

We believe our work opens up the way to automated reasoning about contextual equivalence for all four languages in a common framework, following the approach proposed in [18].

Acknowledgment

We are very grateful to the anonymous referees for helpful comments and for pointing out a missing side condition ("for all $\mathbf{x}$-terms") in Lemma 1 of [7]. We discuss the issue in more detail in the full version of the ESOP paper.

References

[1] R. Milner, "Fully abstract models of typed lambda-calculi," *Theoretical Computer Science*, vol. 4, no. 1, pp. 1–22, 1977.

[2] S. Abramsky, R. Jagadeesan, and P. Malacaria, "Full abstraction for PCF," *Information and Computation*, vol. 163, pp. 409–470, 2000.

[3] J. M. E. Hyland and C.-H. L. Ong, "On Full Abstraction for PCF: I. Models, observables and the full abstraction problem, II. Dialogue games and innocent strategies, III. A fully abstract and universal game model," *Information and Computation*, vol. 163(2), pp. 285–408, 2000.

[4] A. Jeffrey and J. Rathke, "A fully abstract may testing semantics for concurrent objects," *Theor. Comput. Sci.*, vol. 338, no. 1-3, pp. 17–63, 2005.

[5] R. Jagadeesan, C. Pitcher, and J. Riely, "Open bisimulation for aspects," in *Proceedings of AOSD*, ser. ACM International Conference Proceeding Series, vol. 208, 2007, pp. 107–120.

[6] J. Laird, "A fully abstract trace semantics for general references," in *Proceedings of ICALP*, ser. Lecture Notes in Computer Science. Springer, 2007, vol. 4596, pp. 667–679.

[7] G. Jaber and A. S. Murawski, "Complete trace models of state and control," in *Proceedings of ESOP*, ser. Lecture Notes in Computer Science, vol. 12648. Springer, 2021, pp. 348–374, full version available at https://hal.archives-ouvertes.fr/hal-03116698.

[8] K. Støvring and S. B. Lassen, "A complete, co-inductive syntactic theory of sequential control and state," in *POPL*. ACM, 2007, pp. 161–172.

[9] D. Biernacki, S. Lenglet, and P. Polesiuk, "A complete normal-form bisimilarity for state," in *Proceedings of FOSSACS*, ser. Lecture Notes in Computer Science, vol. 11425. Springer, 2019, pp. 98–114.

[10] E. Sumii, "A complete characterization of observational equivalence in polymorphic *lambda*-calculus with general references," in *Proceedings of CSL*, ser. Lecture Notes in Computer Science, vol. 5771. Springer, 2009, pp. 455–469.

[11] D. Sangiorgi, N. Kobayashi, and E. Sumii, "Environmental bisimulations for higher-order languages," *ACM Trans. Program. Lang. Syst.*, vol. 33, no. 1, p. 5, 2011.

[12] V. Koutavas and M. Wand, "Small bisimulations for reasoning about higher-order imperative programs," in *Proceedings of POPL*. ACM, 2006, pp. 141–152.

[13] V. Koutavas, P. B. Levy, and E. Sumii, "From applicative to environmental bisimulation," *Electr. Notes Theor. Comput. Sci.*, vol. 276, pp. 215–235, 2011.

[14] A. M. Pitts and I. D. B. Stark, "Operational reasoning for functions with local state," in *Higher-Order Operational Techniques in Semantics*, A. D. Gordon and A. M. Pitts, Eds. Cambridge University Press, 1998, pp. 227–273.

[15] A. Ahmed, D. Dreyer, and A. Rossberg, "State-dependent representation independence," in *Proceedings of POPL*. ACM, 2009, pp. 340–353.

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

[17] D. Sangiorgi, "A theory of bisimulation for the pi-calculus," *Acta Inf.*, vol. 33, no. 1, pp. 69–97, 1996.

[18] G. Jaber, "SyTeCi: Automating contextual equivalence for higher-order programs with references," *Proc. ACM Program. Lang.*, vol. 4, no. POPL, 2020.

[19] G. Jaber and N. Tabareau, "Kripke open bisimulation - A marriage of game semantics and operational techniques," in *Proceedings of APLAS*, ser. Lecture Notes in Computer Science, vol. 9458, 2015, pp. 271–291.

[20] J. Laird, "A semantic analysis of control," Ph.D. dissertation, University of Edinburgh, 1998.

[21] S. Abramsky, K. Honda, and G. McCusker, "Fully abstract game semantics for general references," in *Proceedings of LICS*. Computer Society Press, 1998, pp. 334–344.

[22] S. B. Lassen and P. B. Levy, "Typed normal form bisimulation," in *Proceedings of CSL*, ser. Lecture Notes in Computer Science. Springer, 2007, vol. 4646, pp. 283–297.

[23] S. B. Lassen, "Eager normal form bisimulation," in *Proceedings of LICS*, 2005, pp. 345–354.

[24] P. B. Levy and S. Staton, "Transition systems over games," in *Proceedings of CSL-LICS*, 2014, pp. 64:1–64:10.

[25] C.-K. Hur, D. Dreyer, G. Neis, and V. Vafeiadis, "The marriage of bisimulations and Kripke logical relations," in *Proceedings of POPL*. ACM, 2012, pp. 59–72.

[26] G. Neis, C.-K. Hur, J.-O. Kaiser, C. McLaughlin, D. Dreyer, and V. Vafeiadis, "Pilsner: A compositionally verified compiler for a higher-order imperative language," *SIGPLAN Not.*, vol. 50, no. 9, p. 166?178, Aug. 2015.

[27] N. Benton and B. Leperchey, "Relational reasoning in a nominal semantics for storage," in *International Conference on Typed Lambda Calculi and Applications*. Springer, 2005, pp. 86–101.

[28] G. Jaber and N. Tzevelekos, "Trace semantics for polymorphic references," in *Proceedings of LICS*. ACM, 2016, pp. 585–594.