Reasoning about Local Variables with Operationally-Based Logical Relations

Andrew M. Pitts*
Cambridge University Computer Laboratory, Pembroke Street, Cambridge CB2 3QG, UK
ap@cl.cam.ac.uk

Abstract

A parametric logical relation between the phrases of an Algol-like language is presented. Its definition involves the structural operational semantics of the language, but was inspired by recent denotationally-based work of O'Hearn and Reynolds on translating Algol into a predicatively polymorphic linear lambda calculus. The logical relation yields an applicative characterisation of contextual equivalence for the language and provides a useful (and complete) method for proving equivalences. Its utility is illustrated by giving simple and direct proofs of some contextual equivalences, including an interesting equivalence due to O'Hearn which hinges upon the undefinability of 'snapback' operations (and which goes beyond the standard suite of 'Meyer-Sieber' examples). Whilst some of the mathematical intricacies of denotational semantics are avoided, the hard work in this operational approach lies in establishing the 'fundamental property' for the logical relation—the proof of which makes use of a compactness property of fixpoint recursion with respect to evaluation of phrases. But once this property has been established, the logical relation provides a verification method with an attractively low mathematical overhead.

1. Introduction

The observable properties of sequentially executed imperative programs involving higher order procedures and locally declared state can be quite subtle. This is so even when the use of local state is quite severely constrained, as it is in languages which are *Algol-like* in the sense of Reynolds [17]—i.e. when the state just consists of variables storing first order values (as opposed to function closures, for example), local variable declarations are only permitted in commands (not in expressions), are statically scoped and are executed using a stack discipline. The subtleties of the externally observable behaviour of such locally declared state are such that, despite the considerable efforts of a number of researchers [5, 11, 10, 16, 8, 20], no concrete denotational model of Algol has yet been constructed which exactly captures observational equivalence for n-th order procedures beyond $n = 2$ or 3 (depending upon how one counts orders).

Nevertheless, useful semantical ideas and techniques have emerged from the effort to construct such 'fully abstract' models. The one that we focus on here concerns the *parametric logical relations* occurring in the work of O'Hearn-Tennent [10], Sieber [20, 21], and O'Hearn-Reynolds [9]. Such relations are used for two intertwined purposes. First, they can be used to identify 'junk' in a model—elements that cannot possibly be meanings of program phrases. Secondly, they can be used to prove that two program phrases are observationally equivalent. The work described in this paper was motivated by the desire to extract the essence of this second aspect of the use of logical relations from the mathematical structures used to model an Algol-like language, and to apply that essence directly to the language itself equipped with the commonly used notion of operational equivalence—a Morris-style contextual equivalence.

Recall that two program phrases M_1 and M_2 are *contextually equivalent*, written $M_1 \cong M_2$, if occurrences of the phrases in any complete program can be interchanged without affecting the observable effects of executing the program. More precisely, $M_1 \cong M_2$ holds if for all program contexts $P[-]$, the programs $P[M_1]$ and $P[M_2]$ have equal observable effects. Although this is a reasonable notion of semantic equality for program phrases, the quantification over all contexts $P[-]$ which occurs in its definition makes it hard to work with directly—because the ways in which a context can make use of its 'hole' can be complicated. One therefore seeks to develop general properties of contextual equivalence. For example, one might hope to establish the familiar functional extensionality property for contextual equivalence

$$F \cong F' : \sigma \to \sigma' \Leftrightarrow \forall A : \sigma \, (F\,A \cong F'\,A : \sigma'). \qquad (1)$$

This serves to reduce contextual equivalence at the function type $\sigma \to \sigma'$ to that at the structurally simpler type σ'. In

*Research partially supported by the EU HCM Research Network on 'Lambda Calcul Typé'.

languages like Scheme or ML this kind of functional extensionality fails, due to the complicated interactions possible between call-by-value function application and locally declared state in expressions of function type, permitting 'leakage' of private names out of the textual scope of local declarations. See Pitts and Stark [15, 23] for examples. By contrast, Algol-like languages, by virtue of having call-by-name function application and having local variable declarations restricted to commands, do satisfy (1). This is part of the Algol *Operational Extensionality Theorem* which we shall prove (Theorem 2.5). It is a generalisation to Algol of Milner's Context Lemma [6] for the purely functional language PCF. The properties of Algol contextual equivalence it expresses will come as no surprise to a connoisseur of the language, but the author was unable to locate any formal statement or proof of these properties in the literature—mainly because most existing work concerns itself solely with denotational semantics of Algol.

There are several ways to prove the Operational Extensionality Theorem. For example, one can deduce it via an extension to imperative languages of the methods developed by Howe [2, 3] for pure functional languages (*cf.* [18]). Here we deduce it from the existence of a certain kind of logical relation, whose definition and properties are the main technical contribution of this paper. This takes the form of a parametric family of relations between closed Algol terms of equal type. Roughly speaking, the parameter ranges over relations between states (which in this case are just assignments of integer values to global variables). The definition of the logical relation proceeds by induction on the structure of types and involves the structural operational semantics of the language. Each clause embodies both extensionality properties and the kind of relational parametricity considered in a denotational setting by O'Hearn and Tennent [10]. The Operational Extensionality Theorem follows from the fact that contextual equivalence coincides with the parametric logical relation when its parameter is instantiated to the identity relation on states. This fact is in turn derived from the logical relation's 'Fundamental Property' (*cf.* [7]), namely that it is preserved by the various term-forming operations of Algol. The proof of the Fundamental Property (Theorem 3.9) is non-trivial because of the presence of recursively defined terms in the language. At this point in a denotational development one can use the familiar characterisation of least fixed points (used to model recursive terms) as least upper bounds of certain ascending chains and apply Scott Induction [19, section 3]. Here we develop an operational analogue of this method. This is similar to the approach taken by Smith *et al* [4], except that we have to deal with an imperative language and we use a structural operational semantics based upon an evaluation (or 'big-step') relation, rather than a transition (or 'small-step') relation. At the heart of the proof is a certain 'compactness' property of evaluation with respect to the canonical sequence of approximations to a recursive term (see the proof of Proposition 3.7).

The pay-off from the development of the operationally-based parametric logical relation is not only a proof of the extensionality properties of Algol contextual equivalence, but also a useful and mathematically lightweight tool for proving particular equivalences. We demonstrate this by example. As well as dealing with the well-known Meyer-Sieber examples [5], we prove a rather subtle equivalence due to O'Hearn (Example 4.1). It illustrates the consequences for contextual equivalence of the inability of the particular Algol-like language under consideration to 'snap-back' the state to some previous point in the thread of computation. We said above that, roughly speaking, the logical relation is parameterized by relations between states. More precisely, the parameter is a binary relation on the flat cpo of states—in other words some states get related to a formal, undefined state. This level of generality is not needed to establish the operational Extensionality Theorem—binary relations on the set of states would be enough (see Remark 3.11). Rather, the generalisation involving undefined states is needed to cope with O'Hearn's example, and was adopted from the recent denotationally-based work of O'Hearn and Reynolds [9]. The precise definition of the logical relation and the extensionality results are given in Section 3, and the application to proving equivalences is given in Section 4. In the final section we discuss some related, operationally-based work and further directions of research.

2. Idealised Algol

We will define the parametric logical relation for Idealised Algol, IA, a small Algol-like language which has been used by several authors for illustrative purposes. It is a simply typed lambda calculus over ground types $bool$ (booleans), int (integers), var (variables for storing integers), and cmd (commands for changing state). It contains terms (of the appropriate types) for lambda abstraction ($\lambda x{:}\sigma\,.\,M$), application ($F\,A$), conditionals ($\mathbf{if}\ B\ \mathbf{then}\ M_1\ \mathbf{else}\ M_2$), fixpoint recursion ($\mathbf{fix}\,x{:}\sigma\,.\,M$), integer and boolean constants ($\mathbf{n}, \mathbf{b}$), and some arithmetic operations and relations ($N_1 * N_2$). We choose to make dereferencing explicit in the syntax: $!V$ is the IA term of type int denoting the contents of a term V of type var. In addition to denumerable sets of identifiers of each type (x^σ), there is a denumerable set of *global variables* ($\mathbf{v}$), each of which is an IA term of type var. Finally, IA contains cmd-building constructs for no-op ($\mathbf{skip}$), integer assignment ($V := N$), sequencing ($C_1\,;C_2$), and local variable blocks ($\mathbf{new}\ x := N\ \mathbf{in}\ C\ \mathbf{end}$), in addition to the conditional and fixpoint constructs which are available at all types (and which permit one to define various recursive control structures, such as $\mathbf{while}\ -\ \mathbf{do}\ -$). We write $M : \sigma$ to indicate that the IA term M has type σ.

2.1. Remark (Binding and substitution). The IA terms for local variable blocks, lambda abstraction and fixpoint recursion are all identifier-binding constructs: free occurrences of x^{var} in the command C become bound in $\mathbf{new}\ x := N\ \mathbf{in}\ C\ \mathbf{end}$; and free occurrences of x^{σ} in the term M become bound in $\lambda x{:}\sigma\,.\,M$ and in $\mathbf{fix}\,x{:}\sigma\,.\,M$. Henceforward *we will identify IA terms up to α-conversion of their bound identifiers*. Then $M[M'/x^{\sigma}]$ will denote the result (well-defined up to α-equivalence) of substituting a term M' of type σ for all free occurrences of the identifier x^{σ} in the term M. Similarly, $M[\mathbf{v}'/\mathbf{v}]$ will denote the result of substituting the global variable $\mathbf{v}'$ for all occurrences of the global variable $\mathbf{v}$ in M. We write $\mathit{fi}(M)$ for the finite set of free identifiers of M, and $\mathit{gv}(M)$ for its finite set of global variables. Let

$$IA_{\sigma}(w) \stackrel{\text{def}}{=} \{M : \sigma \mid \mathit{fi}(M) = \emptyset \ \&\ \mathit{gv}(M) \subseteq w\}$$

denote the set of closed IA terms of type σ with global variables in the set w.

We specify the operational semantics of IA in terms of an inductively defined evaluation relation of the form

$$w \vdash s; M \Downarrow_{\sigma} s'; R. \tag{2}$$

Here w is a finite set of global variables—we call such sets *worlds*, because they are an operational trace of the Kripke-style 'possible world' semantics of block structure using functor categories introduced by Reynolds and Oles [11]. In (2), M and R are elements of $IA_{\sigma}(w)$, and s, s' are *states* of world w—i.e. functions assigning integers to the global variables in w. We write $\mathit{States}(w)$ for the set of all such states. The intended meaning of (2) is that given the initial assignment s of values to the relevant global variables, evaluation of M yields the final result R and state s'. The rules for inductively generating the evaluation relation are given in Figure 1. They are quite standard, apart from some notational choices: $w\mathbf{v}$ denotes the set w augmented by a new element $\mathbf{v} \notin w$, and then $s \otimes \mathbf{v} := \mathbf{n} \in \mathit{States}(w\mathbf{v})$ denotes the state properly extending the function s by mapping $\mathbf{v}$ to the integer $\mathbf{n}$. On the other hand, if $\mathbf{v} \in w$ and $s \in \mathit{States}(w)$, then $s; \mathbf{v} := \mathbf{n} \in \mathit{States}(w)$ denotes the state mapping $\mathbf{v}$ to $\mathbf{n}$ and otherwise acting like s.

The rules in Figure 1 appear more general than they really are when it comes to sequential state change, because IA evaluation does have the familiar and desirable property of an Algol-like language that *evaluation of terms of non-command type is state dependent, but not state changing*. This is the first of a number of important, but quite straightforward properties of the evaluation relation given by the following lemma. Each property can easily be proved by induction on the derivation of evaluations from the rules in Figure 1.

2.2. Lemma. (Side-effect free expressions) *If* (2) *holds and $\sigma \neq cmd$, then $s = s'$.*

(Equivariance). *Given a bijection $\pi : w \cong w'$, write $M[\pi]$ for the result of replacing each $\mathbf{v} \in w$ by $\pi(\mathbf{v})$ in M; and for each state $s \in \mathit{States}(w)$, write $s[\pi]$ for the element of $\mathit{States}(w')$ which maps each $\mathbf{v}' \in w'$ to $s(\pi^{-1}(\mathbf{v}'))$. If* (2) *holds, then $w' \vdash s[\pi]; M[\pi] \Downarrow_{\sigma} s'[\pi]; R[\pi]$.*

(Determinacy) *If $w \vdash s; M \Downarrow_{\sigma} s'_i; R_i$ holds for $i = 1, 2$, then $R_1 = R_2$ and $s'_1 = s'_2$. (Recall that we are identifying IA terms up to α-equivalence.)*

(Weakening and Strengthening) *Suppose that $w = w_1 \cup w_2$ with $w_1 \cap w_2 = \emptyset$, and that $s_i \in \mathit{States}(w_i)$ for $i = 1, 2$. Given $M \in IA_{\sigma}(w_1)$, then $w \vdash s_1 \otimes s_2; M \Downarrow_{\sigma} s'; R$ holds if and only if $R \in IA_{\sigma}(w_1)$ and $s' = s'_1 \otimes s_2$ for some s'_1 with $w_1 \vdash s_1; M \Downarrow_{\sigma} s'_1; R$.*

In view of the first part of the lemma, when $\sigma \neq cmd$ we abbreviate (2) to

$$w \vdash s; M \Downarrow_{\sigma} R. \tag{3}$$

Similarly, since the only result term of type cmd is $\mathbf{skip}$, when $\sigma = cmd$ we abbreviate (2) to

$$w \vdash s; M \Downarrow s'. \tag{4}$$

Finally, we write

$$w \vdash s; M \Uparrow_{\sigma} \tag{5}$$

to indicate that $w \vdash s; M \Downarrow_{\sigma} s'; R$ does not hold for any s' and R.

Having fixed the syntax and operational semantics of our Algol-like language, we can give the formal definition of contextual equivalence. As usual, a *context* $C[-_{\sigma}]$ is a term in which a subexpression of type σ has been replaced by a 'hole', $-_{\sigma}$. The expression resulting from filling the hole with an expression $M : \sigma$ will be denoted by $C[M]$. Since $-_{\sigma}$ may occur within the scope of identifier-binding constructs, free identifiers of M may become bound in $C[M]$. We write $\mathit{traps}(C[-_{\sigma}])$ for the set of identifiers that occur in $C[-_{\sigma}]$ associated to binders containing the hole $-_{\sigma}$ within their scope. This 'capture' of identifiers in $\mathit{fi}(M) \cap \mathit{traps}(C[-_{\sigma}])$ means that although the operation of substituting M for $-_{\sigma}$ in $C[-_{\sigma}]$ respects α-conversion of bound identifiers in M, it does not necessarily respect α-conversion of bound identifiers in $C[-_{\sigma}]$. Therefore we do not identify contexts up to α-conversion. As for terms, so for contexts we write $C[-_{\sigma}] : \sigma'$ to indicate that σ' is the type of the context; $\mathit{fi}(C[-_{\sigma}])$ and $\mathit{gv}(C[-_{\sigma}])$ denote the finite sets of free identifiers and global variables of the context. A *closed* context is one with no free identifiers.

$$w \vdash s; R \Downarrow_\sigma s; R \quad (\text{if } R ::= \mathbf{b} \mid \mathbf{n} \mid \mathbf{v} \mid \mathbf{skip} \mid \lambda x\,.\,M) \qquad \frac{w \vdash s; F \Downarrow_{\sigma\to\sigma'} s'; \lambda x{:}\sigma\,.\,M \quad w \vdash s'; M[A/x^\sigma] \Downarrow_{\sigma'} s''; R}{w \vdash s; (F\,A) \Downarrow_{\sigma'} s''; R}$$

$$\frac{w \vdash s; B \Downarrow_{bool} s'; \mathbf{b} \qquad w \vdash s'; M_{\mathbf{b}} \Downarrow_\sigma s''; R}{w \vdash s; (\mathbf{if}\ B\ \mathbf{then}\ M_{\mathbf{true}}\ \mathbf{else}\ M_{\mathbf{false}}) \Downarrow_\sigma s''; R} \qquad \frac{w \vdash s; M[\mathbf{fix}\,x{:}\sigma\,.\,M/x^\sigma] \Downarrow_\sigma s'; R}{w \vdash s; \mathbf{fix}\,x{:}\sigma\,.\,M \Downarrow_\sigma s'; R}$$

$$\frac{w \vdash s; N_1 \Downarrow_{int} s'; \mathbf{n}_1 \qquad w \vdash s'; N_2 \Downarrow_{int} s''; \mathbf{n}_2}{w \vdash s; (N_1 * N_2) \Downarrow_{type(*)} s''; \mathbf{c}}\ (\text{if } \mathbf{c} = \mathbf{n}_1 * \mathbf{n}_2) \qquad \frac{w \vdash s; V \Downarrow_{var} s'; \mathbf{v}}{w \vdash s; !V \Downarrow_{int} s'; \mathbf{n}}\ (\text{if } \mathbf{n} = s'(\mathbf{v}))$$

$$\frac{w \vdash s; V \Downarrow_{var} s'; \mathbf{v} \qquad w \vdash s'; N \Downarrow_{int} s''; \mathbf{n}}{w \vdash s; V := N \Downarrow_{cmd} (s''; \mathbf{v} := \mathbf{n}); \mathbf{skip}} \qquad \frac{w \vdash s; C_1 \Downarrow_{cmd} s'; \mathbf{skip} \qquad w \vdash s'; C_2 \Downarrow_{cmd} s''; \mathbf{skip}}{w \vdash s; (C_1\ ;\ C_2) \Downarrow_{cmd} s''; \mathbf{skip}}$$

$$\frac{w \vdash s; N \Downarrow_{int} s'; \mathbf{n} \qquad w\mathbf{v} \vdash (s' \otimes \mathbf{v} := \mathbf{n}); C[\mathbf{v}/x] \Downarrow_{cmd} (s'' \otimes \mathbf{v} := \mathbf{n}'); \mathbf{skip}}{w \vdash s; (\mathbf{new}\ x := N\ \mathbf{in}\ C\ \mathbf{end}) \Downarrow_{cmd} s''; \mathbf{skip}}\ (\text{if } \mathbf{v} \notin w)$$

Figure 1. IA **evaluation rules**

2.3. Definition (*IA* contextual equivalence). If M_1 and M_2 are IA terms of type σ with free identifiers contained in a set Γ of identifiers, and global variables contained in a set w of global variables, we write

$$w, \Gamma \vdash M_1 \cong_\sigma M_2$$

to indicate that the terms are *contextually equivalent*. By definition this means that for all worlds $w' \supseteq w$, all closed contexts $C[-_\sigma] : cmd$ with $gv(C[-_\sigma]) \subseteq w'$ and $\Gamma \subseteq traps(C[-_\sigma])$, and all states $s, s' \in States(w')$

$$w' \vdash C[M_1], s \Downarrow s' \Leftrightarrow w' \vdash C[M_2], s \Downarrow s'.$$

(In case $\Gamma = \emptyset$, i.e. when the M_i are closed terms, we just write $w \vdash M_1 \cong_\sigma M_2$ for $w, \emptyset \vdash M_1 \cong_\sigma M_2$.)

Thus two terms are contextually equivalent if occurrences of them in some closed command can be interchanged without affecting the meaning of the command as a partial function from states to states. This is a reasonable notion of program equivalence for IA, given that it is primarily a language for defining state-changing algorithms. It is immediate from the definition that contextual equivalence is a *congruence* for IA, *i.e.* it is an equivalence relation and is respected by the various term-forming operations. However, the quantification over all contexts that occurs in the definition of contextual equivalence makes it hard to establish further properties directly from 2.3. For example, it is not immediately obvious that two closed commands are contextually equivalent if they determine the same partial function from states to states. This is one of a number of useful 'extensionality' properties of IA that are summed up by the Operational Extensionality Theorem for IA given below. In order to state it we introduce the notion of extensional equivalence.

2.4. Definition (Extensional equivalence). If M_1 and M_2 are closed IA terms of type σ with global variables contained in a set w, we write

$$w \vdash M_1 \cong^{\text{ext}}_\sigma M_2$$

to indicate that the terms are *extensionally equivalent*. This notion is defined by induction on the structure of σ, as follows.

- If $\sigma = bool, int$, then $w \vdash M_1 \cong^{\text{ext}}_\sigma M_2$ is defined to hold if for all $s \in States(w)$ and all constants $\mathbf{c}$, $w \vdash M_1, s \Downarrow_\sigma \mathbf{c}$ if and only if $w \vdash M_2, s \Downarrow_\sigma \mathbf{c}$.
- $w \vdash C_1 \cong^{\text{ext}}_{cmd} C_2$ is defined to hold if for all $s, s' \in States(w)$, $w \vdash C_1, s \Downarrow s'$ if and only if $w \vdash C_2, s \Downarrow s'$.
- $w \vdash V_1 \cong^{\text{ext}}_{var} V_2$ is defined to hold if $w \vdash !V_1 \cong^{\text{ext}}_{int} !V_2$ and $w \vdash (V_1 := \mathbf{n}) \cong^{\text{ext}}_{cmd} (V_2 := \mathbf{n})$, for all $\mathbf{n}$.
- $w \vdash F_1 \cong^{\text{ext}}_{\sigma_1\to\sigma_2} F_2$ is defined to hold if for all $w' \supseteq w$, and all $A \in IA_{\sigma_1}(w')$, $w' \vdash F_1\,A \cong^{\text{ext}}_{\sigma_2} F_2\,A$.

We extend extensional equivalence to open terms via closed instantiations: given terms $M_1, M_2 : \sigma$ with free identifiers in $\Gamma = \{x^{\sigma_1}, \ldots, x^{\sigma_n}_n\}$ and global variables in w, we write

$$w, \Gamma \vdash M_1 \cong^{\text{ext}}_\sigma M_2$$

to mean that $w' \vdash M_1[\vec{A}/\vec{x}] \cong^{\mathrm{ext}}_{\sigma} M_1[\vec{A}/\vec{x}]$ holds for all $w' \supseteq w$ and all $A_i \in IA_{\sigma_i}(w')$ $(i = 1, \ldots, n)$.

2.5. Theorem (Operational Extensionality). *IA contextual equivalence coincides with extensional equivalence:*

$$w, \Gamma \vdash M_1 \cong_{\sigma} M_2 \Leftrightarrow w, \Gamma \vdash M_1 \cong^{\mathrm{ext}}_{\sigma} M_2.$$

We will prove this theorem in the next section as a corollary of the properties of the parametric logical relation for IA. We finish this section with some applications of the theorem to proving general properties of IA contextual equivalence from corresponding properties of the evaluation relation.

2.6. Example (Meyer-Sieber [5, Ex. 1]). If $C : cmd$ has its free identifiers contained in Γ and its global variables contained in w, and if $x^{var} \notin \Gamma$, then

$$w, \Gamma \vdash (\mathbf{new}\ x := \mathbf{n}\ \mathbf{in}\ C\ \mathbf{end}) \cong_{cmd} C.$$

Proof. According to Theorem 2.5, it suffices to show for all worlds w, all closed commands $C \in IA_{cmd}(w)$, and all states $s, s' \in States(w)$ that $w \vdash s; C \Downarrow s'$ holds if and only if $w \vdash s; (\mathbf{new}\ x := \mathbf{n}\ \mathbf{in}\ C\ \mathbf{end}) \Downarrow s'$. Because of the structural nature of the rules in Figure 1, the only way that the second evaluation can be deduced is from

$$w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{n}); C[\mathbf{v}/x] \Downarrow s' \otimes \mathbf{v} := \mathbf{n}' \qquad (6)$$

for some $\mathbf{v} \notin w$ and some $\mathbf{n}'$. Since C is closed, $C[\mathbf{v}/x] = C$; in particular $gv(C[\mathbf{v}/x]) = gv(C) \subseteq w$ and so by the 'Weakening and Strengthening' property of $\Downarrow$ (Lemma 2.2), (6) holds if and only if $w \vdash s; C \Downarrow s'$, as required. □

2.7. Example (Meyer-Sieber [5, Ex. 3]). Suppose $C : cmd$ has free identifiers in $\Gamma x_1^{var} x_2^{var}$ and global variables in w. Define:

$$\begin{array}{rcl} C_{12} & \stackrel{\mathrm{def}}{=} & \mathbf{new}\ x_1 := \mathbf{n}_1\ \mathbf{in} \\ & & \quad \mathbf{new}\ x_2 := \mathbf{n}_2\ \mathbf{in} \\ & & \quad\quad C \\ & & \quad \mathbf{end} \\ & & \mathbf{end} \\ C_{21} & \stackrel{\mathrm{def}}{=} & \mathbf{new}\ x_2 := \mathbf{n}_2\ \mathbf{in} \\ & & \quad \mathbf{new}\ x_1 := \mathbf{n}_1\ \mathbf{in} \\ & & \quad\quad C[x_2, x_1/x_1, x_2] \\ & & \quad \mathbf{end} \\ & & \mathbf{end} \end{array}$$

Then $w, \Gamma \vdash C_{12} \cong_{cmd} C_{21}$.

Proof. The argument is similar to that for the previous example, but using the 'Equivariance' property of $\Downarrow$ given in Lemma 2.2. □

Recall that in logics of partially defined terms, two partial terms are often called 'Kleene equivalent' if whenever one term is defined so is the other and in that case they are equal. Following a suggestion of Harper, we adopt this terminology for programming language expressions that may diverge. For IA this leads to the following, rather strong notion of equivalence.

2.8. Definition (Kleene equivalence). We say that closed terms $M_1, M_2 \in IA_{\sigma}(w)$ are *Kleene equivalent*, and write

$$w \vdash M_1 \cong^{\mathrm{kl}}_{\sigma} M_2$$

if for all s, s', R, $w \vdash s; M_1 \Downarrow_{\sigma} s'; R$ holds if and only if $w \vdash s; M_2 \Downarrow_{\sigma} s'; R$.

The following lemma is easily proved by induction on σ.

2.9. Lemma. *If $w \vdash M_1 \cong^{\mathrm{kl}}_{\sigma} M_2$ then $w \vdash M_1 \cong^{\mathrm{ext}}_{\sigma} M_2$.*

Thus in view of Theorem 2.5, *any Kleene equivalent IA terms are contextually equivalent*. Here are a number of examples, singled out because they will be needed later.

2.10. Examples. The following pairs of terms are Kleene equivalent and hence also contextually equivalent. In (vii)–(x), $\bot_{\sigma}$ is an abbreviation for $\mathbf{fix}\, x : \sigma\, .\, x$.

(i) $!(\mathbf{if}\ B\ \mathbf{then}\ V_1\ \mathbf{else}\ V_2)$ and $\mathbf{if}\ B\ \mathbf{then}\ !V_1\ \mathbf{else}\ !V_2$.

(ii) $(\mathbf{if}\ B\ \mathbf{then}\ V_1\ \mathbf{else}\ V_2) := N$ and $\mathbf{if}\ B\ \mathbf{then}\ V_1 := N\ \mathbf{else}\ V_2 := N$.

(iii) $V := (\mathbf{if}\ B\ \mathbf{then}\ N_1\ \mathbf{else}\ N_2)$ and $\mathbf{if}\ B\ \mathbf{then}\ V := N_1\ \mathbf{else}\ V := N_2$.

(iv) $(\mathbf{if}\ B\ \mathbf{then}\ F_1\ \mathbf{else}\ F_2)\, A$ and $\mathbf{if}\ B\ \mathbf{then}\ (F_1\ A)\ \mathbf{else}\ (F_2\ A)$.

(v) $(\lambda x : \sigma\, .\, M)\, A$ and $M[A/x^{\sigma}]$.

(vi) $\mathbf{fix}\, x : \sigma\, .\, M$ and $M[\mathbf{fix}\, x : \sigma\, .\, M/x^{\sigma}]$.

(vii) $\bot_{int}$ and $!(\bot_{var})$.

(viii) $\bot_{cmd}$ and $(\bot_{var} := N)$, or $(V := \bot_{int})$.

(ix) $\bot_{cmd}$ and $(\bot_{cmd}; C)$, or $(C; \bot_{cmd})$.

(x) $\bot_{\sigma'}$ and $(\bot_{\sigma \to \sigma'}\ A)$.

3. The parametric logical relation

If X is a set, the *lift of* X, $(X_{\bot}, \leq)$, is the so-called *flat* partially ordered set whose set of elements is $X \cup \{\bot_X\}$ (where $\bot_X \notin X$) and whose only non-trivial ordering is $\bot_X \leq x$ (any $x \in X$). We need this construct in case $X = States(w)$ is the set of states at world w. We refer

to the elements of $States(w)_\perp$ as *lifted states*, and denote its least element by $\perp$ (for all w). It is convenient to extend the evaluation and divergence relations to lifted states by declaring that $w \vdash \perp; M \Downarrow_\sigma s'; R$ does not hold for any $M, R \in IA_\sigma(w)$, $s' \in States(w)$, but that

$$w \vdash \perp; M \Uparrow_\sigma$$

always holds for any $M \in IA_\sigma(w)$.

We will be working with binary relations between lifted states that relate $\perp$ to itself. For each finite set w of global variables we define

$$Rel(w) \stackrel{\text{def}}{=} \{\mathcal{R} \subseteq States(w)_\perp \times States(w)_\perp \mid (\perp, \perp) \in \mathcal{R}\}.$$

3.1. Definition. The *identity relation*, $\mathcal{I}d_w \in Rel(w)$, is defined to be $\{(s, s) \mid s \in States(w)_\perp\}$. If w_1 and w_2 are disjoint sets of global variables, we write $w_1 w_2$ for their union. The *smash product* $\mathcal{R}_1 \otimes \mathcal{R}_2 \in Rel(w_1 w_2)$ of relations $\mathcal{R}_i \in Rel(w_i)$ is defined to be $\{(s_1 \otimes s_2, s_1' \otimes s_2') \mid (s_1, s_1') \in \mathcal{R}_1 \;\&\; (s_2, s_2') \in \mathcal{R}_2\}$, where

$$s_1 \otimes s_2 \stackrel{\text{def}}{=} \begin{cases} s_1 \cup s_2 & \text{if } s_1 \neq \perp \neq s_2 \\ \perp & \text{if } s_1 = \perp, \text{ or } s_2 = \perp \end{cases}$$

where $s_1 \cup s_2 \in States(w_1 w_2)$ is the state mapping $\mathbf{v}$ to $s_i(\mathbf{v})$ if $\mathbf{v} \in w_i$ (for $i = 1, 2$).

Note that the smash product operation on relations is associative, commutative, and has identity relations as units:

$$(\mathcal{R}_1 \otimes \mathcal{R}_2) \otimes \mathcal{R}_3 = \mathcal{R}_1 \otimes (\mathcal{R}_2 \otimes \mathcal{R}_3),$$
$$\mathcal{R}_1 \otimes \mathcal{R}_2 = \mathcal{R}_2 \otimes \mathcal{R}_1,$$
$$\mathcal{I}d_w \otimes \mathcal{R} = \mathcal{R}.$$

Armed with these notions we can give the principal definition of the paper.

3.2. Definition (Parametric logical relation). For each finite set w of global variables, each type σ, and each relation $\mathcal{R} \in Rel(w)$, we define a binary relation between closed IA terms of type σ with global variables in w, denoted

$$w \vdash M_1 \; \mathcal{R}_\sigma \; M_2 \qquad (M_1, M_2 \in IA_\sigma(w))$$

The relations are defined simultaneously for all w and $\mathcal{R}$, by induction on the structure of σ, as follows. (In giving the clauses, we make use of the extension of the evaluation and divergence relations to lifted states mentioned above.)

- If $\sigma = bool$ (respectively $\sigma = int$), then $w \vdash M_1 \; \mathcal{R}_\sigma \; M_2$ is defined to hold if for all boolean (respectively integer) constants $\mathbf{c}_1, \mathbf{c}_2$, and all $(s_1, s_2) \in \mathcal{R}$

  $$w \vdash s_1; M_1 \Downarrow_\sigma \mathbf{c}_1 \;\&\; w \vdash s_2; M_2 \Downarrow_\sigma \mathbf{c}_2 \Rightarrow \mathbf{c}_1 = \mathbf{c}_2,$$

  $$w \vdash s_1; M_1 \Downarrow_\sigma \mathbf{c}_1 \;\&\; w \vdash s_2; M_2 \Uparrow_\sigma \Rightarrow (s_1, \perp) \in \mathcal{R},$$

  and

  $$w \vdash s_1; M_1 \Uparrow_\sigma \;\&\; w \vdash s_2; M_2 \Downarrow_\sigma \mathbf{c}_2 \Rightarrow (\perp, s_2) \in \mathcal{R}.$$

- If $\sigma = cmd$, then $w \vdash C_1 \; \mathcal{R}_{cmd} \; C_2$ is defined to hold if for all states $s_1', s_2' \in States(w)$, and all $(s_1, s_2) \in \mathcal{R}$

  $$w \vdash s_1; C_1 \Downarrow s_1' \;\&\; w \vdash s_2; C_2 \Downarrow s_2' \Rightarrow (s_1', s_2') \in \mathcal{R},$$

  $$w \vdash s_1; C_1 \Downarrow s_1' \;\&\; w \vdash s_2; C_2 \Uparrow_{cmd} \Rightarrow (s_1', \perp) \in \mathcal{R},$$

  and

  $$w \vdash s_1; C_1 \Uparrow_{cmd} \;\&\; w \vdash s_2; C_2 \Downarrow s_2' \Rightarrow (\perp, s_2') \in \mathcal{R}.$$

- If $\sigma = var$, then $w \vdash V_1 \; \mathcal{R}_{var} \; V_2$ is defined to hold if for all global variables $\mathbf{v}_1, \mathbf{v}_2 \in w$, and all $(s_1, s_2) \in \mathcal{R}$

  $$w \vdash s_1; V_1 \Downarrow_{var} \mathbf{v}_1 \;\&\; w \vdash s_2; V_2 \Downarrow_{var} \mathbf{v}_2 \Rightarrow \forall \mathbf{n}\, ((s_1; \mathbf{v}_1 := \mathbf{n}), (s_2; \mathbf{v}_2 := \mathbf{n})) \in \mathcal{R} \;\&\; s_1(\mathbf{v}_1) = s_2(\mathbf{v}_2),$$

  $$w \vdash s_1; V_1 \Downarrow_{var} \mathbf{v}_1 \;\&\; w \vdash s_2; V_2 \Uparrow_{var} \Rightarrow \forall \mathbf{n}\, ((s_1; \mathbf{v}_1 := \mathbf{n}), \perp) \in \mathcal{R},$$

  and

  $$w \vdash s_1; V_1 \Uparrow_{var} \;\&\; w \vdash s_2; V_2 \Downarrow_{var} \mathbf{v}_2 \Rightarrow \forall \mathbf{n}\, (\perp, (s_2; \mathbf{v}_2 := \mathbf{n})) \in \mathcal{R}.$$

- If $\sigma = \sigma_1 \to \sigma_2$, then $w \vdash F_1 \; \mathcal{R}_{\sigma_1 \to \sigma_2} \; F_2$ is defined to hold if for all $\mathcal{R}' \in Rel(w')$ with w' disjoint from w, and all $A_1, A_2 \in IA_{\sigma_1}(ww')$

  $$ww' \vdash A_1 \; (\mathcal{R} \otimes \mathcal{R}')_{\sigma_1} \; A_2 \Rightarrow ww' \vdash (F_1 \, A_1) \; (\mathcal{R} \otimes \mathcal{R}')_{\sigma_2} \; (F_2 \, A_2).$$

3.3. Remark. The last clause in this definition is an operational version of the kind of relational parametricity for functions used previously by O'Hearn and Tennent (see [10, Sect. 2.2]); it also embodies the typical feature of 'logical relations'—that functions map related arguments to related results. The way the logical relation takes account of divergence in the clauses for $bool$, int, cmd, and var reflects recent work of O'Hearn and Reynolds [9], and is crucial for Example 4.1. There is a more elegant formulation of those clauses, given below. We chose to take the more concrete form in Definition 3.2, because it is useful for calculations. The proof of the following two properties is a tedious, but essentially straightforward case analysis (making use of the determinacy of evaluation, Lemma 2.2).

(i) Let $Val_\sigma(w)$ denote the set of closed syntactic *values* of type σ with global variables in the set w—*i.e.* the set of those IA terms R appearing on the right-hand side of evaluations (2). (For example, $Val_{cmd}(w)$ is just $\{\mathbf{skip}\}$.) For each $s \in States(w)_\perp$ and $M \in IA_\sigma(w)$, let $s \otimes M \in (States(w) \times Val_\sigma(w))_\perp$ be defined by

$$s \otimes M \stackrel{\text{def}}{=} \begin{cases} (s', R) & \text{if } w \vdash s; M \Downarrow_\sigma s'; R \\ \perp & \text{otherwise} \end{cases}$$

Given $\mathcal{R} \in Rel(w)$, let $\mathcal{R} \otimes \mathcal{I}d$ be the binary relation on $(States(w) \times Val_\sigma(w))_\perp$ given by

$$\mathcal{R} \otimes \mathcal{I}d \stackrel{\text{def}}{=} \{(\perp, \perp)\} \cup \{(s_1 \otimes R, s_2 \otimes R) \mid (s_1, s_2) \in \mathcal{R} \;\&\; R \in Val_\sigma(w)\}.$$

Then when $\sigma = bool$, int, or cmd, $w \vdash M_1 \; \mathcal{R}_\sigma \; M_2$ holds if and only if for all $(s_1, s_2) \in \mathcal{R}$

$$(s_1 \otimes M_1, s_2 \otimes M_2) \in \mathcal{R} \otimes \mathcal{I}d.$$

(ii) $w \vdash V_1 \; \mathcal{R}_{var} \; V_2$ holds if and only if $w \vdash !V_1 \; \mathcal{R}_{int} \; !V_2$ and moreover for all $\mathbf{n}$, $w \vdash (V_1{:=}\mathbf{n}) \; \mathcal{R}_{cmd} \; (V_2{:=}\mathbf{n})$.

3.4. Lemma. *The parametric logical relation respects extensional equivalence (Definition* 2.4*), in the sense that if* $w \vdash M_1 \; \mathcal{R}_\sigma \; M_2$ *and* $w \vdash M_i \cong^{\text{ext}}_\sigma M_i'$ *(*$i = 1, 2$*), then* $w \vdash M_1' \; \mathcal{R}_\sigma \; M_2'$.

Proof. This follows directly from Definitions 2.4 and 3.2, by induction on the structure of σ. □

3.5. Definition. Extend the parametric logical relation to open terms as follows. Given $\mathcal{R} \in Rel(w)$ and terms $M_1, M_2{:}\sigma$ with free identifiers in $\Gamma = \{x^{\sigma_1}, \ldots, x^{\sigma_n}_n\}$ and global variables in w, write

$$w, \Gamma \vdash M_1 \; \mathcal{R}_\sigma \; M_2$$

to mean that for all $\mathcal{R}' \in Rel(w')$ with w' disjoint from w, and for all closed terms $A_{i1}, A_{i2} \in IA_{\sigma_i}(ww')$ $(i = 1, \ldots, n)$

$$\forall i \, \big(ww' \vdash A_{i1} \; (\mathcal{R} \otimes \mathcal{R}')_{\sigma_i} \; A_{i2}\big) \Rightarrow M_1[\vec{A_1}/\vec{x}] \; (\mathcal{R} \otimes \mathcal{R}')_\sigma \; M_2[\vec{A_2}/\vec{x}].$$

This definition reduces to the one in Definition 3.2 in the case that $\Gamma = \emptyset$, because of the following weakening property of the parametric logical relation.

3.6. Lemma. *(i) If* $\mathcal{R} \in Rel(w)$ *and* $\mathcal{R}' \in Rel(w')$ *with* $w \cap w' = \emptyset$*, and if* Γ *and* Γ' *are disjoint sets of identifiers, then*

$$w, \Gamma \vdash M_1 \; \mathcal{R}_\sigma \; M_2 \Rightarrow ww', \Gamma\Gamma' \vdash M_1 \; (\mathcal{R} \otimes \mathcal{R}')_\sigma \; M_2.$$

(ii) $w, \emptyset \vdash M_1 \; \mathcal{R}_\sigma \; M_2$ *(*i.e. *the* $\Gamma = \emptyset$ *case of* 3.5*) holds if and only if* $w \vdash M_1 \; \mathcal{R}_\sigma \; M_2$.

Proof. Part (i) reduces (using the associativity of $\otimes$) to proving the corresponding property of the relation between closed terms:

$$w \vdash M_1 \; \mathcal{R}_\sigma \; M_2 \Rightarrow ww' \vdash M_1 \; (\mathcal{R} \otimes \mathcal{R}')_\sigma \; M_2.$$

This is proved by induction on σ, using the corresponding weakening property of evaluation (Lemma 2.2). Part (ii) follows immediately from this property too. □

3.7. Proposition. *The parametric logical relation preserves the term-forming operations of* IA*:*

(i) For $\mathbf{c} = \mathbf{b}, \mathbf{n}$, *and* $\mathbf{skip}$

$$\emptyset \vdash \mathbf{c} \; (\mathcal{I}d_\emptyset)_\sigma \; \mathbf{c}$$

(where $\sigma = bool, int,$ *and* cmd *respectively).*

(ii) If $\mathbf{v} \in w$*, then*

$$w \vdash \mathbf{v} \; (\mathcal{I}d_w)_{var} \; \mathbf{v}.$$

(iii) If $w, \Gamma \vdash B_1 \; \mathcal{R}_{bool} \; B_2$, $w, \Gamma \vdash M_1 \; \mathcal{R}_\sigma \; M_2$*, and* $w, \Gamma \vdash M_1' \; \mathcal{R}_\sigma \; M_2'$*, then*

$$w, \Gamma \vdash (\mathbf{if} \; B_1 \; \mathbf{then} \; M_1 \; \mathbf{else} \; M_1') \; \mathcal{R}_\sigma \; (\mathbf{if} \; B_2 \; \mathbf{then} \; M_2 \; \mathbf{else} \; M_2').$$

(iv) If $w, \Gamma \vdash N_1 \; \mathcal{R}_{int} \; N_2$ *and* $w, \Gamma \vdash N_1' \; \mathcal{R}_{int} \; N_2'$*, then*

$$w, \Gamma \vdash (N_1 * N_1') \; \mathcal{R}_{type(*)} \; (N_2 * N_2').$$

(v) If $w, \Gamma \vdash F_1 \; \mathcal{R}_{\sigma \to \sigma'} \; F_2$ *and* $w, \Gamma \vdash A_1 \; \mathcal{R}_\sigma \; A_2$*, then*

$$w, \Gamma \vdash (F_1 \; A_1) \; \mathcal{R}_{\sigma'} \; (F_2 \; A_2).$$

(vi) If $w, \Gamma \vdash V_1 \; \mathcal{R}_{var} \; V_2$*, then*

$$w, \Gamma \vdash !V_1 \; \mathcal{R}_{int} \; !V_2.$$

(vii) If $w, \Gamma \vdash V_1 \; \mathcal{R}_{var} \; V_2$ *and* $w, \Gamma \vdash N_1 \; \mathcal{R}_{int} \; N_2$*, then*

$$w, \Gamma \vdash (V_1 := N_1) \; \mathcal{R}_{cmd} \; (V_2 := N_2).$$

(viii) If $w, \Gamma \vdash C_1 \; \mathcal{R}_{cmd} \; C_2$ *and* $w, \Gamma \vdash C_1' \; \mathcal{R}_{cmd} \; C_2'$*, then*

$$w, \Gamma \vdash (C_1 \; ; C_1') \; \mathcal{R}_{cmd} \; (C_2 \; ; C_2').$$

(ix) If $w, \Gamma x^\sigma \vdash M_1 \; \mathcal{R}_{\sigma'} \; M_2$*, then*

$$w, \Gamma \vdash \lambda x{:}\sigma \,.\, M \; \mathcal{R}_{\sigma \to \sigma'} \; \lambda x{:}\sigma \,.\, M_2.$$

(x) If $w, \Gamma \vdash N_1 \; \mathcal{R}_{int} \; N_2$ *and for some* $\mathbf{v} \notin w$ *it is the case that* $w\mathbf{v}, \Gamma \vdash C_1[\mathbf{v}/x^{var}] \; \mathcal{R}_{cmd} \; C_2[\mathbf{v}/x^{var}]$*, then*

$$w, \Gamma \vdash (\mathbf{new}\ x := N_1\ \mathbf{in}\ C_1\ \mathbf{end}) \\ \mathcal{R}_{cmd}\ (\mathbf{new}\ x := N_1\ \mathbf{in}\ C_1\ \mathbf{end}).$$

(xi) If $w, \Gamma x^\sigma \vdash M_1 \; \mathcal{R}_\sigma \; M_2$*, then*

$$w, \Gamma \vdash \mathbf{fix}\, x{:}\sigma \,.\, M_1 \; \mathcal{R}_\sigma \; \mathbf{fix}\, x{:}\sigma \,.\, M_2.$$

Proof (sketch). The properties (i)–(x) follow from Definitions 3.2 and 3.5 by relatively straightforward arguments, using closure under extensional equivalence (Lemma 3.4) combined with the particular Kleene, hence extensional (by Lemma 2.9) equivalences of Example 2.10. For part (iii), one has to argue by induction on the structure of σ.

However, the proof of property (xi) requires more work. First, one can show by induction on σ that $w \vdash \bot_\sigma \; \mathcal{R}_\sigma \; \bot_\sigma$ (where $\bot_\sigma \stackrel{\text{def}}{=} \mathbf{fix}\, x{:}\sigma \,.\, x$). The proof of this uses the fact that $(\bot, \bot) \in \mathcal{R}$ (by definition of $Rel(w)$) together with the properties of $\bot_\sigma$ given in Example 2.10. Then from $w, \Gamma x^\sigma \vdash M_1 \; \mathcal{R}_\sigma \; M_2$ one deduces by induction on n that

$$w, \Gamma \vdash \mathbf{fix}^{(n)} x{:}\sigma \,.\, M_1 \; \mathcal{R}_\sigma \; \mathbf{fix}^{(n)} x{:}\sigma \,.\, M_2$$

where in general

$$\mathbf{fix}^{(0)} x{:}\sigma \,.\, M \stackrel{\text{def}}{=} \bot_\sigma$$
$$\mathbf{fix}^{(n+1)} x{:}\sigma \,.\, M \stackrel{\text{def}}{=} M[\mathbf{fix}^{(n)} x{:}\sigma \,.\, M/x].$$

Therefore property (xi) follows once one proves that the relations $\mathcal{R}_\sigma$ satisfy an operational version of chain-completeness. Since such a property is proved by induction on σ, to make that induction go through easily it is convenient to use the following, contextual form of operational chain-completeness:

Proposition (Operational chain-completeness). *The relations* $\mathcal{R}_\sigma$ *have the property that for all contexts* $C[-_\sigma] : \sigma'$

$$w, \Gamma \vdash C[\mathbf{fix}\, x{:}\sigma \,.\, M_1] \mathcal{R}_{\sigma'} C[\mathbf{fix}\, x{:}\sigma \,.\, M_2]$$

holds if for all $m \geq 0$ *there is some* $n \geq m$ *such that*

$$w, \Gamma \vdash C[\mathbf{fix}^{(n)} x{:}\sigma \,.\, M_1] \mathcal{R}_{\sigma'} C[\mathbf{fix}^{(n)} x{:}\sigma \,.\, M_2].$$

The proof of this property follows from a (simple form of a) compactness property of evaluation with respect to the approximations $\mathbf{fix}^{(n)} x{:}\sigma \,.\, M$ to $\mathbf{fix}\, x{:}\sigma \,.\, M$, namely:

Proposition (Compactness of evaluation). *If*

$$w \vdash s; C[\mathbf{fix}\, x{:}\sigma \,.\, M] \Downarrow_\sigma s'; R$$

then $w \vdash C[\mathbf{fix}^{(n)} x{:}\sigma \,.\, M], s \Downarrow_\sigma R', s'$ *holds for some* n *and* R'.

There are several methods for proving this proposition. Our preferred one is a generalisation to IA of [12, Sect. 5], because that approach yields a more involved form of compactness of evaluation which is useful for establishing operational chain-completeness properties in the presence of more complicated datatypes (such as lazy lists). We omit the details here. □

3.8. Remark (Contraction). The syntactic form of IA's $\mathbf{new}\ -:=-\ \mathbf{in}\ -\ \mathbf{end}$ construct suggests the second hypothesis of part (x) of Proposition 3.7 should just be

$$w, \Gamma x^{var} \vdash C_1 \; \mathcal{R}_{cmd} \; C_2$$

rather than

$$w\mathbf{v}, \Gamma \vdash C_1[\mathbf{v}/x^{var}] \; \mathcal{R}_{cmd} \; C_2[\mathbf{v}/x^{var}].$$

The latter is a weaker assumption than the former, and so (x) as stated is a stronger form of preservation than one might expect. It reflects the fact that $\mathbf{new}\ -:=N\ \mathbf{in}\ C[-]\ \mathbf{end}$ is really a binding operation on variables rather than identifiers. From a metalogical point of view, the difference between 'variables' and 'identifiers' (which a metalogician might well prefer to call 'constants' and 'variables' respectively) lies in the structural rules satisfied by the judgements of the form

$$w, \Gamma \vdash M_1 =_\sigma M_2$$

with which we formulate the various notions of equivalence of IA terms considered in this paper. In particular, the two zones of the 'context' (in type theory parlance) on the left-hand side of $\vdash$ have different properties with respect to substitution:

$$\frac{w, \Gamma x^\sigma \vdash M_1 =_{\sigma'} M_2}{w, \Gamma \vdash M_1[M/x^\sigma] =_{\sigma'} M_2[M/x^\sigma]} \text{ (if } w, \Gamma \vdash M{:}\sigma)$$

$$\frac{w\mathbf{v}, \Gamma \vdash M_1 =_{\sigma'} M_2}{w\mathbf{v}', \Gamma \vdash M_1[\mathbf{v}'/\mathbf{v}] =_{\sigma'} M_2[\mathbf{v}'/\mathbf{v}]}$$

In combination with weakening properties (such as Lemma 3.6(i)), this means that the 'Γ-zone' satisfies contraction

$$\frac{w, \Gamma x_1^\sigma x_2^\sigma \vdash M_1 =_{\sigma'} M_2}{w, \Gamma x^\sigma \vdash M_1[x^\sigma, x^\sigma/x_1^\sigma, x_2^\sigma] =_{\sigma'} M_2[x^\sigma, x^\sigma/x_1^\sigma, x_2^\sigma]}$$

whereas the 'w-zone' does not. For example

$$\mathbf{v}_1\mathbf{v}_2, \emptyset \vdash (\mathbf{v}_1 := \mathbf{1}\ ;\ \mathbf{v}_2 := \mathbf{2}) \cong_{cmd} (\mathbf{v}_2 := \mathbf{2}\ ;\ \mathbf{v}_1 := \mathbf{1})$$

holds, but of course the contracted form

$$\mathbf{v}, \emptyset \vdash (\mathbf{v} := \mathbf{1}\ ;\ \mathbf{v} := \mathbf{2}) \cong_{cmd} (\mathbf{v} := \mathbf{2}\ ;\ \mathbf{v} := \mathbf{1})$$

does not.

3.9. Theorem (Fundamental Property). *(i) For any IA term $M{:}\sigma$ with free identifiers contained in Γ and global variables contained in w, it is the case that $w, \Gamma \vdash M\ (\mathcal{I}d_w)_\sigma\ M$.*

(ii) If $w, \Gamma\Gamma' \vdash M_1\ (\mathcal{I}d_w)_\sigma\ M_2$, then for all worlds $w' \supseteq w$ and contexts $C[-_\sigma] : \sigma'$ with $gv(C[-_\sigma]) \subseteq w'$ and $\Gamma \subseteq traps(C[-_\sigma])$, it is the case that $w', \Gamma' \vdash C[M_1]\ (\mathcal{I}d_{w'})_{\sigma'}\ C[M_2]$.

Proof. Both parts are proved by induction on the structure of M and $C[-_\sigma]$, using Proposition 3.7 and Lemma 3.6. □

3.10. Corollary. *The parametric logical relation in case $\mathcal{R} = \mathcal{I}d$ coincides with extensional equivalence:*

$$w, \Gamma \vdash M_1\ (\mathcal{I}d_w)_\sigma\ M_2 \Leftrightarrow w, \Gamma \vdash M_1 \cong^{\mathrm{ext}}_\sigma M_2.$$

Proof. By Theorem 3.9(i), $w, \Gamma \vdash M_1\ (\mathcal{I}d_w)_\sigma\ M_1$. So if $w, \Gamma \vdash M_1 \cong^{\mathrm{ext}}_\sigma M_2$, then by Lemma 3.4 we have that $w, \Gamma \vdash M_1\ (\mathcal{I}d_w)_\sigma\ M_2$. This is half of the required bi-implication. The other half can be proved directly from Definitions 3.2 and 2.4, by induction on the structure of σ. □

We are now in a position to prove the Operational Extensionality Theorem 2.5.

Proof of Theorem 2.5. We split the proof into three parts:

$$w, \Gamma \vdash M_1 \cong^{\mathrm{ext}}_\sigma M_2 \Rightarrow w, \Gamma \vdash M_1 \cong_\sigma M_2 \qquad (7)$$

$$w, \Gamma x^\sigma \vdash M_1 \cong_{\sigma'} M_2 \Rightarrow w', \Gamma \vdash M_1[A/x^\sigma] \cong_{\sigma'} M_2[A/x^\sigma] \qquad (8)$$

for any $A{:}\sigma$ with free identifiers in Γ and global variables in $w' \supseteq w$, and

$$w \vdash M_1 \cong_\sigma M_2 \Rightarrow w \vdash M_1 \cong^{\mathrm{ext}}_\sigma M_2 \qquad (9)$$

Repeated use of (8) reduces the converse of (7) to the special case when $\Gamma = \emptyset$, which is (9). Thus together these properties yield the required bi-implication.

Proof of (7): Suppose that $w, \Gamma \vdash M_1 \cong^{\mathrm{ext}}_\sigma M_2$ and hence by the above Corollary that $w, \Gamma \vdash M_1\ (\mathcal{I}d_w)_\sigma\ M_2$. We wish to show that $w, \Gamma \vdash M_1 \cong_\sigma M_2$, *i.e.* that for all $w' \supseteq w$, all closed contexts $C[-_\sigma]{:}cmd$ with $gv(C[-_\sigma]) \subseteq w'$ and $\Gamma \subseteq traps(C[-_\sigma])$, and all states $s, s' \in States(w')$

$$w' \vdash C[M_1], s \Downarrow s' \Leftrightarrow w' \vdash C[M_2], s \Downarrow s'.$$

But given such a context, by Theorem 3.9(ii) (and Lemma 3.6(ii)) we have $w' \vdash C[M_1]\ (\mathcal{I}d_{w'})_{cmd}\ C[M_2]$. Since $(s, s) \in \mathcal{I}d_{w'}$, it follows from the definition of the parametric logical relation at type cmd and from the definition of $\mathcal{I}d_{w'}$ that the above bi-implication holds.

Proof of (8): If $w, \Gamma x^\sigma \vdash M_1 \cong_{\sigma'} M_2$, then straight from the definition of contextual equivalence we get $w', \Gamma \vdash (\lambda x{:}\sigma\,.\,M_1)\,A \cong_{\sigma\to\sigma'} (\lambda x{:}\sigma\,.\,M_2)\,A$. The result follows by transitivity of $\cong$ together with the fact that β-conversion (Example 2.10(v)) is a valid Kleene equivalence, hence is a valid extensional equivalence (by Lemma 2.9) and so is a valid contextual equivalence, by (7).

Proof of (9): It is straightforward to show that $\cong_\sigma$ satisfies the defining clauses for $\cong^{\mathrm{ext}}_\sigma$ in Definition 2.4, by induction on the structure of σ. □

3.11. Remark (Doing without $\bot$). We have developed a logical relation parameterised by relations between *lifted* states, because that seems necessary for proving some contextual equivalences (specifically, Example 4.1). However, many examples and the operational Extensionality Theorem itself, can be deduced using a simpler logical relation, call it $w \vdash - \overline{\mathcal{R}}_\sigma -$, which is parameterised merely by binary relations on states, $R \subseteq States(w) \times States(w)$. The defining clause at type cmd is: $w \vdash C_1\ \overline{\mathcal{R}}_{cmd}\ C_2$ holds if and only if for all $(s_1, s_2) \in \mathcal{R}$ and $s_1', s_2' \in States(w)$

$$w \vdash s_1; C_1 \Downarrow s_1' \Rightarrow \exists s_2'\,(w \vdash s_2; C_2 \Downarrow s_2'\ \&\ (s_1', s_2') \in \mathcal{R})$$
$$w \vdash s_2; C_2 \Downarrow s_2' \Rightarrow \exists s_1'\,(w \vdash s_1; C_1 \Downarrow s_1'\ \&\ (s_1', s_2') \in \mathcal{R}).$$

The defining clause at types $\sigma = bool, int$ is: $w \vdash M_1\ \overline{\mathcal{R}}_\sigma\ M_2$ holds if and only if for all $(s_1, s_2) \in \mathcal{R}$ and constants $\mathbf{c}$

$$w \vdash s_1; M_1 \Downarrow_\sigma \mathbf{c} \Leftrightarrow w \vdash s_2; M_2 \Downarrow_\sigma \mathbf{c}.$$

At type var we can define $\overline{\mathcal{R}}_{var}$ as in Remark 3.3(ii). Finally, the definition of $\overline{\mathcal{R}}_{\sigma_1\to\sigma_2}$ is the same as for $\mathcal{R}_{\sigma_1\to\sigma_2}$ (except that one is quantifying over a different kind of state-relation).

4. Example equivalences

The Fundamental Property of the parametric logical relation (Theorem 3.9) and its relationship to contextual equivalence (Corollary 3.10 plus Theorem 2.5) enable one to use the definition of the logical relation at function types to reason about properties of procedures with respect to local variables. Here are some examples.

4.1. Example (O'Hearn [8, 2.3]). Let

$$\begin{aligned} C_1 \stackrel{\mathrm{def}}{=}\ & \mathbf{new}\ x := \mathbf{0}\ \mathbf{in} \\ & \quad p\,(x := \mathbf{1})\,; \\ & \quad \mathbf{if}\ !x = \mathbf{1}\ \mathbf{then}\ \bot_{cmd}\ \mathbf{else}\ \mathbf{skip} \\ & \mathbf{end} \\ C_2 \stackrel{\mathrm{def}}{=}\ & p\,(\bot_{cmd}). \end{aligned}$$

where $\bot_{cmd} \stackrel{\mathrm{def}}{=} \mathbf{fix}\,c{:}cmd\,.\,c$. Then $p : cmd \to cmd \vdash C_1 \cong_{cmd} C_2$.

Proof. By the Operational Extensionality Theorem 2.5 it suffices to prove for all worlds w, all terms $P \in IA_{cmd\to cmd}(w)$, and all states $s, s' \in States(w)$, that $w \vdash s; C_1[P/p] \Downarrow s'$ if and only if $w \vdash s; C_2[P/p] \Downarrow s'$. From the rules in Figure 1 it follows that $w \vdash s; C_1[P/p] \Downarrow s'$ holds if and only if for some (any) $\mathbf{v} \notin w$,

$$\exists \mathbf{n} \neq \mathbf{1} \,.\, w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\mathbf{v} := \mathbf{1}) \Downarrow s' \otimes \mathbf{v} := \mathbf{n}. \quad (10)$$

And since $\mathbf{v} \notin gv(P)$, $w \vdash s; C_2[P/p] \Downarrow s'$ holds if and only if

$$w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\bot_{cmd}) \Downarrow s' \otimes \mathbf{v} := \mathbf{0}. \quad (11)$$

Putting these facts together, to verify the example we must prove that (10) holds if and only if (11) does.

Define $\mathcal{R} \in Rel(\{\mathbf{v}\})$ to be

$$\mathcal{R} \stackrel{\text{def}}{=} \{(\bot,\bot)\} \cup \{(s_1,s_2) \mid s_1(\mathbf{v}) = \mathbf{0} = s_2(\mathbf{v})\} \cup \{(s_1,s_2) \mid s_1(\mathbf{v}) = \mathbf{1} \;\&\; s_2 = \bot\}.$$

From the definition of the logical relation at type cmd (Definition 3.2) we find that

$$w\mathbf{v} \vdash \mathbf{v} := \mathbf{1}\ (\mathcal{I}d_w \otimes \mathcal{R})_{cmd}\ \bot_{cmd}$$

holds. (Note that the ability of the parameter of the logical relation to relate states to $\bot$ is crucial for this.) By the Fundamental Property 3.9, $w \vdash P\ (\mathcal{I}d_w)_{cmd\to cmd}\ P$. Hence by the definition of the logical relation at type $cmd \to cmd$, we have

$$w\mathbf{v} \vdash P\,(\mathbf{v} := \mathbf{1})\ (\mathcal{I}d_w \otimes \mathcal{R})_{cmd}\ P\,(\bot_{cmd}). \quad (12)$$

Note that by definition of $\mathcal{R}$, for any $s' \in States(w)$ and $s_2 \in States(w)_\bot$ we have

$$(s' \otimes \mathbf{v} := \mathbf{n}, s_2) \in \mathcal{I}d_w \otimes \mathcal{R} \Leftrightarrow \big(\mathbf{n} = \mathbf{0} \;\&\; s_2 = s' \otimes \mathbf{v} := \mathbf{0}\big) \vee \big(\mathbf{n} = \mathbf{1} \;\&\; s_2 = \bot\big). \quad (13)$$

We apply the definition of $(\mathcal{I}d_w \otimes \mathcal{R})_{cmd}$ to (12) at the pair $(s \otimes \mathbf{v} := \mathbf{0}, s \otimes \mathbf{v} := \mathbf{0}) \in \mathcal{I}d_w \otimes \mathcal{R}$. If (10) holds, then it cannot be that $w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\bot_{cmd}) \Uparrow_{cmd}$, since in that case we would have $(s' \otimes \mathbf{v} := \mathbf{n}, \bot) \in \mathcal{I}d_w \otimes \mathcal{R}$ with $\mathbf{n} \neq \mathbf{1}$, contradicting (13). So $w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\bot_{cmd}) \Downarrow s_2$ holds for some s_2. Since it is the case that $(s' \otimes \mathbf{v} := \mathbf{n}, s_2) \in \mathcal{I}d_w \otimes \mathcal{R}$, (13) implies that $s_2 = s' \otimes \mathbf{v} := \mathbf{0}$ and hence (11) holds. We thus have that (10) implies (11). Starting with the observation that for all $s_1 \in States(w)_\bot$ and $s' \in States(w)$

$$(s_1, s' \otimes \mathbf{v} := \mathbf{0}) \in \mathcal{I}d_w \otimes \mathcal{R} \Leftrightarrow s_1 = s' \otimes \mathbf{v} := \mathbf{0}$$

one can show the converse implication by a similar argument. Thus (10) if and only if (11), as required. □

4.2. Example (Stoughton [5, Ex. 5]). Let

$$C_3 \stackrel{\text{def}}{=} \mathbf{new}\ x := \mathbf{0}\ \mathbf{in}\ p\,(x := !x + \mathbf{2})\,;\ \mathbf{if}\ even(x)\ \mathbf{then}\ \bot_{cmd}\ \mathbf{else}\ \mathbf{skip}\ \mathbf{end}$$

where $\bot_{cmd}$ is as in the previous example and $even$ is a suitable fixpoint term of type $int \to bool$ expressing a test for divisibility by $\mathbf{2}$. Then $p : cmd \to cmd \vdash C_3 \cong_{cmd} \bot_{cmd}$.

Proof. As in the previous example, the problem reduces via the Operational Extensionality Theorem to showing for all worlds w, terms $P \in IA_{cmd\to cmd}(w)$, and states $s \in States(w)$, that $w \vdash C_3[P/p], s \Uparrow_{cmd}$. It follows from the rules in Figure 1 that $w \vdash C_3[P/p], s \Downarrow s'$ holds if and only if for some (any) $\mathbf{v} \notin w$

$$w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\mathbf{v} := !\mathbf{v} + \mathbf{2}), \Downarrow s' \otimes \mathbf{v} := \mathbf{n} \quad (14)$$

holds for some *odd* integer $\mathbf{n}$. Define $\mathcal{R} \in Rel(\{\mathbf{v}\})$ to be

$$\mathcal{R} \stackrel{\text{def}}{=} \{(\bot,\bot)\} \cup \{(s_1,s_2) \mid s_1(\mathbf{v}) = s_2(\mathbf{v}) \text{ is even}\}.$$

Then

$$w\mathbf{v} \vdash (\mathbf{v} := !\mathbf{v} + \mathbf{2})(\mathcal{I}d \otimes \mathcal{R})_{cmd}(\mathbf{v} := !\mathbf{v} + \mathbf{2}).$$

So if (14) holds, since $(s \otimes \mathbf{v} := \mathbf{0}, s \otimes \mathbf{v} := \mathbf{0})$ is in $\mathcal{I}d_w \otimes \mathcal{R}$, so is the pair $(s' \otimes \mathbf{v} := \mathbf{n}, s' \otimes \mathbf{v} := \mathbf{n})$—from which it follows that $\mathbf{n}$ is even, by definition of $\mathcal{R}$. Therefore (14) never holds for odd $\mathbf{n}$, and hence $w \vdash C_3[P/p], s \Uparrow_{cmd}$, as required. □

4.3. Example (Tennent [10]). Let

$$C_4 \stackrel{\text{def}}{=} \mathbf{new}\ x := \mathbf{0}\ \mathbf{in}\ p\,(x := !x + \mathbf{1})\,(!x)\ \mathbf{end}$$

$$C_5 \stackrel{\text{def}}{=} \mathbf{new}\ x := \mathbf{0}\ \mathbf{in}\ p\,(x := !x - \mathbf{1})\,(-!x)\ \mathbf{end}$$

Then $p : cmd \to (int \to cmd) \vdash C_4 \cong_{cmd} C_5$.

Proof. By the Operational Extensionality Theorem 2.5, it suffices to show for all worlds w, all terms $P \in IA_{cmd\to(int\to cmd)}(w)$, and all states $s, s' \in States(w)$, that

$$w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\mathbf{v} := !\mathbf{v} + \mathbf{1})\,(!\mathbf{v}) \Downarrow s' \otimes \mathbf{v} := \mathbf{n} \quad (15)$$

holds for some $\mathbf{n}$ if and only if for some $\mathbf{n}'$

$$w\mathbf{v} \vdash (s \otimes \mathbf{v} := \mathbf{0}); P\,(\mathbf{v} := !\mathbf{v} - \mathbf{1})\,(-!\mathbf{v}) \Downarrow s' \otimes \mathbf{v} := \mathbf{n}'. \quad (16)$$

Letting $\mathcal{R} \in Rel(\{\mathbf{v}\})$ be

$$\{(s_1, s_2) \mid s_1 = \bot = s_2 \vee s_1(\mathbf{v}) = -s_2(\mathbf{v})\},$$

we have that

$$w\mathbf{v} \vdash \mathbf{v} := !\mathbf{v} + \mathbf{1}\ (\mathcal{I}d_w \otimes \mathcal{R})_{cmd}\ \mathbf{v} := !\mathbf{v} - \mathbf{1}$$
$$w\mathbf{v} \vdash !\mathbf{v}\ (\mathcal{I}d_w \otimes \mathcal{R})_{int}\ -!\mathbf{v}$$
$$(s \otimes \mathbf{v} := \mathbf{0}, s \otimes \mathbf{v} := \mathbf{0}) \in \mathcal{I}d_w \otimes \mathcal{R}.$$

Since $w \vdash P\ \mathcal{I}d_{cmd \to (int \to cmd)}\ P$, it follows that

$$w \vdash P\ (\mathbf{v} := !\mathbf{v} + \mathbf{1})\ (!\mathbf{v})\ (\mathcal{I}d_w \otimes \mathcal{R})_{cmd}\ P\ (\mathbf{v} := !\mathbf{v} - \mathbf{1})\ (-!\mathbf{v}).$$

Then by definition of $(\mathcal{I}d_w \otimes \mathcal{R})_{cmd}$ and $\mathcal{R}$, if (15) holds for some $\mathbf{n}$, then (16) holds with $\mathbf{n}' = -\mathbf{n}$, and *vice versa*. □

For our final example we combine use of the logical relation with some equational properties of contextual equivalence—namely its congruence property (evident from its definition) and validity of β-conversion (established in Example 2.10(v)). Note in particular that these two properties imply that contextual equivalence is preserved by the operation of substituting terms for identifiers.

4.4. Example (Sieber [20, p 55]). Given any world w, any global variable $\mathbf{v} \notin w$, and any term $P \in IA_{cmd \to cmd}(w\mathbf{v})$, let

$$F \stackrel{\text{def}}{=} \lambda n : int.\ \mathbf{new}\ x := \mathbf{0}\ \mathbf{in}\ P\ (x := \mathbf{1}\ ;\ \mathbf{v} := n)\ ;\ \mathbf{if}\ !x = \mathbf{1}\ \mathbf{then}\ \bot_{cmd}\ \mathbf{else}\ \mathbf{skip}\ \mathbf{end}$$

Then for any world $w' \supseteq w\mathbf{v}$ and terms $N, N' \in IA_{int}(w')$, one has $w' \vdash F\ N \cong_{cmd} F\ N'$.

Proof. Let $G \stackrel{\text{def}}{=} \lambda n{:}int\,.\,\lambda c{:}cmd\,.\,P\ (c\ ;\ \mathbf{v} := n)$. Applying the substitution property of $\cong$ mentioned above to Example 4.1, we have

$$w' \vdash C_1[G\ N/p] \cong_{cmd} C_2[G\ N/p].$$

Using β-conversion and the congruence property of $\cong$ we deduce that

$$w' \vdash C_1[G\ N/p] \cong_{cmd} F\ N$$
$$w' \vdash C_2[G\ N/p] \cong_{cmd} P\ (\bot_{cmd}\ ;\ \mathbf{v} := N).$$

Now $w' \vdash (\bot_{cmd}\ ;\ \mathbf{v} := N) \cong_{cmd} \bot_{cmd}$ by Example 2.10(ix), and therefore $w' \vdash P\ (\bot_{cmd}\ ;\ \mathbf{v} := N) \cong_{cmd} P\ (\bot_{cmd})$. Putting these facts together, for any $N \in IA_{int}(w')$ we have $w' \vdash F\ N \cong_{cmd} P\ (\bot_{cmd})$ and hence in particular $w' \vdash F\ N \cong_{cmd} F\ N'$, for any $N, N' \in IA_{int}(w')$. □

5. Related and further work

Logical relations on domains have been used for proving program equivalences involving local variables, especially by O'Hearn and Tennent [10], and Sieber [20, 21]. The distinctive feature of the work presented here is that the logical relation is defined directly on the syntax of the language, using an operational semantics rather than a denotational semantics. We claim that this approach can lead to more easily applicable verification methods. The examples given above seem to support this claim, at least as far as proving contextual equivalences is concerned.

It is interesting to compare the results presented here for Algol with the operational methods for reasoning about local state in Scheme-like languages developed by Honsell, Mason, Smith, and Talcott [1]. ML and Scheme combine call-by-value function application with declarations of local state in function expressions. As we mentioned in the Introduction, this can result in very complex properties of contextual equivalence compared with Algol. The root of the problem is that, unlike for Algol, *state grows during evaluation of expressions*—in the sense that the underlying 'world' gets larger. Put another way, the canonical forms to which function expressions evaluate are not simply lambda expressions, but rather expressions of the form

$$\mathbf{new}\ \vec{x} := \vec{\mathbf{n}}\ \mathbf{in}\ \lambda y : \sigma\,.\,M\ \mathbf{end}. \tag{17}$$

In the presence of call-by-value evaluation of function application, such an expression is not necessarily contextually equivalent to a lambda abstraction (one cannot just push the **new** declaration under the λ). Any version of Operational Extensionality for this kind of language has to take account of the fact that two such expressions can be contextually inequivalent even though they give contextually equivalent results when applied to any value. For it may well be that some complicated context can get access to the bound variables $\vec{x}$ and use them in arguments fed to $\lambda y : \sigma\,.\,M$; therefore such a context may produce more observable results than those produced merely by applying (17) to argument values (which, up to α-conversion, do not involve the bound variables $\vec{x}$). See [15, p 130] for an example of this phenomenon.

A technical tool used in [1] is a weak form of extensionality, the (**ciu**) Theorem [*loc. cit.*, 2.3.2]. It allows one to restrict—but only somewhat—the kind of contexts needed to characterise contextual equivalence and is probably the best such result known for this type of local state. By contrast, the Operational Extensionality Theorem presented here (2.5) shows that an extremely restricted collection of contexts (generated by just $[-]\,A$, $![-]$, and $[-] := \mathbf{n}$) suffices to characterise contextual equivalence for Algol-like languages.

Honsell *et al* develop a number of proof principles in [*loc. cit.*] somewhat tailored to the invariance properties

of local state in (Scheme analogues of) particular Meyer-Sieber example equivalences. By contrast, these and other equivalences were proved above in quite a uniform way—by picking suitable instances of the state relation parameter of the logical relation. The extension of this operationally-based logical relation method to the harder case of call-by-value functions with local state will be described in [14]. Although the parametric logical relation presented there is merely sound for contextual equivalence (*i.e.* the relation $w \vdash - \; (\mathcal{I}d_w)_\sigma \; -$ is contained in the relation $w \vdash - \cong_\sigma -$, but is not equal to it), it seems to provide a very useful tool for reasoning about the rather complicated behaviour that such functions can have. Both here and in [14] we restrict to *simply typed* languages, because we rely heavily upon induction over the structure of types when defining the logical relation. For untyped languages, or ones with type-reflexive features (such as storage of higher-order values, or recursively defined datatypes), the mere existence of such logical relations is problematic: we expect to adapt the denotational techniques in [13, Sect. 4] to tackle this extension.

References

[1] F. Honsell, I. A. Mason, S. F. Smith, and C. L. Talcott. A variable typed logic of effects. *Information and Computation*, 119(1):55–90, May 1995.

[2] D. J. Howe. Equality in lazy computation systems. In *4th Annual Symposium on Logic in Computer Science*, pages 198–203. IEEE Computer Society Press, Washington, 1989.

[3] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, Feb. 1996.

[4] I. A. Mason, S. F. Smith, and C. L. Talcott. From operational semantics to domain theory. *Information and Computation*. To appear. Revised and extended version of [22].

[5] A. Meyer and K. Sieber. Towards fully abstract semantics for local variables. In *Proc. 15th Symp. on Principles of Programming Languages, San Diego*, pages 191–203. ACM, 1988.

[6] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[7] J. C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*, pages 365–458. North-Holland, Amsterdam, 1990.

[8] P. W. O'Hearn and U. S. Reddy. *Objects, Interference and the Yoneda Embedding*, volume 1 of *Electronic Notes in Computer Science*. Elsevier Science B. V., 1995. Mathematical Foundations of Programming Semantics, Eleventh Annual Conference, Tulane University New Orleans, LA, April 1995.

[9] P. W. O'Hearn and J. C. Reynolds. From algol to polymorphic linear lambda calculus. lectures at the Isaac Newton Institute for Mathematical Sciences, Cambridge UK, August 1995.

[10] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *Journal of the ACM*. To appear.

[11] F. J. Oles. Types algebras, functor categories and block structure. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, chapter 15, pages 543–574. Cambridge University Press, 1985.

[12] A. M. Pitts. Operationally-based theories of program equivalence. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*. Cambridge University Press. Based on lectures given at the CLICS-II Summer School on Semantics and Logics of Computation, Isaac Newton Institute for Mathematical Sciences, Cambridge UK, September 1995.

[13] A. M. Pitts. Relational properties of domains. *Information and Computation*. To appear. A preliminary version appeared as Cambridge Univ. Computer Laboratory Technical Report Number 321, December, 1993.

[14] A. M. Pitts and I. D. B. Stark. Operational reasoning for functions with local state. In A. D. Gordon and A. M. Pitts, editors, *Higher Order Operational Techniques in Semantics*. To appear.

[15] A. M. Pitts and I. D. B. Stark. Observable properties of higher order functions that dynamically create local names, or: What's new? In *Mathematical Foundations of Computer Science, Proc. 18th Int. Symp., Gdańsk, 1993*, volume 711 of *Lecture Notes in Computer Science*, pages 122–141. Springer-Verlag, Berlin, 1993.

[16] U. S. Reddy. Global state considered unnecessary: Introduction to object-based semantics. *Lisp and Symbolic Computation*, 1995. special issue on State in Programming Languages, to appear.

[17] J. C. Reynolds. The essence of Algol. In J. W. de Bakker and J. C. van Vliet, editors, *Algorithmic Languages. Proceedings of the International Symposium on Algorithmic Languages*, pages 345–372. North-Holland, Amsterdam, 1981.

[18] E. Ritter and A. M. Pitts. A fully abstract translation between a λ-calculus with reference types and Standard ML. In *2nd Int. Conf. on Typed Lambda Calculus and Applications, Edinburgh, 1995*, volume 902 of *Lecture Notes in Computer Science*, pages 397–413. Springer-Verlag, Berlin, 1995.

[19] D. S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121:411–440, 1993.

[20] K. Seiber. Full abstraction for the second order subset of an ALGOL-like language. Technical Report A 04/95, Fach. Informatik, Univ. des Saarlandes, Saarbrücken, Germany, Apr. 1995. To appear in Information & Computation.

[21] K. Sieber. Full abstraction via logical relations. Habilitationsschrift, FB 14 Informatik, Universität des Saarlandes, July 1995.

[22] S. F. Smith. From operational to denotational semantics. In S. B. *et al*, editor, *7th International Conference on Mathematical Foundations of Programming Semantics, Pittsburgh PA*, volume 598 of *Lecture notes in Computer Science*, pages 54–76. Springer-Verlag, Berlin, 1992.

[23] I. D. B. Stark. Names and higher-order functions. Technical Report 363, Cambridge Univ. Computer Laboratory, Apr. 1995.