Relating step-indexed logical relations and bisimulations

Dimitrios Vytiniotis
Microsoft Research, Cambridge
dimitris@microsoft.com

Vasileios Koutavas
Trinity College, Dublin
Vasileios.Koutavas@cs.tcd.ie

March 12, 2009

Abstract

Operational logical relations and bisimulations are two particularly successful syntactic techniques for reasoning about program equivalence. Although both techniques seem to have common intuitions, their basis is on different mathematical principles: induction for the former, and co-induction for the latter. The intuitive understanding of the two techniques seems more common, but their mathematical connection more ambitious, when each is combined with step-based reasoning, such as in the case of Appel-McAllester-Ahmed step-indexed (SI) logical relations [5, 4] and Koutavas-Wand (KW) bisimulations [12, 11].

In this paper we give an alternative formulation of a SI logical relation in the style of Appel-McAllester-Ahmed. We derive this from a definition that is parametric on the indexing scheme by requiring it to satisfy the desirable properties of a SI logical relation. We then argue that SI logical relations and KW bisimulations approximate the same relation each in a distinct way. Finally we prove a somewhat surprising commutation theorem between unions and intersections that may be used as a new proof technique.

1 Introduction

Logical relations based on operational semantics (e.g. [15, 16]), and bisimulations (e.g. [1, 12, 17, 20]) are two particularly successful syntactic techniques for reasoning about program equivalence. They have been used to show the correctness of compiler transformations [2, 21], representation independence [15], and the adherence of various implementations to their functional specifications.

Operational logical relations, defined by induction on types, are an extensional interpretation of the types of a programming language as binary relations between closed expressions. A soundness theorem ensures that this interpretation is a relation that is contained in contextual equivalence. To show equivalence, the user of the method simply has to show that two terms are related by the logical relation.

On the other hand, bisimulations provide a co-inductive way of showing equivalence. This method characterizes equivalence as the greatest fixpoint of an appropriate functional. To show equivalence of two terms, the user of the method has to come up with a relation that contains these terms and satisfies certain consistency conditions (i.e. is a bisimulation). Co-induction gives the containment of this relation in contextual equivalence.

Operational logical relation definitions become more complicated once recursive types are added to the language, as these definitions can no longer use induction on the size of types [7, 8, 14]. A particular way around this problem is to introduce a different metric, related to the number of evaluation steps that are required for an adversary to distinguish between two expressions [5, 4]; in the jargon step-indexed (SI) logical relations.

On the bisimulation side, Koutavas and Wand have observed that, once a bisimulation-based technique is "broken down" to use a similar evaluation-step metric, it becomes easier to use. A similar technique has been proposed by Sumii and Pierce ([18], Section 7). Such tech-

niques are particularly useful for proving a number of complex equivalences involving higher-order functions (e.g. see examples in [11]).

Contributions

In this paper we formulate a SI logical relation for a language with recursive types which is parametric to the indexing scheme. We then derive sufficient conditions on the indexing scheme for the relation to satisfy the *fundamental theorem* of logical relations (the inclusion of syntactic identity in our operational setting). We show that these derived conditions force the indexing scheme to be exactly the indexing scheme of Appel-McAllester-Ahmed.

We investigate then the "microscopic" connection between the logical relation at index k and the KW bisimulations at index k and prove that the logical relation relates values not related by any KW bisimulation at index k. We conjecture that KW bisimulations at index k are contained within the logical relation at index k but we were unable to prove that so far.

We then examine the "macroscopic" connection between the two methods. We show that if two elements are related in KW-bisimulations for every k then they are related by the logical relation. This gives rise to a commuting property between unions and intersections which technically corresponds to an alternative proof technique.

2 Background

Throughout our development we will be using a simply typed λ-calculus with recursive types. The types and terms of the calculus are given in Figure 1. Types σ, τ include type variables, integer type, function types, and recursive types. Terms e include the standard constructs of the λ-calculus, terms for introducing and eliminating recursive types, integer literals i, and an elimination form for integer literals that checks two integer expressions for equality. We assume a call-by-value operational semantics, and our value forms include literals, abstractions, and $(\texttt{fold}\ v)$ terms, where v is a value.

We write $e_1 \longrightarrow e_2$ to mean a single step of reduction, $e_1 \longrightarrow^* e_2$ to mean zero or more steps, and $e_1 \longrightarrow^k e_2$ to mean k-steps of reduction. We also write $\mathsf{value}(e)$ when e is a closed value. We define $e_1 \longrightarrow^0 e_2$ to mean that $e_1 = e_2$. The only interesting rule for reduction is the one for a $\texttt{unfold}\ (\texttt{fold}\ v)$ expression below.

$$\frac{}{\texttt{unfold}\ (\texttt{fold}\ v) \longrightarrow v}\ \text{E-FOLD}$$

The semantics of the $\texttt{if}\ e_1 = e_2\ \texttt{then}\ e_3\ \texttt{else}\ e_4$ expression is straightforward. Once e_1 and e_2 are reduced to some integer literals i_1 and i_2 (first e_1 and then e_2), the expression returns either e_3 if $i_1 = i_2$ and e_4 otherwise. We omit the rules for reasons of space. Notice that the language does not include recursive functions, by we can recover recursion via recursive types, so expressions may well diverge.

The static semantics of the language is also standard (see for example TAPL [13], Chapter 20). The interesting rules concerning recursive types are given below.

$$\frac{\Gamma \vdash e : \tau\{\mu a.\tau/a\}}{\Gamma \vdash \texttt{fold}\ e : \mu a.\tau}\ \text{T-FOLD}$$

$$\frac{\Gamma \vdash e : \mu a.\tau}{\Gamma \vdash \texttt{unfold}\ e : \tau\{\mu a.\tau/a\}}\ \text{T-UNFOLD}$$

For this language it is not hard to show type soundness (which we omit here), via subject reduction and progress.

Proposition 2.1 (Subject reduction)**.** *If* $\Gamma \vdash e : \tau$ *and* $e \longrightarrow e'$ *then* $\Gamma \vdash e' : \tau$.

Proposition 2.2 (Progress)**.** *If* $\vdash e : \tau$ *then either* $value(e)$ *or there exists an* e' *such that* $e \longrightarrow e'$.

In the rest of this document we will be working with typed binary relations on terms.

Definition 2.3 (Term relations)**.** *A relation* r *is a term relation, and we write* $r \in \texttt{Rel}$*, if and only if for all* $(e_1, e_2, \tau) \in r$ *it is* $\vdash e_{1,2} : \tau$.

Definition 2.4 (Value relations)**.** *A term relation* r *is a value relation, and we write* $r \in \texttt{VRel}$*, if and only if for all* $(e_1, e_2, \tau) \in r$*, it is* $\mathsf{value}(e_{1,2})$.

$$
\begin{array}{rcl}
\sigma, \tau & ::= & a \mid \mathtt{int} \mid \tau \to \tau \mid \mu a.\tau \\
e & ::= & x \mid \lambda x{:}\tau.e \mid e\ e \mid \mathtt{fold}\ e \mid \mathtt{unfold}\ e \mid i \mid \mathtt{if}\ e_1 = e_2\ \mathtt{then}\ e_3\ \mathtt{else}\ e_4 \\
u, v, w & ::= & i \mid \lambda x{:}\tau.e \mid \mathtt{fold}\ u
\end{array}
$$

Figure 1: Syntax

2.1 The Koutavas-Wand method

The Koutavas-Wand bisimulation method [11, 12] amounts to transforming the requirement for a co-inductive proof of equivalence to use an inductive argument, based on the steps of evaluation. We outline and explain below the main ideas and definitions.

Definition 2.5 (Contextual closure)**.** *Let $r \in \mathtt{VRel}$. We define the* contextual closure *of r, written r^{cxt}, as follows:*

$$
\begin{array}{rcl}
r^{\mathsf{cxt}} & = & \{(e\{\overline{v/x}\}, e\{\overline{v'/x}\}, \sigma) \mid \\
& & \qquad \overline{x{:}\tau} \vdash e : \sigma \wedge \overline{(v, v', \tau)} \in r\}
\end{array}
$$

Hence, the contextual closure of a relation contains all those contexts in which we have plugged in related values.

The Koutavas-Wand method uses then a standard *evaluation closure* functional that yields a relation consisting of pairs of terms such that if the first converges, the second converges and the resulting values are related in the argument relation. Notice that the evaluation closure functional that we consider here is *not symmetric*, in order to compare with the standard definition of SI logical relations.

Definition 2.6 (Evaluation closure functional)**.** *We let $F \in \mathtt{VRel} \to \mathtt{VRel}$ be defined as in Figure 2.*

Subsequently *adequate* relations are defined to be those satisfying $r^{\mathsf{cxt}} \subseteq F(r^{\mathsf{cxt}})$. The union of all adequate relations (*adequacy*) can be shown to be itself adequate, and prove that adequacy coincides with contextual approximation.

Definition 2.7. *[Adequacy]* $(\precsim) = \bigcup_{r^{\mathsf{cxt}} \subseteq F(r^{\mathsf{cxt}})} r$

Proposition 2.8. $(\precsim)^{\mathsf{cxt}} \subseteq F((\precsim)^{\mathsf{cxt}})$

Proposition 2.9 (Soundness and completeness)**.** *Contextual approximation coincides with* $(\precsim)$.

The above proposition gives a co-inductive argument to establish contextual approximation. In order to show that v_1 approximates v_2 , it suffices to find a relation r for which $(v_1, v_2, \tau) \in r$ and $r^{\mathsf{cxt}} \subseteq F(r^{\mathsf{cxt}})$.

To transform such a requirement into an inductive one, Koutavas and Wand introduce a *step-indexed evaluation closure* functional.

Definition 2.10 (Step-indexed evaluation closure functional)**.** *For any $k \in I\!N$, we let $F_k \in \mathtt{VRel} \to \mathtt{VRel}$ be defined as in Figure 2.*

F_k is similar to F, except that it allows reduction only up to k steps. It follows easily that $\bigcap_{k \in \omega} F_k(r) = F(r)$. Consequently:

Proposition 2.11. $(\precsim) = \bigcup_{\forall k, r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$

Thus, in order to show that $r^{\mathsf{cxt}} \subseteq F(r^{\mathsf{cxt}})$, it suffices to show that $r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})$ for every k. But this obligation can be shown by induction on k. The proof can now use inductive hypotheses for smaller k, which makes the method more useful than direct attempts to prove $r^{\mathsf{cxt}} \subseteq F(r^{\mathsf{cxt}})$.

2.2 Step-indexed logical relations

On the other hand, logical relations are typically defined by induction on the structure of types, and relate terms in an extensional way. For example two values of function type are related if and only if for all related arguments, the results are related. A soundness theorem establishes the fact that the logical relation is contained in contextual equivalence.

In the presence of recursive types we would wish to relate two values $\mathtt{fold}\ v_1$ and $\mathtt{fold}\ v_2$ at type $\mu a.\tau$ if v_1 and

$$
\begin{array}{rcl}
F(r) & = & \{(e_1, e_2, \tau) \mid \cdot \vdash e_{1,2} : \tau \\
& & \quad \wedge \forall v_1, (e_1 \longrightarrow^* v_1) \Longrightarrow \exists v_2, (e_2 \longrightarrow^* v_2) \wedge ((v_1, v_2, \tau) \in r)\} \\
F_k(r) & = & \{(e_1, e_2, \tau) \mid \cdot \vdash e_{1,2} : \tau \\
& & \quad \wedge \forall j \leq k, \forall v_1, (e_1 \longrightarrow^j v_1) \Longrightarrow \exists v_2, (e_2 \longrightarrow^* v_2) \wedge ((v_1, v_2, \tau) \in r)\}
\end{array}
$$

Figure 2: Evaluation closure functionals

v_2 are related at type $\tau\{\mu a.\tau/a\}$. However, such a definition can no longer be given by induction on the size (or structure) of types, as the type $\tau\{\mu a.\tau/a\}$ is potentially larger than $\mu a.\tau$.

Step-indexed logical relations were invented to address this problem. Instead of defining the logical relation by induction on the size of types, we introduce a "step" index and define the logical relation by induction on that index. The intuition is now that if v_1 and v_2 are related at type τ and *at index* k then v_1 approximates v_2 in program contexts that run for at most k steps. If two values are related at every k then they belong in contextual approximation.

Perhaps surprisingly, we show that for the SI logical relation to satisfy certain soundness conditions, the indexing scheme to be used in the definition can be derived. In fact, in the next section we start with a generalized SI logical relation and we show that in order for the definition to satisfy the fundamental property of the logical relation, the definition has to be the one of Appel-McAllester-Ahmed.

3 Generalized SI logical relation

We give the definition of the SI logical relation in Figure 3. As is common in the literature, we distinguish between value relations at index k ($\mathcal{V}_k \in \texttt{VRel}$) and term relations at index k ($\mathcal{C}_k \in \texttt{Rel}$). For $k = 0$ we have chosen to relate all values in $\mathcal{V}_0$. For $k > 0$, we have several cases to consider, depending on the type of the related expressions. In the case of integers we only relate two equal integer literals. For the case of function types $\tau_1 \to \tau_2$, if we apply the function values $\lambda x{:}\tau_1.e_1$ and $\lambda x{:}\tau_1.e_2$ to related arguments w_1 and w_2 at a smaller index j, the results must be terms related in the smaller index j, given with $\mathcal{C}_j$. For recursive types, two values $\texttt{fold}\ v_1$ and $\texttt{fold}\ v_2$ are related if v_1 and v_2 are related for all smaller indices.

The first property that we need to establish about this definition is that it is a well-formed definition by induction on the index argument. We do this by checking that subcalls to $\mathcal{V}_k$ use smaller indices. In order to do this we may inline the use of $\mathcal{C}_j$ inside the definition of $\mathcal{V}_k$. We hence obtain the well-formedness condition:

Condition 3.1. *For all j, n it must be $g(j, n) \leq j$.*

Moreover, as expected the definition gives smaller sets as the indices become larger.

Lemma 3.2. $\mathcal{V}_{k+1} \subseteq \mathcal{V}_k$

Proof. Let $(v_1, v_2, \tau) \in \mathcal{V}_{k+1}$. If $k = 0$ we are trivially done. If $k > 0$, by the definition of $\mathcal{V}_{k+1}$ the triple (v_1, v_2, τ) can be of the following shapes:

- $(i, i, \texttt{int})$. Then trivially $(i, i, \texttt{int}) \in \mathcal{V}_k$.
- $(\lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2, \tau_1 \to \tau_2)$. Then pick some j with $j < k$. Pick $(w_1, w_2, \tau_1) \in \mathcal{V}_j$. We need to show that $(e_1\{w_1/x\}, e_2\{w_2/x\}, \tau_2) \in \mathcal{C}_j$. But we know that $(\lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2, \tau_1 \to \tau_2) \in \mathcal{V}_{k+1}$. Because it is also $j < k + 1$ we get $(e_1\{w_1/x\}, e_2\{w_2/x\}, \tau_2) \in \mathcal{C}_j$ as required.
- $(\texttt{fold}\ v_1, \texttt{fold}\ v_2, \mu a.\tau)$. Straightforward.

□

$$
\begin{array}{rcl}
\mathcal{V}_k \mid k = 0 & = & \{(v_1, v_2, \tau) \mid \cdot \vdash v_{1,2} : \tau\} \\
\mathcal{V}_k \mid k > 0 & = & \{(i, i, \texttt{int})\} \\
& \cup & \{(\lambda x{:}\tau_1.e_1, \lambda x{:}\tau_1.e_2, \tau_1 \to \tau_2) \mid \cdot \vdash \lambda x{:}\tau_1.e_{1,2} : \tau_1 \to \tau_2 \wedge \\
& & \qquad \forall j < k, \forall w_1, w_2, (w_1, w_2, \tau_1) \in \mathcal{V}_j \Longrightarrow (e_1\{w_1/x\}, e_2\{w_2/x\}) \in \mathcal{C}_j\} \\
& \cup & \{(\texttt{fold}\ v_1, \texttt{fold}\ v_2, \mu a.\tau) \mid \cdot \vdash v_{1,2} : \mu a.\tau \wedge \\
& & \qquad \forall j < k, (v_1, v_2, \tau\{\mu a.\tau/a\}) \in \mathcal{C}_j\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{C}_k & = & \{(e_1, e_2, \tau) \mid \vdash e_{1,2} : \tau \wedge \\
& & \qquad \forall j \leq k, (e_1 \longrightarrow_j v_1 \Longrightarrow e_2 \longrightarrow^* v_2 \wedge (v_1, v_2, \tau) \in \mathcal{V}_{g(k,j)})\}
\end{array}
$$

Figure 3: Generalized SI logical relation

3.1 Fundamental property

We move on to show the fundamental property of the logical relation. In our setting this amounts to proving that if $\cdot \vdash e : \tau$ then $(e, e, \tau) \in \bigcap_k \mathcal{C}_k$.To prove it, we have to generalize the statement slightly.

Definition 3.3 (Environment-respecting substitutions)**.** *Let γ be a substitution from variables to triples of the form (v_1, v_2, τ). We write $\gamma \models_k \Gamma$ if and only if*

$$\forall (x{:}\tau) \in \Gamma, \gamma(x) = (v_1, v_2, \tau) \in \mathcal{V}_k$$

When $\gamma(x) = (v_1, v_2, \tau)$ we abbreviate $\gamma^1(x) = v_1$ and $\gamma^2(x) = v_2$.

Theorem 3.4 (Fundamental Theorem)**.** *If $\Gamma \vdash e : \tau$ then for all $\gamma \models_k \Gamma$ it is $(\gamma^1 e, \gamma^2 e, \tau) \in \mathcal{C}_k$.*

Proof. By induction on the typing derivation of $\Gamma \vdash e : \tau$.

- Case $\Gamma \vdash i : \texttt{int}$. Then $(i, i, \texttt{int}) \in \mathcal{V}_j$ for any j and hence also for $j = g(k, 0)$ as is required to show that $(i, i, \texttt{int}) \in \mathcal{C}_k$.
- Case $\Gamma \vdash x : \sigma$ given that $(x{:}\sigma) \in \Gamma$. By assumptions $(\gamma^1 x, \gamma^2 x, \tau) \in \mathcal{V}_k$. To finish the case we need to show that $(\gamma^1 x, \gamma^2 x, \sigma) \in \mathcal{V}_{g(k,0)}$ which will be true if $g(k, 0) \leq k$, which holds by Condition 3.1.
- Case $\Gamma \vdash \lambda x{:}\tau_1.e : \tau_1 \to \tau_2$ given that $\Gamma, (x{:}\tau_1) \vdash e : \tau_2$. Fix $j < g(k, 0)$ (hence also $j < k$). We know that $\gamma \models_k \Gamma$ hence also $\gamma \models_j \Gamma$ because $j < k$. Fix $(w_1, w_2, \tau_1) \in \mathcal{V}_j$. It follows that for $\gamma_x = \gamma, (x \mapsto (w_1, w_2, \tau_1))$ it is $\gamma_x \models_j \Gamma, (x{:}\tau_1)$. Hence, by induction $(\gamma_x^1 e, \gamma_x^2 e, \tau_2) \in \mathcal{C}_j$. Hence we get that $((\gamma^1 e)\{w_1/x\}, (\gamma^2 e)\{w_2/x\}, \tau_2) \in \mathcal{C}_j$ as required.
- Case $\Gamma \vdash e_1\ e_2 : \sigma_2$ given that $\Gamma \vdash e_1 : \sigma_1 \to \sigma_2$ and $\Gamma \vdash e_2 : \sigma_1$. By induction hypothesis we get that:

$$(\gamma^1 e_1, \gamma^2 e_1, \sigma_1 \to \sigma_2) \in \mathcal{C}_k \quad (1)$$
$$(\gamma^1 e_2, \gamma^2 e_2, \sigma_1) \in \mathcal{C}_k \quad (2)$$

We need to show that $(\gamma^1(e_1\ e_2), \gamma^2(e_1\ e_2), \sigma_2) \in \mathcal{C}_k$. Fix $j \leq k$. Let $(\gamma^1 e_1)\ (\gamma^1 e_2) \longrightarrow^j z_1$. This means that:

$$\gamma^1 e_1 \longrightarrow^{l_1} \lambda x{:}\sigma_1.e_1' \quad (3)$$
$$\gamma^1 e_2 \longrightarrow^{l_2} w_1 \quad (4)$$
$$e_1'\{w_1/x\} \longrightarrow^{l_3} z_1 \quad (5)$$

such that $l_1 + l_2 + l_3 + 1 = j$. Hence $l_1, l_2, l_3 < k$. Since $l_1 \leq k$, by (1) we get that $\gamma^2 e_1 \longrightarrow^\star \lambda x{:}\sigma_1.e_2'$ and moreover $(\lambda x{:}\sigma_1.e_1', \lambda x{:}\sigma_1.e_2', \sigma_1 \to \sigma_2) \in \mathcal{V}_{g(k,l_1)}$. Similarly, by (2) we get that $\gamma^2 e_2 \longrightarrow^\star w_2$ and $(w_1, w_2, \sigma_1) \in \mathcal{V}_{g(k,l_2)}$. It suffices then to find

an index n such that:

$$n < g(k, l_1) \quad (6)$$
$$n \leq g(k, l_2) \quad (7)$$
$$l_3 \leq n \quad (8)$$
$$g(n, l_3) \geq g(k, l_1 + l_2 + l_3 + 1) \quad (9)$$

The reason is because in this case it must be that $(w_1, w_2, \sigma_1) \in \mathcal{V}_n$ and it must be that $(e_1'\{w_1/x\}, e_2'\{w_2/x\}, \sigma_2) \in \mathcal{C}_n$. Because $l_3 \leq n$ it must be that $(z_1, z_2, \sigma_2) \in \mathcal{V}_{g(n,l_3)}$ which is contained in $\mathcal{V}_{g(k,l_1+l_2+l_3+1)}$ if $g(n, l_3) \geq g(k, l_1 + l_2 + l_3 + 1)$ holds.

- Case $\Gamma \vdash \texttt{fold}\ e : \mu a.\tau$ given that $\Gamma \vdash e : \tau\{\mu a.\tau/a\}$. By induction we have that $(\gamma^1 e, \gamma^2 e, \tau\{\mu a.\tau/a\}) \in \mathcal{C}_k$. We need to show that $(\texttt{fold}\ \gamma^1 e, \texttt{fold}\ \gamma^2 e, \mu a.\tau) \in \mathcal{C}_k$. Fix $j \leq k$ and assume that $\texttt{fold}\ \gamma^1 e \longrightarrow^j v_1$. It must be then that $\gamma^1 e \longrightarrow^j w_1$ and $v_1 = \texttt{fold}\ w_1$. By induction then $\gamma^2 e \longrightarrow^\star w_2$ and $(w_1, w_2, \tau\{\mu a.\tau/a\})$ in $\mathcal{V}_{g(k,j)}$. Hence it must be that for all $l < g(k, j)$ it is $(w_1, w_2, \tau\{\mu a.\tau/a\}) \in \mathcal{V}_l$, which means that $(\texttt{fold}\ w_1, \texttt{fold}\ w_2, \mu a.\tau) \in \mathcal{V}_{g(k,j)}$ and we are done.
- Case $\Gamma \vdash \texttt{unfold}\, e : \tau\{\mu a.\tau/a\}$ given that $\Gamma \vdash e : \mu a.\tau$. By induction $(\gamma^1 e, \gamma^2 e, \mu a.\tau) \in \mathcal{C}_k$. We want to show that $(\texttt{unfold}\, \gamma^1 e, \texttt{unfold}\, \gamma^2 e, \tau\{\mu a.\tau/a\}) \in \mathcal{C}_k$. Pick $j \leq k$ and assume that $\texttt{unfold}\, \gamma^1 e \longrightarrow^{j-1} \texttt{unfold}\, (\texttt{fold}\ w_1) \longrightarrow w_1$. It follows that $\gamma^2 e \longrightarrow^\star \texttt{fold}\ w_2$ and $(w_1, w_2, \tau\{\mu a.\tau/a\}) \in \mathcal{V}_{g(k,j-1)}$. To finish the case we need to show that $(w_1, w_2, \tau\{\mu a.\tau/a\}) \in \mathcal{V}_{g(k,j)}$ which will follow if g is decreasing in its second argument.
- Case $\texttt{if}\ e_1 = e_2\ \texttt{then}\ e_3\ \texttt{else}\ e_4$ is straightforward.

□

Hence, for the fundamental theorem to be true, we have required certain conditions on g. These derived conditions on g are given in Figure 4.

Under the previous conditions we have then the following corollary:

Corollary 3.5. *If $\cdot \vdash e : \tau$ then for all k, $(e, e, \tau) \in \mathcal{C}_k$.*

It follows that the identity is contained in $\bigcap_k \mathcal{C}_k$.

For example, if we take $g(k, n) = k - n$, conditions C1, C2, C3 are all satisfied by taking $x = k - l - n - 1$ (which is ≥ 0 by assumptions). Moreover, notice that if we take $g(k, n) = k$ we can no longer satisfy C3. If we take $g(k, n) = n$ we can no longer satisfy C2. In fact, the derived conditions fully determine g, as the next theorem shows.

Theorem 3.6. *If the conditions in Figure 4 are true of g, then for all $n \leq k$ it is $g(k, n) = k - n$.*

Proof. We first show some intermediate results. The first is that $g(k, 0) = k$. For this let us pick $l_1 = l_2 = 0$ and $l_3 = k - 1$. Then we get that there exists an x such that $x < g(k, 0)$ and $x \geq k-1$. Hence $k-1 \leq x < g(k, 0) \leq k$. Which means that $k - 1 < g(k, 0) \leq k$, and hence

$$g(k, 0) = k \quad (10)$$

Moreover in that case $x = k - 1$, which also means that $g(k - 1, k - 1) \geq g(k, k)$ and by induction $g(0, 0) \geq g(k, k)$. But $g(0, 0) \leq 0$ and hence $g(0, 0) = 0$. Consequently:

$$g(k, k) = 0 \quad (11)$$

With the above in place we now show that g is *strictly* decreasing in its second argument. Pick $l_2 = l_3 = 0$. Then we require the existence of an x such that:

$$x \leq g(k, 0)$$
$$x < g(k, l_1)$$
$$0 \leq x$$
$$g(x, 0) \geq g(k, l_1 + 1)$$

Therefore (and using (10)) $x \leq k$, $x < g(k, l_1)$, $x \geq g(k, l_1 + 1)$. It follows that

$$g(k, l_1 + 1) \leq x < g(k, l_1)$$

and therefore g is strictly decreasing in its second argument.

Hence we have that $g(k, 0) = k$, $g(k, k) = 0$ and g strictly decreasing in its second argument. Therefore, by the pigeonhole principle it must be that $g(k, n) = k - n$ for all $n \leq k$. □

(C1)	For well-formedness:	$g(k,0) \leq k$
(C2)	For the recursive types case:	$n_1 \leq n_2 \implies g(k,n_2) \leq g(k,n_1)$
(C3)	For the application case:	$l_1 + l_2 + l_3 + 1 \leq k \implies$ $\exists x, x < g(k,l_1) \wedge x \leq g(k,l_2) \wedge l_3 \leq x \wedge g(x,l_3) \geq g(k,l_1+l_2+l_3+1)$

Figure 4: Derived conditions on g

In what follows we will simply be using $g(k,n) = k - n$ as the definition of the SI logical relation. This algebraic derivation coincides with the following intuition.

Assume that we have values $(v_1, v_2, \tau\{\mu a.\tau\}) \in \mathcal{V}_k$. The definition relates $(\texttt{fold}\ v_1, \texttt{fold}\ v_2, \mu a.\tau) \in \mathcal{V}_{k+1}$. For the fundamental property to be true, it must be that $(\texttt{unfold}\,(\texttt{fold}\ \ v_1), \texttt{unfold}\,(\texttt{fold}\ \ v_2), \tau\{\mu a.\tau\}) \in \mathcal{C}_{k+1}$. Then, assuming $j \leq k+1$ and $\texttt{unfold}\,(\texttt{fold}\ \ v_1) \rightarrow^j v_1$, it must be that $\texttt{unfold}\,(\texttt{fold}\ \ v_2) \rightarrow^\star v_2$. Furthermore the resulting values must be related in an appropriate $\mathcal{V}_k$. Since $j = 1$ the values should be related in $\mathcal{V}_k$. Taking $g(k,n) = k - n$ gives us exactly that.

3.2 Transitivity of logical relation

Lemma 3.7 (Transitivity)**.** *If* $(v_1, v_2, \tau) \in \mathcal{V}_k$ *and* $(v_2, v_3, \tau) \in \bigcap_k \mathcal{V}_k$ *then* $(v_1, v_3, \tau) \in \mathcal{V}_k$.

Proof. We proceed by induction on k. We do a case analysis on τ. The only hard case is that of function types; i.e; $\tau = \tau_1 \rightarrow \tau_2$. In this case let us pick $j < k$ and $(w_1, w_2, \tau_1) \in \mathcal{V}_j$. and let $v_{1,2,3} = \lambda x{:}\tau_1.e_{1,2,3}$. Assume that $e_1\{w_1/x\} \longrightarrow^l z_1$ and $l \leq j$. Then $v_2\{w_2/x\} \longrightarrow^m z_2$ such that $(z_1, z_2) \in \mathcal{V}_{j-l}$. It must then be that $v_3\{w_2/x\} \longrightarrow^\star z_3$ such that $(z_2, z_3) \in \bigcap_k \mathcal{V}_k$. It follows by induction hypothesis that $(z_1, z_3) \in \mathcal{V}_{j-l}$ as required. □

Notice that the above lemma is provably false if F_k is replaced in the definition of $\mathcal{C}_k$ with a symmetric version of F_k.

Additionally, notice that the theorem is also false if the intersection $\bigcap_k \mathcal{V}_k$ is replaced to $\mathcal{V}_k$. Indeed:

$$(\lambda x{:}\texttt{int}.3, \lambda x{:}\texttt{int}.\ \nearrow^{50}; 3) \in \mathcal{V}_{30}$$
$$(\lambda x{:}\texttt{int}.\ \nearrow^{50}; 3, \lambda x{:}\texttt{int}.\ \nearrow^{50}; 4) \in \mathcal{V}_{30}$$
$$(\lambda x{:}\texttt{int}.3, \lambda x{:}\texttt{int}.\ \nearrow^{50}; 4) \notin \mathcal{V}_{30}$$

4 Connections

We now turn our focus to the mathematical connections between SI logical relations and bisimulations. In Section 4.1 we examine whether there exists a connection per-index, and in Section 4.2 we examine the connection between the limits of these relations and *similarity*.

4.1 Microscopic connections

We examine here how that logical relation at index k ($\mathcal{V}_k$) relates to k-adequate relations. The result below is a negative one.

Lemma 4.1. *For every* k *sufficiently large there exist* $(v_1, v_2, \tau) \in \mathcal{V}_k$ *such that* $(v_1, v_2, \tau) \notin r$*, for any* r *which is* k*-adequate.*

Proof. Consider $\nearrow^n$ to be some computation that reduces to a value in n steps and $\nearrow$ to be some divergent computation. We have that

$$(\lambda x{:}\texttt{int}.\ \nearrow^{30}; 0, \lambda x{:}\texttt{int}.\ \nearrow, \texttt{int} \rightarrow \texttt{int}) \in \mathcal{V}_9$$

The reason is because if we apply the two functions to related arguments, the first is going to take more than 9 steps to terminate.

Moreover, there exists no 60-adequate relation that relates the above terms—if there was one, by the definition the application of the second function should converge. Therefore,there exists no 60-adequate relation that contains the following triple.

$$(\lambda x{:}\mathtt{int}.(\nearrow^{50};\lambda y{:}\mathtt{int}.(\nearrow^{30};0)),\\ \lambda x{:}\mathtt{int}.(\nearrow^{50};\lambda y{:}\mathtt{int}.\nearrow), \mathtt{int}\to\mathtt{int}\to\mathtt{int})$$

Nevertheless, the above triple is in $\mathcal{V}_{60}$: if we apply the two to $(i, i, \mathtt{int}) \in \mathcal{V}_{59}$ we have that:

$$(\nearrow^{50};\lambda y{:}\mathtt{int}.(\nearrow^{30};0)) \longrightarrow^{50} \lambda y{:}\mathtt{int}.(\nearrow^{30};0)$$
$$(\nearrow^{50};\lambda y{:}\mathtt{int}.\nearrow) \longrightarrow^{50} \lambda y{:}\mathtt{int}.\nearrow$$
$$(\lambda x{:}\mathtt{int}.\nearrow^{30};0, \lambda x{:}\mathtt{int}.\nearrow, \mathtt{int}\to\mathtt{int}) \in \mathcal{V}_{59-50}$$

□

Corollary 4.2. $\mathcal{V}_k^{\mathsf{cxt}} \not\subseteq F_k(\mathcal{V}_k^{\mathsf{cxt}})$.

We conjecture that the converse is true.
Conjecture 4.3. *If r is k-adequate then $r \subseteq \mathcal{V}_k$.*

The intuition why this conjecture should be true arises from the function case. For two functions to be related at some k we may test with all the arguments which belong in $\mathcal{V}_j$ for $j < k$. Of course some of them may not belong in r (as the previous counterexample shows). But in that case the only way to distinguish between them would be to run them for more than j steps Hence, we may test with less "sound" arguments, but we then give looser conditions on the computations that are the results of the function applications.

4.2 Macroscopic connections

We start with a contextual-closure property of $\mathcal{V}_k$.
Lemma 4.4. $(\mathcal{V}_k)^{\mathsf{cxt}} \subseteq \mathcal{C}_k$

Proof. Assume that $(e_1, e_2, \sigma) \in (\mathcal{V}_k)^{\mathsf{cxt}}$. This means that there exist $\overline{x}, \overline{\tau}, \overline{v_1}, \overline{v_2}$ such that $\overline{x{:}\tau} \vdash e : \sigma$, $(\overline{v_1}, \overline{v_2}, \overline{\tau}) \in \mathcal{V}_k$, $e_1 = e\{\overline{v_1/x}\}$, and $e_2 = e\{\overline{v_2/x}\}$. By the fundamental property (Theorem 3.4) it follows that $(e_1, e_2, \sigma) \in \mathcal{C}_k$. □

We are now able to give some "macroscopic" connections for the limit of the SI logical relation.
Lemma 4.5. $(\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}} \subseteq \bigcap_k \mathcal{C}_k$

Proof. We first prove that if $(e_1, e_2, \sigma) \in (\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}}$ then $(e_1, e_2, \sigma) \in C_k$ for any k. It must be the case that $e_1 = e\{\overline{v_1/x}\}$ and $e_2 = e\{\overline{v_2/x}\}$ for some $\overline{x{:}\tau} \vdash e : \sigma$ and $(\overline{v_1}, \overline{v_2}, \overline{\tau}) \in \bigcap_k \mathcal{V}_k$. Hence it must be that $(\overline{v_1}, \overline{v_2}, \overline{\tau}) \in \mathcal{V}_k$ as well. By Theorem 3.4 then $(e_1, e_2, \overline{\tau}) \in \mathcal{C}_k$. It follows that $(e_1, e_2, \sigma) \in \bigcap_k \mathcal{C}_k$. □

Then we can show that $\bigcap_k \mathcal{V}_k$ is itself k-adequate.
Lemma 4.6. $(\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}} \subseteq F_k((\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}})$

Proof. We know that $(\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}} \subseteq \bigcap_k \mathcal{C}_k$ by Lemma 4.5. Assume that $(e_1, e_2, \sigma) \in \bigcap_k \mathcal{C}_k$. Then, let $e_1 \longrightarrow^j v_1$ with $j \leq k$. This means that $e_2 \longrightarrow^* v_2$ such that $(v_1, v_2, \sigma) \in \mathcal{V}_{k-j}$. Clearly it must be the case that for all $l \leq k - j$ also $(v_1, v_2, \sigma) \in \mathcal{V}_l$ so all we have to show is that it is also the case that $(v_1, v_2, \sigma) \in \mathcal{V}_l$ for all $l > k-j$. But we know that $(e_1, e_2, \sigma) \in \bigcap_k \mathcal{C}_k$ therefore also $(e_1, e_2, \sigma) \in \mathcal{C}_{l+j}$ for the particular l. Since $j \leq l+j$ it must be that $(v_1, v_2, \sigma) \in \mathcal{V}_l$ as well. □

We then show the connection of $\bigcap \mathcal{V}_k$ to $\precsim$, starting off with the following auxiliary lemma.
Lemma 4.7. *If $(v_1, v_2, \tau) \in \mathcal{V}_k$ and $(v_2, v_3, \tau) \in (\precsim)$ then $(v_1, v_3) \in \mathcal{V}_k$.*

Proof. By induction on k. For $k = 0$ it is trivially true. Assume that $k > 0$. We proceed with a case analysis on τ:

- Case $\tau = \mathtt{int}$. Trivial.
- Case $\tau = \tau_1 \to \tau_2$. Then pick $j < k$ and pick $(w_1, w_2, \tau_1) \in \mathcal{V}_j$. It must be that $v_1 = \lambda x{:}\tau_1.e_1$ and $v_2 = \lambda x{:}\tau_1.e_2$. It suffices to show that $(e_1\{w_1/x\}, e_3\{w_2/x\}, \tau_2) \in \mathcal{C}_j$. But we know that $(v_1, v_2, \tau) \in \mathcal{V}_k$ and hence $v_3 = \lambda x{:}\tau_1.e_3$ such that $(e_2\{w_1/x\}, e_3\{w_2/x\}, \tau_2) \in \mathcal{C}_j$. Assume now that $e_1\{w_1/x\} \longrightarrow^l u_1$ such that $l \leq j$. It must be that $e_2\{w_2/x\} \longrightarrow^\star u_2$ such that $(u_1, u_2, \tau_2) \in \mathcal{V}_{j-l}$. We know that $(v_2, v_3, \tau) \in (\precsim)$ and hence it is easy to show that $e_3\{w_2/x\} \longrightarrow^\star u_3$ such that

$(u_2, u_3, \tau_2) \in (\precsim)$. Hence, by induction hypothesis ($j - l < k$) we get that $(u_1, u_3, \tau_3) \in \mathcal{V}_{j-l}$ which finishes the case.

- Case $\tau = \mu a.\tau$. Straightforward by induction hypothesis.

□

Lemma 4.8. $(\precsim) = \bigcap_k \mathcal{V}_k$

Proof. We show the two directions:

- $(\precsim) \subseteq \bigcap_k \mathcal{V}_k$. Assume that $v_1 \precsim v_2 : \tau$. Then by the fundamental theorem $(v_1, v_1) \in \mathcal{V}_k$ and by Lemma 4.7 we get that $(v_1, v_2) \in \mathcal{V}_k$.
- $\bigcap_k \mathcal{V}_k \subseteq (\precsim)$. We know that $\bigcap_k \mathcal{V}_k$ is k-adequate for every k (Lemma 4.6) and hence $(\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}} \subseteq F((\bigcap_k \mathcal{V}_k)^{\mathsf{cxt}})$.

□

5 Union-intersection commutation

In what follows we use the logical relation as an intermediate step to establish a different formulation of the KW bisimulation method.

We start with some auxiliary lemmas.

Lemma 5.1. *Assume an infinite sequence of value relations $r_0 \dots r_i \dots$ such that $r_i^{\mathsf{cxt}} \subseteq F_i(r_i^{\mathsf{cxt}})$. If $(v_1, v_2, \tau) \in \bigcap_k r_k$ and $e\{v_1/x\} \longrightarrow^\star z_1$ then $e\{v_2/x\} \longrightarrow^\star z_2$ and there exists an infinite sequence of value relations $s_0 \dots s_i \dots$ with $s_i^{\mathsf{cxt}} \subseteq F_i(s_i^{\mathsf{cxt}})$ and $(z_1, z_2, \sigma) \in \bigcap_k s_k$.*

Proof. Assume that $e\{v_1/x\} \longrightarrow^m z_1$. We know that $(v_1, v_2) \in r_k$ for all $k \geq m$. It follows, since $r_m^{\mathsf{cxt}} \subseteq F_m(r_m^{\mathsf{cxt}})$ that $e\{v_2/x\} \longrightarrow^\star z_2$ and $(z_1, z_2) \in r_k^{\mathsf{cxt}}$ and hence $(z_1, z_2) \in \bigcap_{k \in m \dots \infty} r_k^{\mathsf{cxt}}$. So it suffices to take $s_k = r_k^{\mathsf{cxt}} \mid_{values}$. It remains to check what to take for s_k where $k \in 0 \dots m$. But we know that $s_m^{\mathsf{cxt}} \subseteq F_m(s_m^{\mathsf{cxt}})$ hence also it must be the case that $s_m^{\mathsf{cxt}} \subseteq F_k(s_m^{\mathsf{cxt}})$ for all $k \leq m$. So just take $s_k = s_m$ for all $k \leq m$. It follows that $(z_1, z_2) \in \bigcap_k s_k$. □

Let us call such an infinite sequence r_k for which it is the case that $r_i^{\mathsf{cxt}} \subseteq F_i(r_i^{\mathsf{cxt}})$ an *adequate sequence*.

Lemma 5.2. *For all k, for all adequate sequences $\overline{r_k}$, for all v_1, v_2, v_3, if $(v_1, v_2) \in \mathcal{V}_k$ and $(v_2, v_3) \in \bigcap_k r_k$ then $(v_1, v_3) \in \mathcal{V}_k$.*

Proof. By induction on k. We will examine the case for function types. The rest cases are easy. Assume that $v_1 = \lambda x{:}\tau_1.e_1$, $v_2 = \lambda x{:}\tau_1.e_2$, $v_3 = \lambda x{:}\tau_1.e_3$. Let us assume that $(\lambda x{:}\tau_1.e_1, \lambda x{:}\tau_1.e_2) \in \bigcap_k r_k$ and that $(\lambda x{:}\tau_1.e_2, \lambda x{:}\tau_1.e_3) \in \mathcal{V}_k$. We need to show that $(\lambda x{:}\tau_1.e_1, \lambda x{:}\tau_1.e_3) \in \mathcal{V}_k$. Let us pick a $j < k$. Let us pick $(w_1, w_2) \in \mathcal{V}_j$. We need to show that $(e_1\{w_1/x\}, e_3\{w_2/x\}) \in \mathcal{C}_j$. Let us assume that $e_1\{w_1/x\} \longrightarrow^{(l \leq j)} z_1$. It follows that $e_2\{w_2/x\} \longrightarrow^\star z_2$ such that $(z_1, z_2) \in \mathcal{V}_{j-l}$. Moreover, from Lemma 5.1 it must be the case that $e_3\{w_2/x\} \longrightarrow^\star z_3$ and there exists an adequate sequence s_k such that $(z_2, z_3) \in \bigcap_k s_k$. By induction hypothesis then $(z_1, z_3) \in \mathcal{V}_{j-l}$ as required. □

Lemma 5.3. *If $(v_1, v_2) \in \bigcap_k r_k$ where r_k is an adequate sequence then $(v_1, v_2) \in \mathcal{V}_k$*

Proof. Follows by Lemma 5.2 and the fundamental theorem. □

Lemma 5.4. *Assume that $(v_1, v_2) \in \bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$. Then $(v_1, v_2) \in \mathcal{V}_k$.*

Proof. From the assumptions we get that $(v_1, v_2) \in \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$, for every k. Hence, there exists an adequate sequence r_k such that $(v_1, v_2) \in \bigcap_k r_k$. By Lemma 5.3 it follows that $(v_1, v_2) \in \mathcal{V}_k$. □

Lemma 5.5. $\bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r \subseteq \bigcap_k \mathcal{V}_k$

Proof. Follows from Lemma 5.4. □

We hence obtain the following commuting property between unions and intersections.

Theorem 5.6 (Commutation)**.** $\bigcup_{\forall k, r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r = \bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$

Proof. We show the two directions independently:

- $\bigcup_{\forall k, r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r \subseteq \bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$. Pick $(e_1, e_2, \sigma) \in r$ such that $\forall k, r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})$. Then clearly it has to be the case that $r \in \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$ and hence $(e_1, e_2, \sigma) \in \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$. Since k is arbitrary it must also be that $(e_1, e_2, \sigma) \in \bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$.
- $\bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r \subseteq \bigcup_{\forall k, r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$. By Lemma 5.5 we have that

$$\bigcap_k \bigcup_{r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r \subseteq \bigcap_k \mathcal{V}_k \equiv (\precsim) \equiv \bigcup_{\forall k, r^{\mathsf{cxt}} \subseteq F_k(r^{\mathsf{cxt}})} r$$

□

This theorem implies in turn that, in order to show equivalence of two values it suffices to find an adequate infinite sequence of relations so that the two values belong in each of them. In contrast, the original KW method requires the construction of a single relation that is k-adequate for every k. Though the new characterization is interesting, we do not have an example where the new technique is easier to apply than the original.

6 Related and future work

There is a large body of work that employs step-indexed logical relations to deal with recursive types, also in the presence of other language features as, for example, local state [5, 4, 2, 3].

Step-indexed bisimulations, an extension to Sumii and Pierce's bisimulations [20, 19], have been applied to languages with higher-order features and local store [12, 9, 10, 6]. An alternative bisimulation method that builds upon Sumii and Pierce's bisimulations has been proposed by Sangiorgi *et al.* [17], where the explicit use of indices in the definition of the bisimulation is avoided.

The most important direction for future work is to answer the question whether every k-adequate relation is contained within the SI logical relation at index k. This will shed more light in the microscopic connection of the two approximations.

As future work we would like to extend the results in this paper so that we are able to answer the following questions: given a SI logical relation proof of equivalence, is there a straightforward construction of KW bisimulations that witness the same equivalence? And conversely: given a KW bisimulation proof, is there a straightforward translation to a logical relation proof? The fact that the approximations of equivalence seem to be distinct for the two methods make us less optimistic about easy "transliterations" of proofs but nevertheless this seems a natural next step for investigation.

The derived conditions on the "generalized" SI logical relation are sufficient conditions for the fundamental theorem to be true. We believe that they are also necessary and that would be another interesting property to show.

Once the value relation definition of the SI logical relation is fixed ($\mathcal{V}_k$), we saw that the indexing scheme of the computation lifting of the logical relation ($\mathcal{C}_k$) gets fixed. However we are in quest of different definitions of the SI logical relation altogether that will correspond better to the KW bisimulations. One example of such a different definition could use a more traditional presentation of SI logical relations, that is given by interpretation of types at a particular index: $\mathcal{V}[\tau]_k$. The advantage of this approach is that the relation can now be defined by induction on the lexicographic pair $(k, size(\tau))$, which in the case of functions may allow us to check against arguments that are related in the same index as the index of the functions themselves.

Acknowledgments Thanks to Johannes Borgström for his help in proving Theorem 3.6.

References

[1] *Functional Programming and Input/Output*. Cambridge University Press, 1994.

[2] Amal Ahmed and Matthias Blume. Typed closure conversion preserves observational equivalence. In *Proc. 13th ACM SIGPLAN International Conference on Functional Programming (ICFP 2008)*, pages 157–168, New York, NY, USA, 2008. ACM.

[3] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. State-dependent representation independence. In *POPL '09: Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 340–353, New York, NY, USA, 2009. ACM.

[4] Amal J. Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In Peter Sestoft, editor, *ESOP*, volume 3924 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2006.

[5] Andrew W. Appel and David McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.

[6] Nick Benton and Vasileios Koutavas. A mechanized bisimulation for the nu-calculus, November 2007. Draft.

[7] Lars Birkedal and Robert Harper. Relational interpretations of recursive types in an operational setting (summary). In *TACS '97: Proceedings of the Third International Symposium on Theoretical Aspects of Computer Software*, pages 458–490, London, UK, 1997. Springer-Verlag.

[8] Karl Crary and Robert Harper. Syntactic logical relations for polymorphic and recursive types. *Electron. Notes Theor. Comput. Sci.*, 172:259–299, 2007.

[9] V. Koutavas and M. Wand. Bisimulations for untyped imperative objects. In P. Sestoft, editor, *Proc. 15th European Symposium on Programming (ESOP 2006), Programming Languages and Systems*, volume 3924 of *Lecture Notes in Computer Science*, pages 146–161, Berlin, Heidelberg, and New York, 2006. Springer-Verlag.

[10] V. Koutavas and M. Wand. Reasoning about class behavior. Appeared in FOOL/WOOD 2007 Workshop, January 2007.

[11] Vasileios Koutavas. *Reasoning about Imperative and Higher-Order Programs*. PhD thesis, Northeastern University, September 2008.

[12] Vasileios Koutavas and Mitchell Wand. Small bisimulations for reasoning about higher-order imperative programs. In *POPL '06: Proceedings of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*, pages 141–152, New York, NY, USA, 2006. ACM Press.

[13] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[14] A. M. Pitts. A note on logical relations between semantics and syntax. *Logic Journal of the Interest Group in Pure and Applied Logics*, 5(4):589–601, July 1997.

[15] A. M. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. The MIT Press, 2005.

[16] Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in Computer Science*, 10:321–359, 2000.

[17] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *Proc. 22th Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 293–302. IEEE Computer Society, 2007.

[18] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. In *Proc. 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL 2005)*, pages 63–74, New York, NY, USA, 2005. ACM.

[19] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1–3):169–192, May 2007.

[20] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM*, 54(5):1–43, 2007.

[21] Mitchell Wand and Igor Siveroni. Constraint systems for useless variable elimination. In *Proc. 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL 1999)*, pages 291–302, 1999.