A note on the transitivity of step-indexed logical relations

Lars Birkedal and Aleš Bizjak

November 1, 2012

Abstract

We present and discuss a simple semantic approach to force step-indexed logical relations to be transitive.

1 Introduction

Step-indexed logical relations have proved to be useful for reasoning about operational equivalence of programs in programming languages with features that are otherwise hard to model, such as recursive types, general ML-like references or other variations of higher-order store [AAV02, Ahm06, DAB09, DNB10, CD11, HDA10, BRS+11, HDNV12].

However, it has been unclear from the beginning of the development of step-indexed logical relations, whether step-indexed logical relations are actually transitive [Ahm06]. In many cases step-indexed logical relations have primarily been used as a proof-technique for contextual equivalence, and since contextual equivalence is transitive, one has then been able to compose equivalences (even if the step-indexed logical relation itself was possibly not transitive). However, in other applications, e.g., for giving more abstract meaning to low-level languages, we would like to use step-indexed logical relations for *defining* the right notion of equivalence. In this case we cannot rely on a pre-existing notion of contextual equivalence but would really like to ensure that the step-indexed logical relation itself defines a good notion of equality, i.e., that it is transitive.

In this short note, we show how one can force a step-indexed logical relation to be transitive by using a variation of transitive closure proposed by Hoshino in his more abstract account of step-indexed realizability [Hos12].

We end this introduction by remarking that Hur et. al. [HDNV12, HNDV12] also discuss the problems with transitivity for step-indexed logical relations. Indeed, they go on to develop a new notion of relation, called relation transition systems, and prove, via a very technical tour-de-force, that those relations are indeed transitive. Hur et. al. refer to the issue they are addressing as "transitive composability", because they are partly motivated by compiler correctness applications for which one would like to be able to compose logical relations. However, the formal notion Hur et. al. address is nothing but transitivity of the logical relation, as the relation they consider is defined on a single language. In the same vein, the solution we present only addresses transitivity of step-indexed logical relations; it does not address the well-known problem of composability of logical relations. We think that is an orthogonal problem, which might be addressed using ideas from prelogical relations [HS02].

2 Transitive closure of quasi-reflexive symmetric interior

Suppose A and B are nonempty sets, $\varphi : A \to \mathcal{P}(B)$ a function, $\wr : \prod(a : A)\,(\mathcal{P}(\varphi(a) \times \varphi(a)))$ an indexed relation and (C, a_C, b_C, c_C), where $a_C, b_C, c_C \in A$ and $C : \varphi(a_C) \times \varphi(b_C) \to \varphi(c_C)$.

Assume that $\wr$ is a congruence relation for the operation C, meaning, explicitly,

$$\forall a, b \in \varphi(a_C), \forall a', b' \in \varphi(b_C), a \wr_{a_C} b \wedge a' \wr_{b_C} b' \implies C(a, a') \wr_{c_C} C(b, b').$$

Suppose that $\wr$ is not a pointwise transitive relation and let $\equiv$ be its pointwise transitive closure. There is no reason to believe that $\equiv$ is also a congruence for C. Indeed, there are simple counterexamples, even when $\wr$ is symmetric (but not when it is reflexive!).

If we try to examine where the obvious attempt gets stuck we see that the problem is that we can have essentially different lengths of chains. To be specific, we have

$$a \wr_{ac} a_1 \wr_{ac} a_2 \wr_{ac} \cdots \wr_{ac} a_n = b$$

$$a' \wr_{bc} a'_1 \wr_{bc} a'_2 \wr_{bc} \cdots \wr_{bc} a'_{n+k} = b'$$

so we can prove $C(a, a') \wr_{cc} C(a_1, a'_1) \wr_{cc} \cdots \wr_{cc} C(a_n, a'_n)$ but now we cannot continue, so we cannot conclude $C(a, a') \equiv_{cc} C(b, b')$. If we knew that b was related to itself in the original relation, we could extend the first chain with k b's and so we could conclude $C(a, a') \equiv C(b, b')$. Quasi reflexive relations in [Hos12] accomplish exactly this.

Definition 1. A relation $\bigcirc$ on a set X is quasi-reflexive when

$$\forall a, b \in X, a \bigcirc b \implies a \bigcirc a \wedge b \bigcirc b$$

or, in prose, a relation is quasi reflexive, when if an element is related to any other element, it must be related to itself.

Definition 2. For a relation $\bigcirc$ on X its quasi-reflexive *interior*, denoted by $\bigcirc^\bullet$, is defined to be its largest quasi-reflexive *subrelation*. It can be specified explicitly as

$$\bigcirc^\bullet = \left\{(a, b) \;\middle|\; a \bigcirc a \wedge a \bigcirc b \wedge b \bigcirc b\right\}.$$

Definition 3. For a relation $\bigcirc$ on X its quasi-reflexive *symmetric interior*, denoted by $\bigcirc^\diamond$, is defined to be its largest quasi-reflexive *symmetric subrelation*. It can be specified explicitly as

$$\bigcirc^\diamond = \left\{(a, b) \;\middle|\; a \bigcirc a \wedge a \bigcirc b \wedge b \bigcirc a \wedge b \bigcirc b\right\}.$$

Proposition 4. *Let $\wr^\bullet$ and $\wr^\diamond$ be the pointwise quasi-reflexive and quasi-reflexive symmetric interiors of $\wr$, respectively. We list some properties of these constructions which are easily proved.*

- $\wr^\bullet$ *is a congruence for operation C*
- $\wr^\diamond$ *is a congruence for operation C*
- $(\wr^\diamond)^*$ *is a congruence*[1]
- *If $\wr$ is pointwise symmetric and transitive (a pointwise per), then $\wr = \wr^\bullet = \wr^\diamond = (\wr^\diamond)^*$.*
- $\forall a \in A, \forall b \in \varphi(a), b \wr_a b \iff b \, (\wr^\diamond_a)^* \, b$

3 A Transitive Step-Indexed Logical Relation

Let us now see how the results in the previous section can be used to obtain a transitive step-indexed logical relation.

Suppose that we have a step-indexed logical approximation relation $\Theta \mid \Gamma \vdash c \prec c' : A$. In this section, we assume the reader is familiar with such definitions; for readers who are not, we recall a specific definition in appendix, for a higher-order polymorphic call-by-value language with recursive types.

Let us be more precise about what kind of relation this is. Let $\mathcal{S}$ be the set of type variable contexts, for $\Theta \in \mathcal{S}$ let $\mathcal{D}_\Theta$ be the set of well formed contexts in type variable context Θ and $\mathcal{T}_\Theta$ the set of well formed types in type variable context Θ. For a variable context Γ let $\mathbb{E}_\Gamma$ denote the set of terms with at most variables in Γ free.

Then $\cdot \mid \cdot \vdash \cdot \prec \cdot : \cdot$ can be seen as a dependent function of type

$$\prod (\Theta \in \mathcal{S}) \, (\Gamma \in \mathcal{D}_\Theta) \, (A \in \mathcal{T}_\Theta) \, . \mathcal{P} \left(\mathbb{E}_\Gamma, \mathbb{E}_\Gamma\right)$$

[1]$*$ denotes the pointwise transitive closure

or for each $\Theta \in \mathcal{S}$, $\Gamma \in \mathcal{D}_\Theta$ and $A \in \mathcal{S}_\Theta$, $\Theta \mid \Gamma \vdash \cdot \prec \cdot : A$ is a binary relation on $\mathbb{E}_\Gamma$.

For this relation we prove compatibility lemmas, the fundamental property of logical relations, and that it is sound with respect to contextual approximation.

We then define the relation $\Theta \mid \Gamma \vdash c \sim c' : A$ to be the pointwise quasi-reflexive symmetric interior of the standard step-indexed logical relation and then we define $\Theta \mid \Gamma \vdash c \approx c' : A$ to be the pointwise transitive closure of this quasi reflexive symmetric interior.

Proposition 4 then tells us that the last relation has the following properties

- It is a congruence relation.
- The compatibility lemmas hold.
- $\Theta \mid \Gamma \vdash c \prec c : A \iff \Theta \mid \Gamma \vdash c \approx c : A$

The last item above means that our resulting logical relation satisfies the fundamental property of logical relations. It is also sound with respect to contextual equivalence, because contextual equivalence is a transitive relation.

Note that for all of these points, it doesn't matter how the original relation was defined, but just that it satisfied certain properties.

Let us see one case in the proof that this transitive closure is indeed a congruence relation, the case of application. Suppose $\Theta \mid \Gamma \vdash c \approx d : A \to B$ and $\Theta \mid \Gamma \vdash c' \approx d' : A$. We then wish to conclude that $\Theta \mid \Gamma \vdash c\ c' \approx d\ d' : B$. By assumption we have a sequence of terms $c_1, \ldots, c_n$, such that

$$\Theta \mid \Gamma \vdash c \sim c_1 : A \to B, \quad \Theta \mid \Gamma \vdash c_n \sim d : A \to B, \quad \forall i \in \{1, 2, \ldots, n-1\}, \Theta \mid \Gamma \vdash c_i \sim c_{i+1} : A \to B$$

and a sequence of terms $c'_1, c'_2, \ldots, c'_m$ such that

$$\Theta \mid \Gamma \vdash c' \sim c'_1 : A, \quad \Theta \mid \Gamma \vdash c'_m \sim d' : A, \quad \forall i \in \{1, 2, \ldots, n-1\}, \Theta \mid \Gamma \vdash c'_i \sim c'_{i+1} : A$$

Assume further that $m \geq n$, the other case being completely symmetric. Define $c_{n+1} = c_{n+2} = \cdots = c_{m+1} = d$. We then have

$$\Theta \mid \Gamma \vdash c \sim c_1 : A \to B$$

and

$$\forall i \in \{1, 2, \ldots, m\}, \Theta \mid \Gamma \vdash c_i \sim c_{i+1} : A \to B$$

crucially due to the fact that $\sim$ is quasi reflexive. Using the fact that $\sim$ is a congruence we get the following sequence of equalities

$$\begin{gathered}\Theta \mid \Gamma \vdash c\ c' \sim c_1\ c'_1 : B \\ \Theta \mid \Gamma \vdash c_1\ c'_1 \sim c_2\ c'_2 : B \\ \vdots \\ \Theta \mid \Gamma \vdash c_m\ c'_m \sim c_{m+1}\ d' : B\end{gathered}$$

and as $c_{m+1} = d$, we have $\Theta \mid \Gamma \vdash c\ c' \approx d\ d' : B$.

Further, if the original relation was complete with respect to contextual equivalence then in particular it was pointwise symmetric and transitive. Then Proposition 4 tells us that we get back the same relation, so it is still complete.

4 Discussion

In [Ahm06], the author notices that the immediate proof of transitivity of the defined logical relation does not work, and the problem is *precisely* that terms that are related to other terms are not necessarily related to themselves. The solution Ahmed proposes is to construct the relation with more typing information built in so that the fundamental property of logical relations can be used to get transitivity. This change is disruptive and requires essentially reproving all of the auxiliary lemmas. Moreover, this approach does not work in general when constructing relations over low-level machine languages. If we just need a reflexive logical relation that is well behaved, using the quasi-reflexive symmetric interior and then the transitive closure seems to be a clean approach. We preserve the essential characteristics of logical relations, but we also get a proper equality relation on open terms.

Moreover, the approach described in this note can be seen as the semantic analogue to a more syntactic solution described in [Ahm06]. The solution in *loc. cit.* exploited the fact that relations could be defined over well-typed terms to get a quasi-reflexive and transitive relation, whereas we propose to take any relation and make it transitive in a way that preserves its quintessential properties. We also note that this approach works even in the case of low-level languages, where we cannot run to the safety of typed realizers.

A "Standard" step-indexed logical relation

For completeness, we include here a definition of a reasonably standard step-indexed logical relation. For detailed studies see [DNB10, Ahm06, DAB09]. The language under consideration is a an extension of simply typed lambda calculus with recursive types, polymorphism and call by value evaluation order. We employ the Curry-style semantics where terms are not intrinsically typed. Let $\mathbb{V}$ denote the set of closed values, $\mathbb{E}$ the set of closed terms, $\mathcal{E}$ the set of evaluation contexts and $\mathbb{T}$ the (countably infinite) set of type variables.

Let Ξ be the set of decreasing sequences of binary relations on values, explicitly

$$\Xi = \left\{ X : \mathbb{N} \to \mathcal{P}(\mathbb{V} \times \mathbb{V}) \;\middle|\; \forall n \in \mathbb{N}, X_n \supseteq X_{n+1} \right\}.$$

For $\Theta \subset_{\mathrm{fin}} \mathbb{T}$ we define

$$\mathbf{Map}(\Theta) = \Theta \to \Xi.$$

Next, we define three relations, a value relation, an evaluation relation and a computation relation, the last one being here just for convenience. These are binary relations on closed values, evaluation contexts and closed terms, respectively, and are defined by recursion on the type and the step index.

Specifically, for each type A in type variable context Θ we have

$$\begin{aligned} \llbracket \Theta \mid A \rrbracket^{\cdot}_{\cdot} &: \mathbb{N} \to \mathbf{Map}(\Theta) \to \mathcal{P}(\mathbb{V} \times \mathbb{V}) \\ \mathcal{E}\llbracket \Theta \mid A \rrbracket^{\cdot}_{\cdot} &: \mathbb{N} \to \mathbf{Map}(\Theta) \to \mathcal{P}(\mathcal{E} \times \mathcal{E}) \\ \mathcal{C}\llbracket \Theta \mid A \rrbracket^{\cdot}_{\cdot} &: \mathbb{N} \to \mathbf{Map}(\Theta) \to \mathcal{P}(\mathbb{E} \times \mathbb{E}) \end{aligned}$$

and for a given $\varphi \in \mathbf{Map}(\Theta)$ and $n \in \mathbb{N}$ the application is then written as $\llbracket \Theta \mid A \rrbracket^{\varphi}_{n} \subseteq \mathbb{V} \times \mathbb{V}$.

The following definitions are essentially the same as (for a subset of) the language in [DNB10], but where the worlds are just natural numbers.

$$\begin{aligned} \llbracket \Theta \mid \alpha \rrbracket^{\varphi}_{n} &= \varphi(\alpha)_n \\ \llbracket \Theta \mid \mathbf{1} \rrbracket^{\varphi}_{n} &= \{(\langle\rangle, \langle\rangle)\} \\ \llbracket \Theta \mid A \times B \rrbracket^{\varphi}_{n} &= \left\{ (\langle a, b\rangle, \langle a', b'\rangle) \;\middle|\; (a, a') \in \llbracket \Theta \mid A \rrbracket^{\varphi}_{n} \wedge (b, b') \in \llbracket \Theta \mid B \rrbracket^{\varphi}_{n} \right\} \\ \llbracket \Theta \mid A \to B \rrbracket^{\varphi}_{n} &= \left\{ \begin{array}{l|r} (\boldsymbol{\lambda} x.b, \boldsymbol{\lambda} x.b') & \forall i \leq n, \forall (a, a') \in \llbracket \Theta \mid A \rrbracket^{\varphi}_{i}, \forall (e, e') \in \mathcal{E}\llbracket \Theta \mid B \rrbracket^{\varphi}_{i}, \\ & e[b[a/x]] \Downarrow^{\leq i} \implies e'[b'[a'/x]] \Downarrow \end{array} \right\} \\ \llbracket \Theta \mid \forall \alpha.A \rrbracket^{\varphi}_{n} &= \left\{ \begin{array}{l|r} (\boldsymbol{\lambda} \alpha.b, \boldsymbol{\lambda} \alpha.b') & \forall i \leq n, \forall R \in \Xi, \forall (e, e') \in \mathcal{E}\llbracket \Theta, \alpha \mid A \rrbracket^{\varphi[\alpha \mapsto R]}_{i}, \\ & e[b] \Downarrow^{\leq i} \implies e'[b'] \Downarrow \end{array} \right\} \\ \llbracket \Theta \mid \mu\alpha.A \rrbracket^{\varphi}_{n} &= \left\{ (\mathbf{fold}\ a, \mathbf{fold}\ a') \;\middle|\; \forall i < n, (a, a') \in \llbracket \Theta \mid A[\mu\alpha.A/\alpha] \rrbracket^{\varphi}_{i} \right\} \\ \mathcal{E}\llbracket \Theta \mid A \rrbracket^{\varphi}_{n} &= \left\{ (e, e') \;\middle|\; \forall i \leq n, \forall (a, a') \in \llbracket \Theta \mid A \rrbracket^{\varphi}_{i}, e[a] \Downarrow^{\leq i} \implies e'[a'] \Downarrow \right\} \\ \mathcal{C}\llbracket \Theta \mid A \rrbracket^{\varphi}_{n} &= \left\{ (c, c') \;\middle|\; \forall (e, e') \in \mathcal{E}\llbracket \Theta \mid A \rrbracket^{\varphi}_{n}, e[c] \Downarrow^{\leq n} \implies e'[c'] \Downarrow \right\} \end{aligned}$$

To extend the relations to open terms we introduce an auxiliary relation $\mathcal{G}$ as follows (γ_0 denotes the empty substitution)

$$\mathcal{G}\left[\!\left[\Theta \mid \emptyset\right]\!\right]_n^{\varphi} = \{(\gamma_0, \gamma_0)\}$$
$$\mathcal{G}\left[\!\left[\Theta \mid \Gamma, x : A\right]\!\right]_n^{\varphi} = \left\{(\gamma\left[x \mapsto v\right], \gamma'\left[x \mapsto v'\right]) \;\middle|\; (v, v') \in \left[\!\left[\Theta \mid A\right]\!\right]_n^{\varphi} \wedge (\gamma, \gamma') \in \mathcal{G}\left[\!\left[\Theta \mid \Gamma\right]\!\right]_n^{\varphi}\right\}$$

We can then define the relations we are interested in as

$$\Theta \mid \Gamma \vdash c \prec c' : A \Leftrightarrow \forall n \in \mathbb{N}, \forall \varphi \in \mathbf{Map}\left(\Theta\right), \forall (\gamma, \gamma') \in \mathcal{G}\left[\!\left[\Theta \mid \Gamma\right]\!\right]_n^{\varphi}, (c\gamma, c'\gamma') \in \mathcal{C}\left[\!\left[\Theta \mid A\right]\!\right]_n^{\varphi}$$
$$\Theta \mid \Gamma \vdash c \overset{\circ}{\sim} c' : A \Leftrightarrow (\Theta \mid \Gamma \vdash c \prec c' : A) \wedge (\Theta \mid \Gamma \vdash c' \prec c : A)$$
$$\Theta \mid \Gamma \vdash c \sim c' : A \Leftrightarrow (\Theta \mid \Gamma \vdash c \prec c' : A) \wedge (\Theta \mid \Gamma \vdash c' \prec c : A) \wedge (\Theta \mid \Gamma \vdash c \prec c : A) \wedge (\Theta \mid \Gamma \vdash c' \prec c' : A)$$

The first relation is the standard approximation relation and the subsequent two are its subrelations. The approximation relation is a congruence relation and is sound with respect to contextual approximation. This is shown with the standard tedious argument, but the specifics are not important for our current purposes.

References

[AAV02] Amal J. Ahmed, Andrew W. Appel, and Roberto Virga. A stratified semantics of general references embeddable in higher-order logic. In *Proceedings of 17th Annual IEEE Symposium Logic in Computer Science*, pages 75–86. IEEE Computer Society Press, 2002.

[Ahm06] Amal Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In *of Lecture Notes in Computer Science*, pages 69–83. Springer, 2006.

[BRS$^+$11] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. In *Proceedings of POPL*, 2011.

[CD11] Chung Kil-Hur and Derek Dreyer. A kripke logical relation between ml and assembly. In *POPL 2011*, 2011.

[DAB09] D. Dreyer, A. Ahmed, and L. Birkedal. Logical step-indexed logical relations. In *Logic In Computer Science, 2009. LICS '09. 24th Annual IEEE Symposium on*, pages 71 –80, aug. 2009.

[DNB10] Derek Dreyer, Georg Neis, and Lars Birkedal. The impact of higher-order state and control effects on local relational reasoning. *SIGPLAN Not.*, 45(9):143–156, September 2010.

[HDA10] Aquinas Hobor, Robert Dockins, and Andrew Appel. A theory of indirection via approximation. In *POPL*, 2010.

[HDNV12] Chung-Kil Hur, Derek Dreyer, Georg Neis, and Viktor Vafeiadis. The marriage of bisimulations and kripke logical relations. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '12, pages 59–72, New York, NY, USA, 2012. ACM.

[HNDV12] Chung-Kil Hur, Georg Neis, Derek Dreyer, and Viktor Vafeiadis. The transitive composability of relation transition systems. Technical Report MPI-SWS-2012-002, MPI-SWS, 2012.

[Hos12] Naohiko Hoshino. Step indexed realizability semantics for a call-by-value language based on basic combinatorial objects. In *Proceedings of LICS 2012*, 2012.

[HS02] Furio Honsell and Donald Sannella. Prelogical relations. *Inf. Comput.*, 178(1):23–43, October 2002.