Logical Step-Indexed Logical Relations

Derek Dreyer

Max Planck Institute for Software Systems
Saarbrücken, Germany

LICS 2009
UCLA
August 12, 2009

Joint work with Amal Ahmed and Lars Birkedal
Logical Relations

\[ \mathcal{V} \left[ \text{nat} \right] \rho = \left\{ (n, n) \mid n \in \mathbb{N} \right\} \]

\[ \mathcal{V} \left[ \tau' \rightarrow \tau'' \right] \rho = \left\{ (\lambda x. e_1, \lambda x. e_2) \mid \forall v_1, v_2. \\
\quad (v_1, v_2) \in \mathcal{V} \left[ \tau' \right] \rho \implies \\
\quad (e_1[v_1/x], e_2[v_2/x]) \in \mathcal{E} \left[ \tau'' \right] \rho \right\} \]

\[ \mathcal{V} \left[ \exists \alpha. \tau \right] \rho = \left\{ (\text{pack } \tau_1, v_1 \text{ as } \cdots, \text{pack } \tau_2, v_2 \text{ as } \cdots) \mid \exists \chi \in \text{Rel}(\tau_1, \tau_2). \\
\quad (v_1, v_2) \in \mathcal{V} \left[ \tau \right] \rho, \alpha \mapsto (\tau_1, \tau_2, \chi) \right\} \]

\[ \mathcal{V} \left[ \alpha \right] \rho = \chi \quad \text{where } \rho(\alpha) = (\tau_1, \tau_2, \chi) \]
Logical Relations for Recursive Types?

\[ \forall \left[ \mu \alpha. \tau \right] \rho = \left\{ (\text{fold} \, v_1, \text{fold} \, v_2) \mid (v_1, v_2) \in \forall \left[ \tau[\mu \alpha. \tau / \alpha] \right] \rho \right\} \]
Logical Relations for Recursive Types?

\[ \nu [\mu \alpha. \tau] \rho = \{ (\text{fold } v_1, \text{fold } v_2) \mid (v_1, v_2) \in \nu [\tau[\mu \alpha. \tau/\alpha]] \rho \} \]

**Problem:** The definition is no longer well-founded!
**Idea:** Index logical relations by $n \in \mathbb{N}$ representing “the number of steps left until the clock runs out.”

- Two terms are related “infinitely” iff they are $n$-related (for all $n$).

$$\mathcal{V} [\mu \alpha. \tau] \rho = \{(n, \text{fold } v_1, \text{fold } v_2) \mid (n - 1, v_1, v_2) \in \mathcal{V} [\tau [\mu \alpha. \tau / \alpha]] \rho\}$$

Intuitively, this makes sense because it takes a step of computation to extract $v_i$ from $\text{fold } v_i$. 
Advantages of Step-Indexed Logical Relations

Easy to develop using only elementary mathematical constructions.

Applicable to “difficult” languages, e.g., with higher-order state:

- Imperative self-adjusting computation (Acar et al., POPL’08)
- Representation independence for “generative” ADTs (POPL’09)
- Parametricity in the presence of dynamic typing (ICFP’09)
- Compiler correctness (Benton et al., e.g., TLDI’09, ICFP’09)
- …
Comparison With Other Approaches

With more mathematically sophisticated approaches (e.g., minimal invariance, FM-cpos, ultra-metric spaces):

- ✗ Hard to construct, not as (obviously) widely applicable

With step-indexed logical relations:

- ✔ Easy to construct, widely applicable
Comparison With Other Approaches

With more mathematically sophisticated approaches (e.g., minimal invariance, FM-cpos, ultra-metric spaces):

× Hard to construct, not as (obviously) widely applicable
✓ Easy to develop high-level equational proof principles

With step-indexed logical relations:

✓ Easy to construct, widely applicable
× Hard to develop high-level equational proof principles

You get what you pay for!
Steps make constructing the model easy, but the *user* of the model shouldn’t have to deal with them.

- Important to develop clean, abstract, **step-free** proof principles
Steps make constructing the model easy, but the user of the model shouldn’t have to deal with them.

- Important to develop clean, abstract, step-free proof principles

E.g. Appel-McAllester claim this extensionality property:

- $f_1$ and $f_2$ are infinitely related (e.g., related for any # of steps) iff for all $v_1$ and $v_2$ that are infinitely related, $f_1 v_1$ and $f_2 v_2$ are, too.
Steps make constructing the model easy, but the *user* of the model shouldn’t have to deal with them.

- Important to develop clean, abstract, step-free proof principles

E.g. Appel-McAllester claim this *extensionality property*:

- $f_1$ and $f_2$ are infinitely related (e.g., related for any # of steps) *iff* for all $v_1$ and $v_2$ that are infinitely related, $f_1 v_1$ and $f_2 v_2$ are, too.

Unfortunately, it is *false*!

- In fact, $f_1$ and $f_2$ are infinitely related *iff*, for any step level $n$, for all $v_1$ and $v_2$ that are *$n$-related*, $f_1 v_1$ and $f_2 v_2$ are, too.
Step-indexed logical relations are fundamentally *asymmetric*, i.e., they model approximation (\(\leq\)), not equivalence (\(\equiv\)).

- We can define \(e_1 \equiv e_2\) to mean \(e_1 \leq e_2 \land e_2 \leq e_1\).
Step-indexed logical relations are fundamentally \textit{asymmetric}, \textit{i.e.}, they model approximation ($\leq$), not equivalence ($\equiv$).

- We can define $e_1 \equiv e_2$ to mean $e_1 \leq e_2 \land e_2 \leq e_1$.

We would like a \textit{symmetric} extensionality principle, \textit{e.g.},

- $f_1 \equiv f_2$ iff $\forall v_1, v_2$. we have that $v_1 \equiv v_2$ implies $f_1 v_1 \equiv f_2 v_2$. 
Problem #2: Lack of Equational Proof Principles

Step-indexed logical relations are fundamentally *asymmetric*, i.e., they model approximation ($\leq$), not equivalence ($\equiv$).

- We can define $e_1 \equiv e_2$ to mean $e_1 \leq e_2 \land e_2 \leq e_1$.

We would like a *symmetric* extensionality principle, e.g.,

- $f_1 \equiv f_2$ iff $\forall v_1, v_2$. we have that $v_1 \equiv v_2$ implies $f_1 v_1 \equiv f_2 v_2$.

But even ignoring Problem #1, this is false:

- To show $f_1 \equiv f_2$, we must show that $v_1 \leq v_2$ implies $f_1 v_1 \leq f_2 v_2$, and that $v_2 \leq v_1$ implies $f_2 v_2 \leq f_1 v_1$. 
Our Contributions

Define a relational modal logic, LSLR, for expressing step-indexed logical relations without mentioning steps.

Define a step-free logical relation in LSLR for reasoning about program (in-)equivalence in System F + recursive types.

Show logical relation is sound w.r.t. contextual equivalence by defining a suitable “step-indexed” model of LSLR.

Develop a set of useful derivable rules concerning the logical relation.

Demonstrate the effectiveness of our approach by proving several representative examples of contextual equivalences from the literature.
Outline

1. The Language $F^\mu$

2. The Logic LSLR

3. Encoding a Logical Relation for $F^\mu$ in LSLR

4. Derivable Rules
Outline

1 The Language $F^\mu$

2 The Logic LSLR

3 Encoding a Logical Relation for $F^\mu$ in LSLR

4 Derivable Rules
The Language $F^\mu$

**Types**  $\tau ::= \alpha \mid \text{unit} \mid \text{int} \mid \text{bool} \mid \tau_1 \times \tau_2 \mid \tau_1 + \tau_2 \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha. \tau \mid \exists \alpha. \tau \mid \mu \alpha. \tau$

**Prim Ops**  $o ::= \ + \mid \ - \mid \ = \mid \ < \mid \ \leq \mid \ldots$

**Terms**  $e ::= x \mid () \mid \pm n \mid o(e_1,\ldots,e_n) \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid \langle e_1, e_2 \rangle \mid \text{fst } e \mid \text{snd } e \mid \text{inl}_\tau e \mid \text{inr}_\tau e \mid \text{case } e \text{ of } \text{inl } x_1 \Rightarrow e_1 \mid \text{inr } x_2 \Rightarrow e_2 \mid \lambda x : \tau. e \mid e_1 e_2 \mid \Lambda \alpha. e \mid e[\tau] \mid \text{pack } \tau, e \text{ as } \exists \alpha. \tau' \mid \text{unpack } e_1 \text{ as } \alpha, x \in e_2 \mid \text{fold}_\tau e \mid \text{unfold } e$

**Values**  $v ::= x \mid () \mid \pm n \mid \text{true} \mid \text{false} \mid \langle v_1, v_2 \rangle \mid \text{inl}_\tau v \mid \text{inr}_\tau v \mid \lambda x : \tau. e \mid \Lambda \alpha. e \mid \text{pack } \tau_1, v \text{ as } \exists \alpha. \tau \mid \text{fold}_\tau v$
Outline

1. The Language $F^\mu$
2. The Logic LSLR
3. Encoding a Logical Relation for $F^\mu$ in LSLR
4. Derivable Rules
The Logic LSLR (Basic Idea)

Start with Plotkin and Abadi’s “logic for parametric polymorphism” (TLCA’93)

- Adapt it to reason operationally about CBV small-step semantics

Extend it with recursively defined relations

- Enables straightforward logical relation for recursive types
- To make sense of circularity, introduce “later” operator $\triangleright A$ from Appel, Melliès, Richards, and Vouillon’s “very modal model” paper (POPL’07), which in turn was adapted from Gödel-Löb logic of provability
The Logic LSLR (Syntax)

Rel. Var’s \( r, s \in \text{RelVar} \)

\( \text{F}^\mu \) Ctxt’s \( \Gamma ::= \cdot | \Gamma, \alpha | \Gamma, x : \tau | \Gamma, t : \tau \)

Rel. Ctxt’s \( \Delta ::= \cdot | \Delta, r : \text{VRel}(\tau_1, \tau_2) | r : \text{TRel}(\tau_1, \tau_2) \)

Log. Ctxt’s \( \Theta ::= \cdot | \Theta, A \)

Atomic Prop’s \( P ::= e_1 = e_2 | e_1 \mapsto e_2 | e_1 \mapsto^0 e_2 | e_1 \mapsto^1 e_2 \)

Propositions \( A, B ::= P | \top | \bot | A \land B | A \lor B | A \supset B | \forall \Gamma. A | \exists \Gamma. A | \forall \Delta. A | \exists \Delta. A | (e_1, e_2) \in R | \triangleright A \)

Relations \( R, S ::= r | (x_1 : \tau_1, x_2 : \tau_2).A | (t_1 : \tau_1, t_2 : \tau_2).A | \mu r. R \)
Γ; Δ; Θ ⊢ A
(v₁, v₂) ∈ (x₁ : τ₁, x₂ : τ₂).A ≡ A[v₁/x₁, v₂/x₂]

(e₁, e₂) ∈ (t₁ : τ₁, t₂ : τ₂).A ≡ A[e₁/t₁, e₂/t₂]

(e₁, e₂) ∈ µr.R ≡ (e₁, e₂) ∈ R[µr.R/r]
Monotonicity

\[ A \subseteq \triangleright A \]
Löb Rule

\((\Box A \supset A) \supset A\)
Distributivity Laws

\[ \triangleright (A \land B) \equiv \triangleright A \land \triangleright B \]
\[ \triangleright (A \lor B) \equiv \triangleright A \lor \triangleright B \]
\[ \triangleright (A \supset B) \equiv \triangleright A \supset \triangleright B \]
\[ \forall \Gamma . A \equiv \forall \Gamma . \triangleright A \]
\[ \forall \Delta . A \equiv \forall \Delta . \triangleright A \]
\[ \exists \Gamma . A \equiv \exists \Gamma . \triangleright A \]
\[ \exists \Delta . A \equiv \exists \Delta . \triangleright A \]
Outline

1. The Language \( F^\mu \)
2. The Logic LSLR
3. Encoding a Logical Relation for \( F^\mu \) in LSLR
4. Derivable Rules
Logical Relation for Values

\[ \mathcal{V}[\alpha] \rho \overset{\text{def}}{=} R, \text{ where } \rho(\alpha) = (\tau_1, \tau_2, R) \]

\[ \mathcal{V}[\tau_b] \rho \overset{\text{def}}{=} (x_1 : \tau_b, x_2 : \tau_b). x_1 = x_2, \text{ where } \tau_b \in \{ \text{unit, int, bool} \} \]

\[ \mathcal{V}[\tau' \times \tau''] \rho \overset{\text{def}}{=} (x_1 : \rho_1(\tau' \times \tau''), x_2 : \rho_2(\tau' \times \tau'')). \]
\[ \exists x'_1, x''_1, x'_2, x''_2. \quad x_1 = \langle x'_1, x''_1 \rangle \land x_2 = \langle x'_2, x''_2 \rangle \land \]
\[ (x'_1, x'_2) \in \mathcal{V}[\tau'] \rho \land (x''_1, x''_2) \in \mathcal{V}[\tau''] \rho \]

\[ \mathcal{V}[\tau' + \tau''] \rho \overset{\text{def}}{=} (x_1 : \rho_1(\tau' + \tau''), x_2 : \rho_2(\tau' + \tau'')). \]
\[ (\exists x'_1, x'_2. \quad x_1 = \text{inl } x'_1 \land x_2 = \text{inl } x'_2 \land (x'_1, x'_2) \in \mathcal{V}[\tau'] \rho) \]
\[ \lor (\exists x''_1, x''_2. \quad x_1 = \text{inr } x''_1 \land x_2 = \text{inr } x''_2 \land (x''_1, x''_2) \in \mathcal{V}[\tau''] \rho) \]

\[ \mathcal{V}[\tau' \rightarrow \tau''] \rho \overset{\text{def}}{=} (x_1 : \rho_1(\tau' \rightarrow \tau''), x_2 : \rho_2(\tau' \rightarrow \tau'')). \]
\[ \forall y_1, y_2. \quad (y_1, y_2) \in \mathcal{V}[\tau'] \rho \supset (x_1 y_1, x_2 y_2) \in \mathcal{E}[\tau''] \rho \]
\[ \forall \alpha.\, \tau \] \rho \overset{\text{def}}{=} (x_1 : \rho_1(\forall \alpha.\, \tau), x_2 : \rho_2(\forall \alpha.\, \tau)). \\
\forall \alpha_1, \alpha_2. \forall r : \text{VRel}(\alpha_1, \alpha_2). \\
(x_1 [\alpha_1], x_2 [\alpha_2]) \in E[\tau] \rho, \alpha \mapsto (\alpha_1, \alpha_2, r)

\[ \exists \alpha.\, \tau \] \rho \overset{\text{def}}{=} (x_1 : \rho_1(\exists \alpha.\, \tau), x_2 : \rho_2(\exists \alpha.\, \tau)). \\
\exists \alpha_1, \alpha_2, y_1, y_2. \exists r : \text{VRel}(\alpha_1, \alpha_2). \\
x_1 = \text{pack} \alpha_1, y_1 \text{ as } \cdots \land x_2 = \text{pack} \alpha_2, y_2 \text{ as } \cdots \land \\
(y_1, y_2) \in V[\tau] \rho, \alpha \mapsto (\alpha_1, \alpha_2, r)
\[ \mathcal{V} [\mu \alpha . \tau] \rho \overset{\text{def}}{=} \mu r . (x_1 : \rho_1(\mu \alpha . \tau), x_2 : \rho_2(\mu \alpha . \tau)). \]
\[ \exists y_1, y_2. \ x_1 = \text{fold} y_1 \land x_2 = \text{fold} y_2 \land \]
\[ \triangledown (y_1, y_2) \in \mathcal{V} [\tau] \rho, \alpha \mapsto (\rho_1(\mu \alpha . \tau), \rho_2(\mu \alpha . \tau), r) \]
1. The Language $F^\mu$

2. The Logic LSLR

3. Encoding a Logical Relation for $F^\mu$ in LSLR

4. Derivable Rules
Coincidence of Value and Term Relations

\[
\Gamma; \Delta; \Theta \vdash (v_1, v_2) \in V \llbracket \tau \rrbracket \rho
\]
\[
\Gamma; \Delta; \Theta \vdash (v_1, v_2) \in E \llbracket \tau \rrbracket \rho
\]
Extensionality

\[\Gamma, x_1, x_2; \Delta; \Theta, (x_1, x_2) \in \mathcal{V}[\tau'] \rho \vdash (v_1 x_1, v_2 x_2) \in \mathcal{E}[\tau''] \rho\]

\[\Gamma; \Delta; \Theta \vdash (v_1, v_2) \in \mathcal{V}[\tau' \rightarrow \tau''] \rho\]
Evaluation Rules

\[ \Gamma; \Delta; \Theta \vdash e_1 \mapsto^* e'_1 \quad \Gamma; \Delta; \Theta \vdash e_2 \mapsto^* e'_2 \]

\[ \Gamma; \Delta; \Theta \vdash (e'_1, e'_2) \in \mathcal{E} \llbracket \tau \rrbracket \rho \]

\[ \Gamma; \Delta; \Theta \vdash (e_1, e_2) \in \mathcal{E} \llbracket \tau \rrbracket \rho \]

\[ \Gamma; \Delta; \Theta \vdash (e_1, e_2) \in \mathcal{E} \llbracket \tau \rrbracket \rho \]

\[ \Gamma, x_1, x_2; \Delta; \Theta, e_1 \mapsto^* x_1, e_2 \mapsto^* x_2, (x_1, x_2) \in \mathcal{V} \llbracket \tau \rrbracket \rho \]

\[ \vdash (E_1[x_1], E_2[x_2]) \in \mathcal{E} \llbracket \tau' \rrbracket \rho' \]

\[ \Gamma; \Delta; \Theta \vdash (E_1[e_1], E_2[e_2]) \in \mathcal{E} \llbracket \tau' \rrbracket \rho' \]
Useful Rules Concerning the ▷ Modality

\[ \Gamma; \Delta; \Theta_1, \Theta_2 \vdash B \]
\[ \frac{\Gamma; \Delta; \Theta_1, \Theta_2 \vdash B}{\Gamma; \Delta; \Theta_1, \Theta_2 \vdash \Box B} \]

\[ \Gamma; \Delta; \Theta \vdash e_1 \overset{1}{\mapsto} e_1' \quad \Gamma; \Delta; \Theta \vdash e_2 \overset{1}{\mapsto} e_2' \]
\[ \frac{\Gamma; \Delta; \Theta \vdash \Box (e_1', e_2') \in \mathcal{E}[\tau]\rho}{\Gamma; \Delta; \Theta \vdash (e_1, e_2) \in \mathcal{E}[\tau]\rho} \]

\[ \Gamma; \Delta; \Theta, \Box A \vdash A \]
\[ \frac{\Gamma; \Delta; \Theta \vdash A}{\Gamma; \Delta; \Theta \vdash A} \]
Useful Rules Concerning the $\triangleright$ Modality

\[
\Gamma; \Delta; \Theta_1, \Theta_2 \vdash B \\
\Gamma; \Delta; \Theta_1, \triangleright \Theta_2 \vdash \triangleright B
\]

\[
\Gamma; \Delta; \Theta \vdash e_1 \mapsto e'_1 \quad \Gamma; \Delta; \Theta \vdash e_2 \mapsto e'_2 \\
\Gamma; \Delta; \Theta \vdash \triangleright (e'_1, e'_2) \in \mathcal{E} \left[ \tau \right] \rho \\
\Gamma; \Delta; \Theta \vdash (e_1, e_2) \in \mathcal{E} \left[ \tau \right] \rho
\]

\[
\Gamma; \Delta; \Theta, \triangleright A \vdash A \\
\Gamma; \Delta; \Theta \vdash A
\]
Fixed-Point Induction

\[ F_i = \text{fun } f(x_i) \text{ is } e_i \]

\[
\Gamma, x_1, x_2; \Delta; \Theta, (F_1, F_2) \in \mathcal{V}[\tau' \rightarrow \tau''] \rho, (x_1, x_2) \in \mathcal{V}[\tau'] \rho
\]

\[
\vdash (e_1[F_1/f], e_2[F_2/f]) \in \mathcal{E}[\tau''] \rho
\]

\[
\Gamma; \Delta; \Theta \vdash (F_1, F_2) \in \mathcal{V}[\tau' \rightarrow \tau''] \rho
\]
• Encoding of $\mathcal{E} \llbracket \tau \rrbracket \rho$ in the logic
• More derivable rules (both equational and inequational)
• Model of the logic
• Proof of soundness of LR w.r.t. contextual equivalence
• Example proofs of contextual equivalences
• Comparison with related work
Future Work

- Generalize our approach to handle (higher-order) state
  - We’ve already done this (paper under submission)
- Explore connection to bisimulation-based methods (Sumii, Pierce, Sangiorgi, *et al.*)
- Mechanize our metatheory!
Thank You!