The Marriage of Bisimulations and Kripke Logical Relations

Chung-Kil Hur    Derek Dreyer
Georg Neis    Viktor Vafeiadis

Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern and Saarbrücken, Germany

POPL 2012
Philadelphia, USA
Canonical definition: **Contextual equivalence**

- Observable equivalence under an arbitrary context
- **Hard to reason about**, due to the quantification over arbitrary contexts

Various methods developed for **local reasoning**

- **Bisimulations and Kripke Logical Relations (KLRs)**
- Handle higher-order functions, abstract types, recursive types, general references, exceptions, continuations, etc.
Motivation #1: Marrying complementary approaches

KLRs’ treatment of local state is more powerful.
- Transition systems for controlling evolution of state.
- Subsumes the power of environmental bisimulations.

Bisimulations’ treatment of recursion is cleaner.
- Coinduction simpler and more direct than step-indexing.

Can we join them together in a single method?
Motivation #2: Inter-language reasoning

Goal: compositional equivalences between programs in different languages

- e.g., compositional certified compilation
Goal: compositional equivalences between programs
in different languages

- e.g., compositional certified compilation
Motivation #2: Inter-language reasoning

Goal: compositional equivalences between programs in different languages

- e.g., compositional certified compilation

Horizontal compositionality is preservation of equivalence under linking of modules.
Vertical compositionality is transitive composition of equivalence proofs.

C.-K. Hur, D. Dreyer, G. Neis, V. Vafeiadis
The Marriage of Bisimulations and Kripke Logical Relations
Motivation #2: Inter-language reasoning

- **Horizontal compositionality** is preservation of equivalence under linking of modules.
- **Vertical compositionality** is transitive composition of equivalence proofs.

\[ p_1 \text{ in Src} \xrightarrow{\text{tr}_1} p'_1 \text{ in IL}_1 \xrightarrow{\text{tr}_2} p''_1 \text{ in IL}_2 \xrightarrow{\text{tr}_3} p'''_1 \text{ in Asm} \]
\[ p_2 \text{ in Src} \xrightarrow{\text{tr}_4} p'_2 \text{ in IL}_3 \]
\[ p_3 \text{ in Src} \xrightarrow{\text{tr}_5} p'''_2 \text{ in Asm} \]
\[ p'_3 \text{ in Asm} \xrightarrow{\text{tr}_6} p_3 \text{ in Asm} \]
Motivation #2: Inter-language reasoning

KLRs are not transitively composable
- Due to their use of "step-indexing" for recursive features
- Hur et al. [ICFP09, POPL11] only studied one-pass compilers

Bisim’s do not scale (in an obvious way) to inter-language reasoning
- Due to their use of “syntactic” devices for H-O functions

Can we remove these limitations?
Contributions of this work

A new method for local relational reasoning:

Relation Transition Systems (RTSs)

- Combinesthe “most appealing” features of KLRs and bisimulations
- Potential to scale to inter-language reasoning
  - Does not rely on syntactic devices for H-O functions
  - Supports transitive composition of equivalence proofs
Contributions of this work

A new method for local relational reasoning:

Key idea

Don’t just support local reasoning. Demand it!

- Supports transitive composition of equivalence proofs
Existing methods support local reasoning but don’t demand it

- There’s nothing preventing one from sneaking a “brute-force” proof in through the back door

Our method will demand strictly local reasoning

- Brute-force proofs will not be permitted!

Benefit of our approach: More compositionality
Language: Simply typed $\lambda$-calculus with recursive types

$\tau \in \text{Type} ::= \alpha \mid \tau_{\text{base}} \mid \tau_1 \rightarrow \tau_2 \mid \tau_1 \times \tau_2 \mid \mu\alpha. \tau$
Coinductive approach (similar to bisimulations)

\[ \nu_1 \approx \nu_2 : \tau \quad \text{def} \]

If you want to prove \( \nu_1 \) equivalent to \( \nu_2 \),
Coinductive approach (similar to bisimulations)

\[ \nu_1 \approx \nu_2 : \tau ~ \overset{\text{def}}{=} ~ \exists \sim_L. \nu_1 \sim_L \nu_2 : \tau \]

If you want to prove \( \nu_1 \) equivalent to \( \nu_2 \),

1. Find a “local knowledge” \( \sim_L \) relating \( \nu_1 \) and \( \nu_2 \)
Coinductive approach (similar to bisimulations)

\[ \nu_1 \approx \nu_2 : \tau \overset{\text{def}}{=} \exists \sim_L. \nu_1 \sim_L \nu_2 : \tau \wedge \text{consistent}(\sim_L) \]

If you want to prove \( \nu_1 \) equivalent to \( \nu_2 \),

1. Find a “local knowledge” \( \sim_L \) relating \( \nu_1 \) and \( \nu_2 \)

2. Show that \( \sim_L \) is consistent
Coinductive approach (similar to bisimulations)

\[ v_1 \approx v_2 : \tau \quad \text{def} \quad \exists \sim_L. \; v_1 \sim_L v_2 : \tau \wedge \text{consistent}(\sim_L) \]

If you want to prove \( v_1 \) equivalent to \( v_2 \),

1. Find a “local knowledge” \( \sim_L \) relating \( v_1 \) and \( v_2 \)

2. Show that \( \sim_L \) is consistent
Restrict $\sim_L$ to only function types

Derive $\sim_L$ from $\sim_L$ by induction

$$f_1 \sim_L f_2 : \sigma \rightarrow \tau \quad c \in \llbracket \tau_{\text{base}} \rrbracket \quad \frac{\ c \sim_L c : \tau_{\text{base}}}{\ c \sim_L c : \tau_{\text{base}}}$$

$$v_1 \sim_L v_2 : \tau \quad v'_1 \sim_L v'_2 : \tau' \quad \langle v_1, v'_1 \rangle \sim_L \langle v_2, v'_2 \rangle : \tau \times \tau'$$

$$\frac{\ v_1 \sim_L v_2 : \tau[\mu\alpha.\tau/\alpha]}{\ \text{roll} \ v_1 \sim_L \text{roll} \ v_2 : \mu\alpha.\tau}$$
Coinductive approach (similar to bisimulations)

\[ \nu_1 \approx \nu_2 : \tau \overset{\text{def}}{=} \exists \sim_L . \nu_1 \sim_L \nu_2 : \tau \wedge \text{consistent}(\sim_L) \]

If you want to prove \( \nu_1 \) equivalent to \( \nu_2 \),

1. Find a “local knowledge” \( \sim_L \) relating \( \nu_1 \) and \( \nu_2 \)

2. Show that \( \sim_L \) is consistent
Coinductive approach (similar to bisimulations)

\[ \nu_1 \approx \nu_2 : \tau \quad \text{def} \quad \exists \sim_L . \, \nu_1 \overset{\sim_L}{\sim} \nu_2 : \tau \quad \land \, \text{consistent}(\sim_L) \]

If you want to prove \( \nu_1 \) equivalent to \( \nu_2 \),

1. Find a “local knowledge” \( \sim_L \) relating \( \nu_1 \) and \( \nu_2 \)

2. Show that \( \sim_L \) is consistent
Coinductive approach (similar to bisimulations)

\[ \nu_1 \sim \nu_2 : \tau \overset{\text{def}}{=} \exists \sim_L . \nu_1 \sim_L \nu_2 : \tau \land \text{consistent}(\sim_L) \]

If you want to prove \( \nu_1 \) equivalent to \( \nu_2 \),

1. Find a "local knowledge" \( \sim_L \) relating \( \nu_1 \) and \( \nu_2 \)
2. Show that \( \sim_L \) is consistent
Definition of consistent($\sim_L$)

\[ \lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau \]

\[ \equiv \]

\[ \forall v_1 \sim_1^? v_2 : \sigma. \]

\[ e_1[v_1/x] \sim_2^? e_2[v_2/x] : \tau \]
Definition of \( \sim_L \)

What should \( \sim_1 \) be?

\[
\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau
\]

\[
\Rightarrow
\]

\[
\forall \nu_1 \sim_1 \nu_2 : \sigma.
\]

\[
e_1[\nu_1/x] \sim_2 e_2[\nu_2/x] : \tau
\]
Definition of consistent($\sim_L$)

$\sim_L^? = \sim_L \quad \text{: Unsound}$

Because $v_1, v_2$ come from the context

$$\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau$$

$$\Rightarrow$$

$$\forall v_1 \sim_L^? v_2 : \sigma.$$

$$e_1[v_1/x] \sim_2^? e_2[v_2/x] : \tau$$
Definition of consistent($\sim_L$)

$\sim_1$ should be a global notion of equivalence $\sim_G$

$$\lambda x. \; e_1 \sim_L \lambda x. \; e_2 : \sigma \rightarrow \tau$$

$\implies$

$$\forall v_1 \sim_G v_2 : \sigma.\; e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau$$
Intuition: Global vs. local knowledge

$\sim_L$ represents local knowledge
- Functions our proof/module says are equivalent

$\sim_G$ represents global knowledge
- Functions the whole program says are equivalent
Intuition: Global vs. local knowledge

$\sim_L$ represents local knowledge
- Functions our proof/module says are equivalent

$\sim_G$ represents global knowledge
- Functions the whole program says are equivalent

Defining $\sim_G$ “semantically” is hard!
- It’s as hard as the original problem of finding a good relational model of ML!

So existing H-O bisimulations all define $\sim_G$ as some variation on syntactic identity
- Applicative, environmental, normal form bisim’s
Our key insight

What is $\sim_G$?

Idea: Parameterize our whole model over $\sim_G$!
We will make some assumptions about it ($\sim_G \supseteq \sim_L$), but $\sim_G$ may relate any two values at function type. $\sim_G$ can even contain "junk" like $(4 \sim_G \text{true} : \text{int} \rightarrow \text{int})$!

Highly reminiscent of the Girard/Reynolds method for reasoning about parametricity of ADTs

Takehome #1
Girard/Reynolds: Clients of ADT are parametric w.r.t. relational interpretation of abstract types
Our method: Equivalence proofs are parametric w.r.t. relational interpretation of function types

C.-K. Hur, D. Dreyer, G. Neis, V. Vafeiadis
The Marriage of Bisimulations and Kripke Logical Relations
Our key insight: Ignorance is bliss!

What is $\sim_G$? Who cares?

Idea: Parameterize our whole model over $\sim_G$!

- We will make some assumptions about it ($\sim_G \supseteq \sim_L$), but $\sim_G$ may relate any two values at function type.
- $\sim_G$ can even contain “junk” like ($4 \sim_G \text{true} : \text{int} \rightarrow \text{int}$)!
- Highly reminiscent of the Girard/Reynolds method for reasoning about parametricity of ADTs
Our key insight: Ignorance is bliss!

What is $\sim_G$? Who cares?

Takehome #1

- **Girard/Reynolds**: Clients of ADT are parametric w.r.t. relational interpretation of abstract types

- **Our method**: Equivalence proofs are parametric w.r.t. relational interpretation of function types
Definition of consistent($\sim_L$)

Instead of defining $\sim_G$ . . .

\[
\begin{align*}
\lambda x. e_1 & \sim_L \lambda x. e_2 : \sigma \rightarrow \tau \\
\iff \\
\forall v_1 \sim_G v_2 : \sigma. \\
e_1[v_1/x] & \sim_2^{?} e_2[v_2/x] : \tau
\end{align*}
\]
... we parameterize over \( \sim_G \)!

\[
\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau
\]

\[
\Rightarrow
\]

\[
\forall \sim_G \supseteq \sim_L . \forall v_1 \sim_G v_2 : \sigma .
\]

\[
e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau
\]
Definition of consistent ($\sim_L$)

What should $\sim_2$ be?

$$\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau$$

$$\Rightarrow$$

$$\forall \sim_G \supseteq \sim_L . \forall v_1 \sim_G v_2 : \sigma . \ e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau$$

C.-K. Hur, D. Dreyer, G. Neis, V. Vafeiadis

The Marriage of Bisimulations and Kripke Logical Relations
Definition of consistent ($\sim_L$)

Both diverge or both converge to related values?

$$\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau$$

$$\iff$$

$$\forall \sim_G \supseteq \sim_L . \forall v_1 \sim_G v_2 : \sigma .

\quad e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau$$

C.-K. Hur, D. Dreyer, G. Neis, V. Vafeiadis

The Marriage of Bisimulations and Kripke Logical Relations
Definition of consistent ($\sim_L$)

Both diverge or both converge to related values?

$$
\lambda f. f(0) \sim_L \lambda f. f(0) : (\text{int} \rightarrow \text{int}) \rightarrow \text{int}
$$

$$
\rightarrow

\forall \sim_G \supseteq \sim_L . \forall \upsilon_1 \sim_G \upsilon_2 : \text{int} \rightarrow \text{int}.

\upsilon_1(0) \sim_2 \upsilon_2(0) : \text{int}
$$
Definition of \textit{consistent}(\sim_L)

Both diverge or both converge to related values?

\[
\lambda f. f(0) \sim_L \lambda f. f(0) : (\text{int} \rightarrow \text{int}) \rightarrow \text{int}
\]

\[
\forall \sim_G \supseteq \sim_L . \; 4 \sim_G \text{true} : \text{int} \rightarrow \text{int}.
\]

\[
4(0) \sim_2 \text{true}(0) : \text{int}
\]
Definition of consistent($\sim_L$)

$\sim^2_2$ should be “local term equivalence” $\sim^\text{exp}_G$

$$\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau$$

$$\forall \sim_G \supseteq \sim_L. \ \forall v_1 \sim_G v_2 : \sigma.$$  

$$e_1[v_1/x] \sim^\text{exp}_G e_2[v_2/x] : \tau$$

C.-K. Hur, D. Dreyer, G. Neis, V. Vafeiadis
The Marriage of Bisimulations and Kripke Logical Relations
Intuition: Local term equivalence

To show your terms are locally equivalent

\[ e_1 \sim^\exp_G e_2 \]
Intuition: Local term equivalence

Execute them “locally” until...

\[ e_1 \sim^{G} \exp \rightarrow^* \rightarrow^* K_1[f_1(v_1)] \]

\[ e_2 \sim^{G} \exp \rightarrow^* \rightarrow^* K_2[f_2(v_2)] \]
Intuition: Local term equivalence

...they pass control to “external” functions

\[ e_1 \sim^G e_2 \]

\[ K_1[f_1(v_1)] \leftrightarrow K_2[f_2(v_2)] \]
Assume you get back control with related return values

\[ e_1 \sim^{\exp}_G e_2 \]

\[ K_1[f_1(v_1)] \rightarrow^* K_1[r_1] \]

\[ K_2[f_2(v_2)] \rightarrow^* K_2[r_2] \]
Show your continuations are locally equivalent

\[
\begin{align*}
e_1 & \sim^G e_2 \\
K_1[f_1(v_1)] & \xrightarrow{*} K_2[f_2(v_2)] \\
K_1[r_1] & \xrightarrow{*} K_2[r_2]
\end{align*}
\]
Definition of local term equivalence $\sim^\text{exp}_G$

- Derive $\sim^\text{exp}_G$ from $\sim_G$ by coinduction

\[ e_1 \sim^\text{exp}_G e_2 : \top \]
Definition of local term equivalence $\sim^\text{exp}_G$

- Derive $\sim^\text{exp}_G$ from $\sim_G$ by coinduction

\[
\begin{align*}
e_1 & \sim^\text{exp}_G e_2 \\
\downarrow \omega & \downarrow \omega
\end{align*}
\]

**Case 1**: Both diverge
Definition of local term equivalence $\sim^\text{exp}_G$

- Derive $\sim^\text{exp}_G$ from $\sim_G$ by coinduction

$$
\begin{align*}
\tau_1 \Downarrow^* & \quad \sim_G \quad \tau_2 \\
\tau_2 & \quad \Downarrow^* \\
\end{align*}
$$

Case 2: Both terminate
Definition of local term equivalence $\sim_{\text{exp}}^G$

- Derive $\sim_{\text{exp}}^G$ from $\sim_G$ by coinduction

\[
\begin{align*}
\text{e}_1 & \sim^G \quad \text{e}_2 : \tau \\
\downarrow^* & \quad \downarrow^* \\
K_1[f_1(v_1)] & \quad K_2[f_2(v_2)] : \tau
\end{align*}
\]

1. $f_1 \sim_G f_2 : \tau_{\text{arg}} \rightarrow \tau_{\text{ret}}$
2. $v_1 \sim_G v_2 : \tau_{\text{arg}}$
3. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \ K_1[r_1] \sim^G_{\text{exp}} K_2[r_2] : \tau$

Case 3: Both call a function
Definition of local term equivalence $\sim^\text{exp}_G$

- Derive $\sim^\text{exp}_G$ from $\sim_G$ by coinduction

\[ e_1 \sim^\text{exp}_G e_2 : \tau \]

\[ K_1[f_1(v_1)] \quad K_2[f_2(v_2)] : \tau \]

1. \[ f_1 \sim_G f_2 : \tau_{\text{arg}} \rightarrow \tau_{\text{ret}} \]
2. \[ v_1 \sim_G v_2 : \tau_{\text{arg}} \]
3. \[ \forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \quad K_1[r_1] \sim^\text{exp}_G K_2[r_2] : \tau \]

**Case 3:** Both call a function
Definition of local term equivalence $\sim^{\text{exp}}_G$

- Derive $\sim^{\text{exp}}_G$ from $\sim_G$ by coinduction

\[
\begin{align*}
\text{Case 3: Both call a function} & \\
\text{Takehome #2} & \quad \text{Since our proofs are parametric w.r.t. } \sim_G, \text{ we CAN and we MUST reason locally!}
\end{align*}
\]
Definition of local term equivalence $\sim^{\text{exp}}_G$

- Derive $\sim^{\text{exp}}_G$ from $\sim_G$ by coinduction

\[
\begin{align*}
  &\quad e_1 \sim_G e_2 : \mathcal{T} \\
  &\downarrow^* \downarrow^* \\
  &K_1[f_1(v_1)] \quad K_2[f_2(v_2)] : \mathcal{T}
\end{align*}
\]

1. $f_1 \sim_G f_2 : \tau_{\text{arg}} \rightarrow \tau_{\text{ret}}$
2. $v_1 \sim_G v_2 : \tau_{\text{arg}}$
3. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \ K_1[r_1] \sim^{\text{exp}}_G K_2[r_2] : \mathcal{T}$

**Case 3:** Both call a function
Definition of local term equivalence $\sim_{\text{exp}}^G$

- Derive $\sim_{\text{exp}}^G$ from $\sim_G$ by coinduction

$e_1 \sim_{\text{exp}}^G e_2 : \tau$

$\downarrow^\ast$

$K_1[f_1(v_1)]$

$\downarrow^\ast$

$K_2[f_2(v_2)] : \tau$

1. $f_1 \sim_G f_2 : \tau_{\text{arg}} \rightarrow \tau_{\text{ret}}$
2. $v_1 \sim_G v_2 : \tau_{\text{arg}}$
3. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}$. $K_1[r_1] \sim_{\text{exp}}^G K_2[r_2] : \tau$

**Case 3**: Both call a function
Definition of local term equivalence \( \sim_{\text{exp}}^G \)

- Derive \( \sim_{\text{exp}}^G \) from \( \sim_G \) by coinduction

Takehome #2

Since our proofs are parametric w.r.t. \( \sim_G \), we CAN and we MUST reason locally!

Case 3: Both call a function
Summary: The benefits of “proof parametricity”

1. Horizontal compositionality (aka congruence)
   - The less proofs about different modules assume about $\sim_G$, the easier they are to link together

2. Vertical compositionality (aka transitivity)
   - Since equivalence proofs must use “local” reasoning, their structure is highly constrained, making them easier to compose transitively
Closely related work

“Normal form” (or “open”) bisimulations
- Related fcn arguments represented by a fresh variable \( x \)
- Hence, bisimulation must account for terms getting stuck
- Definition very similar to our “local term equivalence”

Mendler-style coinduction
- \( L \) is a “robustly post-fixed point (rpfp)” of an endofunction \( F \) if \( \forall G \geq L. \ L \leq F(G) \)
- Rpfp’s are closed under joins even for non-monotone \( F \)
- Our consistency condition is a variant of this
Generalization to open terms
- Requires parameterizing $\sim_L$ over $\sim_G$

Extension of model with abstract types
- Based on [Sumii-Pierce ’05]

Extension of model with higher-order state
- Based on [Dreyer-Neis-Birkedal ’10]

Transitivity proved for pure fragment
- Proof for full model currently under submission

All results mechanized in Coq

Future work:
- Inter-language reasoning (certified ML/C compilers with FFI)
- Supporting refined type system (e.g., effect system)
- Supporting concurrency