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
Motivation #2: Inter-language reasoning

Goal: compositional equivalences between programs in different languages

- e.g., compositional certified compilation

![Diagram]

Compiler

\[ p_1 \text{ in } \text{Src} \]
\[ \text{tr}_1 \]
\[ p'_1 \text{ in } \text{IL}_1 \]
\[ \text{tr}_2 \]
\[ p''_1 \text{ in } \text{IL}_2 \]
\[ \text{tr}_3 \]
\[ p'''_1 \text{ in } \text{Asm} \]
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.
Motivation #2: Inter-language reasoning

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

- Combines the “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
Key idea

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 λ-calculus with recursive types

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

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

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)

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

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
Coinductive approach (similar to bisimulations)

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

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

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

The Marriage of Bisimulations and Kripke Logical Relations
Value closure $\sim_L = \text{Value equivalence modulo } \sim_L$

1. **Restrict $\sim_L$ to only function types**

2. **Derive $\overline{\sim_L}$ from $\sim_L$ by induction**

   \[
   \frac{f_1 \sim_L f_2 : \sigma \to \tau}{f_1 \overline{\sim_L} f_2 : \sigma \to \tau}
   \]

   \[
   \frac{c \in \llbracket \tau_{\text{base}} \rrbracket}{c \overline{\sim_L} c : \tau_{\text{base}}}
   \]

   \[
   \frac{v_1 \sim_L v_2 : \tau}{\langle v_1, v'_1 \rangle \overline{\sim_L} \langle v_2, v'_2 \rangle : \tau \times \tau'}
   \]

   \[
   \frac{v_1 \sim_L v_2 : \tau}{\text{roll } v_1 \overline{\sim_L} \text{roll } v_2 : \mu\alpha. \tau}
   \]

   \[
   \frac{v_1 \sim_L v_2 : \tau}{\langle v_1, v'_1 \rangle \overline{\sim_L} \langle v_2, v'_2 \rangle : \tau \times \tau'}
   \]
Coinductive approach (similar to bisimulations)

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

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
Coinductive approach (similar to bisimulations)

\[ \forall_1 \approx \forall_2 : \tau \overset{\text{def}}{=} \exists \sim_L . \forall_1 \sim_L \forall_2 : \tau \land \text{consistent}(\sim_L) \]

If you want to prove \( \forall_1 \) equivalent to \( \forall_2 \),

1. Find a “local knowledge” \( \sim_L \) relating \( \forall_1 \) and \( \forall_2 \)

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

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

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 \textit{consistent}(\sim_L)

\[
\begin{align*}
\lambda x. \ e_1 \ & \sim_L \ \lambda x. \ e_2 \ : \ \sigma \rightarrow \tau \\
\iff \\
\forall \nu_1 \ & \sim_1 \ \nu_2 \ : \ \sigma. \\
e_1[\nu_1/x] \ & \sim_2 \ e_2[\nu_2/x] \ : \ \tau
\end{align*}
\]
Definition of consistent ($\sim_L$)

What should $\sim_1$ be?

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

$$\implies$$

$$\forall v_1 \sim_1 v_2 : \sigma.$$  

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

$\sim_1 ? = \overline{\sim_L} : \text{Unsound}$

Because $\nu_1, \nu_2$ come from the context

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

$\Rightarrow$

$\forall \nu_1 \sim_1 ? \nu_2 : \sigma.$

$e_1[\nu_1/x] \sim_2 ? e_2[\nu_2/x] : \tau$
$\sim_1$ should be a **global** notion of equivalence $\sim_G$.

$$
\lambda x. e_1 \sim_L \lambda x. e_2 : \sigma \rightarrow \tau \quad \implies \quad \forall v_1 \sim_G v_2 : \sigma. \\
\ e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau
$$
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 \( \sim_G \)?
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 $\sim_G$ true : int $\rightarrow$ int)!
- Highly reminiscent of the Girard/Reynolds method for reasoning about parametricity of ADTs
Our key insight: Ignorance is bliss!

What is $\sim$? 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 $\sim_G$ . . .

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

$$\Rightarrow$$

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

$$e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau$$
... we parameterize over $\sim_G$!

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

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

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

What should \( \sim_2 \) be?

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

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

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

Both diverge or both converge to related values?

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

\[ \Rightarrow \]

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

\[ e_1[v_1/x] \sim_2 e_2[v_2/x] : \tau \]
Both diverge or both converge to related values?

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

\[ \Rightarrow \]

\[ \forall \sim_G \supseteq \sim_L . \forall v_1 \sim_G v_2 : \text{int} \rightarrow \text{int}. \]

\[ v_1(0) \sim^?_2 v_2(0) : \text{int} \]
Definition of consistent \((\sim_L)\)

Both diverge or both converge to related values?

\[
\lambda f. f(0) \sim_L \lambda f. f(0) : (\text{int} \rightarrow \text{int}) \rightarrow \text{int} \quad \Rightarrow \\
\forall \sim_G \supseteq \sim_L . 
4 \sim_G \text{true} : \text{int} \rightarrow \text{int}.
\]

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

$\sim_2$ should be “local term equivalence” $\sim_{\exp}^G$

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

$$\implies$$

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

$$e_1[v_1/x] \sim_{\exp}^G e_2[v_2/x] : \tau$$
To show your terms are locally equivalent

\[ e_1 \sim^{\exp}_G e_2 \]
Intuition: Local term equivalence

Execute them “locally” until...

\[ e_1 \xrightarrow{\star} K_1[f_1(v_1)] \]

\[ e_2 \xrightarrow{\exp G} K_2[f_2(v_2)] \]
Intuition: Local term equivalence

...they pass control to “external” functions

\[ \mathcal{e}_1 \sim^G \mathcal{e}_2 \]

\[ \mathcal{K}_1[f_1(v_1)] \quad \mathcal{K}_2[f_2(v_2)] \]
Assume you get back control with related return values.
Intuition: Local term equivalence

Show your continuations are locally equivalent

\[ e_1 \sim^G e_2 \]

\[ [f_1(v_1)] \quad [f_2(v_2)] \]

\[ r_1 \quad r_2 \]
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

\[
\begin{align*}
\tau & \downarrow^* \\
\nu_1 & \sim_G \\
\end{align*}
\]

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

\[
\begin{align*}
\tau & \downarrow^* \\
\nu_2 & \sim_G \\
\end{align*}
\]

**Case 2**: Both terminate
Definition of local term equivalence $\sim^\text{exp}_G$

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

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

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. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \ K_1[r_1] \sim^\text{exp}_G K_2[r_2] : \tau$

**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

\[
\begin{align*}
    e_1 & \sim^\text{exp}_G e_2 : \tau \\
    K_1[f_1(v_1)] & \downarrow^* \\
    K_2[f_2(v_2)] & \downarrow^* : \tau
\end{align*}
\]

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. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \ K_1[r_1] \sim^\text{exp}_G K_2[r_2] : \tau$

**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

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

\[ \downarrow * \]

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

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

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

The Marriage of Bisimulations and Kripke Logical Relations
Definition of local term equivalence $\sim_{\text{exp}}^G$

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

\[
\begin{align*}
\downarrow^* & & \downarrow^* \\
K_1[f_1(v_1)] & & K_2[f_2(v_2)] : \tau
\end{align*}
\]

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. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \ K_1[r_1] \sim_{\text{exp}}^G K_2[r_2] : \tau$

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

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

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. $\forall r_1 \sim_G r_2 : \tau_{\text{ret}}. \ K_1[r_1] \sim^{\text{exp}}_G K_2[r_2] : \tau$

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

Case 3: Both call a function

Takehome #2

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

<table>
<thead>
<tr>
<th align="center">$\tau$ :=</th>
<th align="center">$\mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int}$</th>
</tr>
</thead>
<tbody>
<tr>
<td align="center">$\text{foo}_1(f)$ :=</td>
<td align="center">$666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int}$</td>
</tr>
<tr>
<td align="center">$\text{foo}_2(f)$ :=</td>
<td align="center">$(\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int}$</td>
</tr>
</tbody>
</table>

Want to show

$$\text{foo}_1 \approx \text{foo}_2$$
Simple illustrating example

\[ \tau := \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int} \]
\[ \text{foo}_1(f) := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int} \]
\[ \text{foo}_2(f) := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int} \]

1 Define \( \sim_L \) as \( \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \rightarrow \text{int} \} \)
Simple illustrating example

\[ \tau \ := \ \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int} \]

\[ \text{foo}_1(f) \ := \ 666 + (\text{unroll } f)(\text{foo}_1) \ : \ \tau \rightarrow \text{int} \]

\[ \text{foo}_2(f) \ := \ (\text{unroll } f)(\text{foo}_2) + 666 \ : \ \tau \rightarrow \text{int} \]

1. Define \( \sim_L \) as \( \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \rightarrow \text{int} \} \)

2. Show consistent(\( \sim_L \))
Simple illustrating example

$$
\tau \ := \ \mu \alpha. \ (\alpha \to \text{int}) \to \text{int}
$$

$$
\text{foo}_1(f) \ := \ 666 + (\text{unroll } f)(\text{foo}_1) \ : \ \tau \to \text{int}
$$

$$
\text{foo}_2(f) \ := \ (\text{unroll } f)(\text{foo}_2) + 666 \ : \ \tau \to \text{int}
$$

1 Define $$\sim_L$$ as \{ $$\text{foo}_1 \sim_L \text{foo}_2 : \tau \to \text{int}$$ \}

2 $$\forall \sim_G \supseteq \sim_L . \ \forall v_1 \sim_G v_2 : \tau.$$  

$$
666 + (\text{unroll } v_1)(\text{foo}_1) \sim_G^{\text{exp}} (\text{unroll } v_2)(\text{foo}_2) + 666
$$
Simple illustrating example

\[
\begin{align*}
\tau & := \mu \alpha. (\alpha \to \text{int}) \to \text{int} \\
\text{foo}_1(f) & := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \to \text{int} \\
\text{foo}_2(f) & := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \to \text{int}
\end{align*}
\]

1. Define \( \sim_L \) as \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \to \text{int} \} 

2. \( \forall \sim_G \supseteq \sim_L . \forall f_1 \sim_G f_2 : (\tau \to \text{int}) \to \text{int} \).

\[666 + (\text{unroll } (\text{roll } f_1))(\text{foo}_1) \sim^\text{exp}_G (\text{unroll } (\text{roll } f_2))(\text{foo}_2) + 666\]
Simple illustrating example

\[ \tau := \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int} \]

\[ \text{foo}_1(f) := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int} \]

\[ \text{foo}_2(f) := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int} \]

1. Define \( \sim_L \) as \( \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \rightarrow \text{int} \} \)

2. \( \forall \sim_G \supseteq \sim_L . \forall f_1 \sim_G f_2 : (\tau \rightarrow \text{int}) \rightarrow \text{int} \).

\[
666 + (\text{unroll (roll } f_1))(\text{foo}_1) \overset{\text{exp}}{\sim}_G \text{ (unroll (roll } f_2))(\text{foo}_2) + 666
\]

\[
\downarrow^* \quad \downarrow^*
\]

\[ 666 + f_1(\text{foo}_1) \quad f_2(\text{foo}_2) + 666 \]

**Case 3:** Both call a function
Simple illustrating example

\[ \tau := \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int} \]

\[ \text{foo}_1(f) := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int} \]

\[ \text{foo}_2(f) := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int} \]

1. Define \( \sim_L \) as \( \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \rightarrow \text{int} \} \)

2. \( \forall \sim_G \supseteq \sim_L . \forall f_1 \sim_G f_2 : (\tau \rightarrow \text{int}) \rightarrow \text{int}. \)

\[ 666 + (\text{unroll } \text{roll } f_1)(\text{foo}_1) \sim^\text{exp}_G (\text{unroll } \text{roll } f_2)(\text{foo}_2) + 666 \]

\[ \downarrow^* \]

\[ 666 + f_1(\text{foo}_1) \sim_G f_2(\text{foo}_2) + 666 \]

Case 3: Both call a function

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

The Marriage of Bisimulations and Kripke Logical Relations
Simple illustrating example

$$\tau := \mu \alpha. (\alpha \to \text{int}) \to \text{int}$$

$$\text{foo}_1(f) := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \to \text{int}$$

$$\text{foo}_2(f) := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \to \text{int}$$

1. Define $$\sim_L$$ as \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \to \text{int} \}

2. $$\forall \sim_G \supseteq \sim_L. \forall f_1 \sim_G f_2 : (\tau \to \text{int}) \to \text{int}.$$

$$666 + (\text{unroll } (\text{roll } f_1))(\text{foo}_1) \sim^\text{exp}_G (\text{unroll } (\text{roll } f_2))(\text{foo}_2) + 666$$

$$\downarrow^*$$

$$666 + f_1(\text{foo}_1) \sim^G f_2(\text{foo}_2) + 666$$

Case 3: Both call a function
Simple illustrating example

\[
\tau := \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int}
\]

\[
\text{foo}_1(f) := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int}
\]

\[
\text{foo}_2(f) := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int}
\]

1. Define \(\sim_L\) as \(\{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \rightarrow \text{int} \}\)

2. \(\forall \sim_G \supseteq \sim_L. \forall f_1 \sim_G f_2 : (\tau \rightarrow \text{int}) \rightarrow \text{int}.\)

\[
666 + (\text{unroll } (\text{roll } f_1))(\text{foo}_1) \sim^\text{exp}_G (\text{unroll } (\text{roll } f_2))(\text{foo}_2) + 666
\]

\[
\downarrow^* \\
666 + f_1(\text{foo}_1)
\]

\[
\downarrow^* \\
f_2(\text{foo}_2) + 666
\]

**Case 3:** Both call a function
Simple illustrating example

\[\tau := \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int}\]

\[\text{foo}_1(f) := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int}\]

\[\text{foo}_2(f) := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int}\]

1. Define \(\sim_L\) as \(\{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \rightarrow \text{int} \}\)

2. \(\forall \sim_G \supseteq \sim_L . \forall r_1 \sim_G r_2 : \text{int.} \)

\[666 + r_1 \sim^\text{exp}_G r_2 + 666\]
Simple illustrating example

\[ \tau : = \mu \alpha . (\alpha \to \text{int}) \to \text{int} \]
\[ \text{foo}_1(f) : = 666 + (\text{unroll } f)(\text{foo}_1) : \tau \to \text{int} \]
\[ \text{foo}_2(f) : = (\text{unroll } f)(\text{foo}_2) + 666 : \tau \to \text{int} \]

1. Define \( \sim_L \) as \( \{ \text{foo}_1 \sim_L \text{foo}_2 : \tau \to \text{int} \} \)

2. \( \forall \sim_G \supseteq \sim_L . \forall n \in [\text{int}] . \)

\[ 666 + n \sim_G^{\exp} n + 666 \]
Simple illustrating example

\[
\begin{align*}
\tau & := \mu \alpha. (\alpha \rightarrow \text{int}) \rightarrow \text{int} \\
\text{foo}_1(f) & := 666 + (\text{unroll } f)(\text{foo}_1) : \tau \rightarrow \text{int} \\
\text{foo}_2(f) & := (\text{unroll } f)(\text{foo}_2) + 666 : \tau \rightarrow \text{int}
\end{align*}
\]

1. Define \( \sim^L \) as \( \{ \text{foo}_1 \sim^L \text{foo}_2 : \tau \rightarrow \text{int} \} \)
2. \( \forall \sim_G \supseteq \sim^L . \forall n \in [\text{int}] . \)

\[
\begin{align*}
666 + n & \sim^\text{exp}_G n + 666 \\
\downarrow^* & \quad \downarrow^*
\end{align*}
\]

\[
\begin{align*}
n + 666 & \sim^G n + 666 \quad \text{Case 2: Both terminates}
\end{align*}
\]
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
What else is in the paper?

- 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