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

Program equivalence in ML-like languages

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

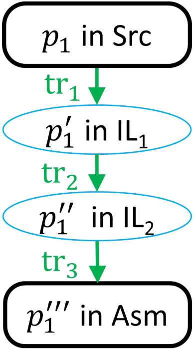
- *e.g.*, compositional certified compilation

Motivation #2: Inter-language reasoning

Goal: compositional equivalences between programs in different languages

- *e.g.*, compositional certified compilation

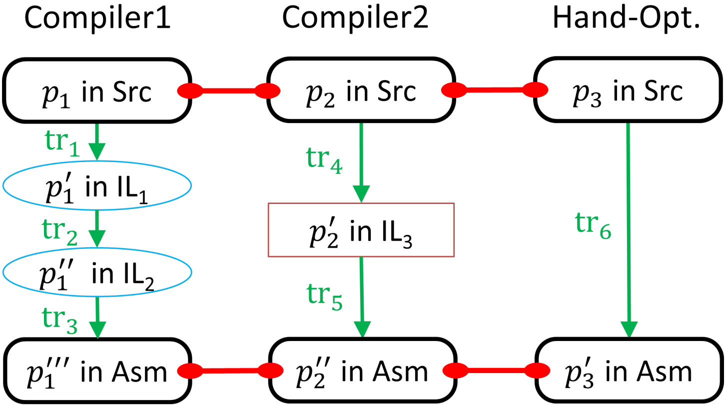
Compiler



Motivation #2: Inter-language reasoning

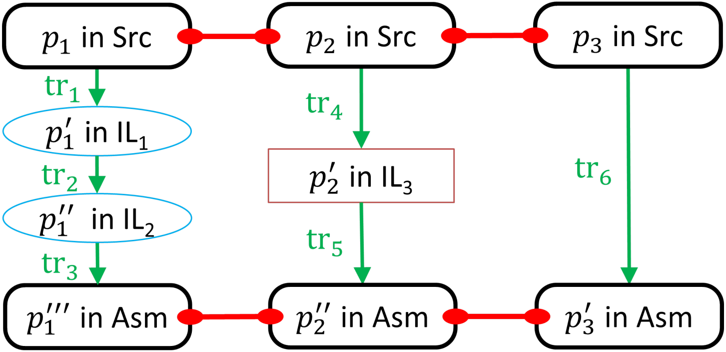
Goal: compositional equivalences between programs in different languages

- *e.g.*, compositional certified compilation



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

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

$$\tau \in \mathrm{Type} \ ::= \ \alpha \mid \tau_{\mathrm{base}} \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2 \mid \mu\alpha.\,\tau$$

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \stackrel{\text{def}}{=}$$

If you want to prove v_1 equivalent to v_2,

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \stackrel{\text{def}}{=} \exists \sim_{\mathrm{L}}.\ v_1 \sim_{\mathrm{L}} v_2 : \tau$$

If you want to prove v_1 equivalent to v_2,

1. Find a "local knowledge" $\sim_{\mathrm{L}}$ relating v_1 and v_2

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \quad \stackrel{\text{def}}{=} \quad \exists \sim_{\mathrm{L}}.\ v_1 \sim_{\mathrm{L}} v_2 : \tau \\ \wedge\ \mathit{consistent}(\sim_{\mathrm{L}})$$

If you want to prove v_1 equivalent to v_2,

1. Find a "local knowledge" $\sim_{\mathrm{L}}$ relating v_1 and v_2
2. Show that $\sim_{\mathrm{L}}$ is consistent

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \quad \stackrel{\text{def}}{=} \quad \exists \boxed{\sim_{\mathrm{L}}}.\ v_1 \sim_{\mathrm{L}} v_2 : \tau \\ \wedge\ \mathit{consistent}(\sim_{\mathrm{L}})$$

If you want to prove v_1 equivalent to v_2,

1. Find a "local knowledge" $\sim_{\mathrm{L}}$ relating v_1 and v_2
2. Show that $\sim_{\mathrm{L}}$ is consistent

Value closure $\overline{\sim_{\mathrm{L}}}$ = Value equivalence modulo $\sim_{\mathrm{L}}$

1. Restrict $\sim_{\mathrm{L}}$ to only function types
2. Derive $\overline{\sim_{\mathrm{L}}}$ from $\sim_{\mathrm{L}}$ by induction

$$\frac{f_1 \sim_{\mathrm{L}} f_2 \;:\; \sigma \to \tau}{f_1 \overline{\sim_{\mathrm{L}}} f_2 \;:\; \sigma \to \tau} \qquad \frac{c \in [\![\tau_{\mathrm{base}}]\!]}{c \overline{\sim_{\mathrm{L}}} c \;:\; \tau_{\mathrm{base}}}$$

$$\frac{v_1 \overline{\sim_{\mathrm{L}}} v_2 \;:\; \tau \qquad v_1' \overline{\sim_{\mathrm{L}}} v_2' \;:\; \tau'}{\langle v_1, v_1' \rangle \overline{\sim_{\mathrm{L}}} \langle v_2, v_2' \rangle \;:\; \tau \times \tau'}$$

$$\frac{v_1 \overline{\sim_{\mathrm{L}}} v_2 \;:\; \tau[\mu\alpha.\,\tau/\alpha]}{\mathsf{roll}\ v_1 \overline{\sim_{\mathrm{L}}} \mathsf{roll}\ v_2 \;:\; \mu\alpha.\,\tau}$$

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \quad \stackrel{\text{def}}{=} \quad \exists \sim_{\mathrm{L}} .\ v_1 \sim_{\mathrm{L}} v_2 : \tau$$
$$\wedge\ \mathit{consistent}(\sim_{\mathrm{L}})$$

If you want to prove v_1 equivalent to v_2,

1. Find a "local knowledge" $\sim_{\mathrm{L}}$ relating v_1 and v_2
2. Show that $\sim_{\mathrm{L}}$ is consistent

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \stackrel{\text{def}}{=} \exists \sim_{\mathrm{L}} .\ v_1 \overline{\sim_{\mathrm{L}}} v_2 : \tau$$
$$\wedge\ \mathit{consistent}(\sim_{\mathrm{L}})$$

If you want to prove v_1 equivalent to v_2,

1. Find a "local knowledge" $\sim_{\mathrm{L}}$ relating v_1 and v_2
2. Show that $\sim_{\mathrm{L}}$ is consistent

Coinductive approach (similar to bisimulations)

$$v_1 \approx v_2 : \tau \;\stackrel{\text{def}}{=}\; \exists \sim_{\mathrm{L}} .\; v_1 \,\overline{\sim_{\mathrm{L}}}\, v_2 : \tau \;\wedge\; \boxed{consistent(\sim_{\mathrm{L}})}$$

If you want to prove v_1 equivalent to v_2,

1. Find a "local knowledge" $\sim_{\mathrm{L}}$ relating v_1 and v_2
2. Show that $\sim_{\mathrm{L}}$ is consistent

Definition of *consistent*($\sim_{\mathrm{L}}$)

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \to \tau$$

$$\Longrightarrow$$

$$\forall v_1 \sim^{?}_{1} v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \sim^{?}_{2} e_2[v_2/x] \;:\; \tau$$

What should $\sim_1^?$ be?

$$\lambda x.\, e_1 \ \sim_{\mathrm{L}} \ \lambda x.\, e_2 \ : \ \sigma \to \tau$$

$$\Longrightarrow$$

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

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

Definition of *consistent*$(\sim_{\mathrm{L}})$

$$\sim_1^? = \overline{\sim_{\mathrm{L}}} \ : \text{Unsound}$$

Because v_1, v_2 come from the context

$$\lambda x.\, e_1 \ \sim_{\mathrm{L}} \ \lambda x.\, e_2 \ : \ \sigma \to \tau$$

$$\Longrightarrow$$

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

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

$\sim_1^?$ should be a global notion of equivalence $\overline{\sim_{\mathrm{G}}}$

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \to \tau$$

$$\Longrightarrow$$

$$\forall v_1 \;\overline{\sim_{\mathrm{G}}}\; v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \;\sim_2^?\; e_2[v_2/x] \;:\; \tau$$

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

$\sim_{\mathrm{L}}$ represents local knowledge

- Functions our proof/module says are equivalent

$\sim_{\mathrm{G}}$ represents global knowledge

- Functions the whole program says are equivalent

Defining $\sim_{\mathrm{G}}$ "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 $\sim_{\mathrm{G}}$ as some variation on syntactic identity

- Applicative, environmental, normal form bisim's

What is $\sim_{\mathrm{G}}$?

What is $\sim_{\mathrm{G}}$? Who cares?

Idea: Parameterize our whole model over $\sim_{\mathrm{G}}$!

- We will make *some* assumptions about it ($\sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}$), but $\sim_{\mathrm{G}}$ may relate any two values at function type.
- $\sim_{\mathrm{G}}$ can even contain "junk" like ($4 \sim_{\mathrm{G}} \mathsf{true} : \mathsf{int} \to \mathsf{int}$)!
- Highly reminiscent of the Girard/Reynolds method for reasoning about parametricity of ADTs

Our key insight: Ignorance is bliss!

What is … ? 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*$(\sim_{\mathrm{L}})$

Instead of defining $\sim_{\mathrm{G}}$ ...

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \to \tau$$

$$\Longrightarrow$$

$$\forall v_1 \;\overline{\sim_{\mathrm{G}}}\; v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \;\sim_2^{?}\; e_2[v_2/x] \;:\; \tau$$

. . . we parameterize over $\sim_{\mathrm{G}}$!

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \to \tau$$

$$\Longrightarrow$$

$$\boxed{\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}} .}\; \forall v_1 \;\overline{\sim_{\mathrm{G}}}\; v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \;\sim^{?}_{2}\; e_2[v_2/x] \;:\; \tau$$

What should $\sim_2^?$ be?

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \to \tau$$

$$\Longrightarrow$$

$$\boxed{\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}} .}\; \forall v_1 \,\overline{\sim_{\mathrm{G}}}\, v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \;\sim_2^?\; e_2[v_2/x] \;:\; \tau$$

Both diverge or both converge to related values?

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \to \tau$$

$$\Longrightarrow$$

$$\boxed{\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.}\; \forall v_1 \;\overline{\sim_{\mathrm{G}}}\; v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \;\sim_2^{?}\; e_2[v_2/x] \;:\; \tau$$

Both diverge or both converge to related values?

$$\lambda f.\, f(0) \;\sim_{\mathrm{L}}\; \lambda f.\, f(0) \;:\; (\mathsf{int} \to \mathsf{int}) \to \mathsf{int}$$

$$\Longrightarrow$$

$$\boxed{\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.}\; \forall v_1 \;\overline{\sim_{\mathrm{G}}}\; v_2 \;:\; \mathsf{int} \to \mathsf{int}.$$

$$v_1(0) \;\sim_2^{?}\; v_2(0) \;:\; \mathsf{int}$$

Both diverge or both converge to related values?

$$\lambda f.\, f(0) \;\sim_{\mathrm{L}}\; \lambda f.\, f(0) \quad : \quad (\mathsf{int} \to \mathsf{int}) \to \mathsf{int}$$

$$\Longrightarrow$$

$$\boxed{\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.}\; 4 \;\overline{\sim_{\mathrm{G}}}\; \mathsf{true} \quad : \quad \mathsf{int} \to \mathsf{int}.$$

$$4(0) \;\sim_2^{?}\; \mathsf{true}(0) \quad : \quad \mathsf{int}$$

Definition of *consistent*$(\sim_{\mathrm{L}})$

$\sim_2^?$ should be "local term equivalence" $\sim_{\mathrm{G}}^{\mathrm{exp}}$

$$\lambda x.\, e_1 \;\sim_{\mathrm{L}}\; \lambda x.\, e_2 \;:\; \sigma \rightarrow \tau$$

$$\Longrightarrow$$

$$\boxed{\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.}\; \forall v_1 \;\overline{\sim_{\mathrm{G}}}\; v_2 \;:\; \sigma.$$

$$e_1[v_1/x] \sim_{\mathrm{G}}^{\mathrm{exp}} e_2[v_2/x] \;:\; \tau$$

To show your terms are locally equivalent

$$e_1 \quad \sim_{\mathrm{G}}^{\mathsf{exp}} \quad e_2$$

Execute them “locally” until...

$$
\begin{array}{ccc}
e_1 & \sim^{\mathsf{exp}}_{\mathrm{G}} & e_2 \\
\big\downarrow^{*} & & \big\downarrow^{*} \\
K_1[f_1(v_1)] & & K_2[f_2(v_2)]
\end{array}
$$

. . . they pass control to “external” functions

$$e_1 \quad \sim_{\mathrm{G}}^{\mathrm{exp}} \quad e_2$$

$$\downarrow^{*} \qquad\qquad \downarrow^{*}$$

$$\leftsquigarrow \; K_1[f_1(v_1)] \qquad\qquad K_2[f_2(v_2)] \; \rightsquigarrow$$

Intuition: Local term equivalence

Assume you get back control
with related return values

$$
\begin{array}{ccccc}
 & e_1 & \sim^{\text{exp}}_{\mathrm{G}} & e_2 & \\
 & \downarrow^{*} & & \downarrow^{*} & \\
\leftarrow\!\!\rightsquigarrow & K_1[f_1(v_1)] & & K_2[f_2(v_2)] & \rightsquigarrow \\
\rightsquigarrow & K_1[r_1] & & K_2[r_2] & \leftarrow\!\!\rightsquigarrow
\end{array}
$$

Intuition: Local term equivalence

Show your continuations are locally equivalent

$$
\begin{tikzcd}
 & e_1 \arrow[d, "*"] & \sim^{\mathrm{exp}}_{\mathrm{G}} & e_2 \arrow[d, "*"] & \\
{} & K_1[f_1(v_1)] \arrow[l, squiggly] & & K_2[f_2(v_2)] \arrow[r, squiggly] & {} \\
{} \arrow[r, squiggly] & K_1[r_1] \arrow[d, "*"] & \sim^{\mathrm{exp}}_{\mathrm{G}} & K_2[r_2] \arrow[d, "*"] & {} \arrow[l, squiggly] \\
 & {} & & {} &
\end{tikzcd}
$$

Definition of local term equivalence $\sim_{G}^{exp}$

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

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

- Derive $\sim_{\mathrm{G}}^{\mathsf{exp}}$ from $\sim_{\mathrm{G}}$ by coinduction

$$
\begin{array}{ccccc}
e_1 & \sim_{\mathrm{G}}^{\mathsf{exp}} & e_2 & : & \tau \\
\downarrow_{\omega} & & \downarrow_{\omega} & &
\end{array}
$$

Case 1: Both diverge

Definition of local term equivalence $\sim_{\mathrm{G}}^{\mathsf{exp}}$

- Derive $\sim_{\mathrm{G}}^{\mathsf{exp}}$ from $\sim_{\mathrm{G}}$ by coinduction

$$\begin{array}{ccccc} e_1 & \sim_{\mathrm{G}}^{\mathsf{exp}} & e_2 & : & \tau \\ \downarrow^{*} & & \downarrow^{*} & & \\ v_1 & \overline{\sim_{\mathrm{G}}} & v_2 & : & \tau \end{array}$$

Case 2: Both terminate

Definition of local term equivalence $\sim_{G}^{\mathsf{exp}}$

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

$$
\begin{array}{ccccc}
e_1 & \sim_{G}^{\mathsf{exp}} & e_2 & & : \tau \\
\downarrow^{*} & & \downarrow^{*} & & \\
K_1[f_1(v_1)] & & K_2[f_2(v_2)] & & : \tau
\end{array}
$$

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

Case 3: Both call a function

Definition of local term equivalence $\sim_G^{\text{exp}}$

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

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

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

Case 3: Both call a function

- Derive $\sim_{\mathrm{G}}^{\mathsf{exp}}$ from $\sim_{\mathrm{G}}$ by coinduction

$$
\begin{array}{ccccc}
e_1 & \sim_{\mathrm{G}}^{\mathsf{exp}} & e_2 & : & \tau \\
\downarrow_* & & \downarrow_* & & \\
K_1[f_1(v_1)] & & K_2[f_2(v_2)] & : & \tau
\end{array}
$$

1. $f_1 \sim_{\mathrm{G}} f_2 : \tau_{\mathrm{arg}} \to \tau_{\mathrm{ret}}$
2. $\boxed{v_1 \ \overline{\sim_{\mathrm{G}}} \ v_2 : \tau_{\mathrm{arg}}}$
3. $\forall r_1 \ \overline{\sim_{\mathrm{G}}} \ r_2 : \tau_{\mathrm{ret}}.\ K_1[r_1] \sim_{\mathrm{G}}^{\mathsf{exp}} K_2[r_2] : \tau$

Case 3: Both call a function

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

$$
\begin{array}{ccccc}
e_1 & \sim_{G}^{\text{exp}} & e_2 & : & \tau \\
\downarrow^{*} & & \downarrow^{*} & & \\
K_1[f_1(v_1)] & & K_2[f_2(v_2)] & : & \tau
\end{array}
$$

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

Case 3: Both call a function

Definition of local term equivalence $\sim_G^{\text{exp}}$

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

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

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

Case 3: Both call a function

Definition of local term equivalence $\sim_{G}^{\mathrm{exp}}$

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

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

Simple illustrating example

$$
\begin{array}{rcll}
\tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\
\mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\
\mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int}
\end{array}
$$

Want to show

$$\mathrm{foo}_1 \approx \mathrm{foo}_2$$

Simple illustrating example

$$
\begin{array}{rcll}
\tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\
\mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\
\mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int}
\end{array}
$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$

Simple illustrating example

$$
\begin{array}{rcll}
\tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\
\mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\
\mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int}
\end{array}
$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. Show *consistent*$(\sim_{\mathrm{L}})$

Simple illustrating example

$$\begin{array}{rcll} \tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\ \mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\ \mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int} \end{array}$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.\ \forall v_1 \overline{\sim_{\mathrm{G}}}\, v_2 : \tau.$

$$666 + (\mathsf{unroll}\ v_1)(\mathrm{foo}_1) \sim_{\mathrm{G}}^{\mathsf{exp}} (\mathsf{unroll}\ v_2)(\mathrm{foo}_2) + 666$$

Simple illustrating example

$$
\begin{array}{rcll}
\tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\
\mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\
\mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int}
\end{array}
$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.\ \forall f_1 \overline{\sim_{\mathrm{G}}} f_2 : (\tau \to \mathsf{int}) \to \mathsf{int}.$

$$666 + (\mathsf{unroll}\,(\mathsf{roll}\ f_1))(\mathrm{foo}_1) \sim_{\mathrm{G}}^{\mathsf{exp}} (\mathsf{unroll}\,(\mathsf{roll}\ f_2))(\mathrm{foo}_2) + 666$$

Simple illustrating example

$$\begin{array}{rcll} \tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\ \mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\ \mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int} \end{array}$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}.\ \forall f_1 \overline{\sim_{\mathrm{G}}} f_2 : (\tau \to \mathsf{int}) \to \mathsf{int}.$

$$\begin{array}{ccc} 666 + (\mathsf{unroll}\,(\mathsf{roll}\ f_1))(\mathrm{foo}_1) & \sim_{\mathrm{G}}^{\mathsf{exp}} & (\mathsf{unroll}\,(\mathsf{roll}\ f_2))(\mathrm{foo}_2) + 666 \\ \downarrow^{*} & & \downarrow^{*} \\ 666 + f_1(\mathrm{foo}_1) & & f_2(\mathrm{foo}_2) + 666 \end{array}$$

Case 3: Both call a function

Simple illustrating example

$$\begin{array}{rcll} \tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\ \mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\ \mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int} \end{array}$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}.\ \forall f_1 \overline{\sim_{\mathrm{G}}} f_2 : (\tau \to \mathsf{int}) \to \mathsf{int}.$

$$\begin{array}{ccc} 666 + (\mathsf{unroll}\,(\mathsf{roll}\ f_1))(\mathrm{foo}_1) & \sim_{\mathrm{G}}^{\mathsf{exp}} & (\mathsf{unroll}\,(\mathsf{roll}\ f_2))(\mathrm{foo}_2) + 666 \\ \downarrow^{*} & & \downarrow^{*} \\ 666 + f_1(\mathrm{foo}_1) & \overline{\sim_{\mathrm{G}}} & f_2(\mathrm{foo}_2) + 666 \end{array}$$

Case 3: Both call a function

Simple illustrating example

$$
\begin{array}{rcll}
\tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\
\mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\
\mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int}
\end{array}
$$

1. Define $\sim_{\mathrm{L}}$ as $\{\ \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int}\ \}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}.\ \forall f_1 \overline{\sim_{\mathrm{G}}} f_2 : (\tau \to \mathsf{int}) \to \mathsf{int}.$

$$
\begin{array}{ccc}
666 + (\mathsf{unroll}\,(\mathsf{roll}\ f_1))(\mathrm{foo}_1) & \sim_{\mathrm{G}}^{\mathsf{exp}} & (\mathsf{unroll}\,(\mathsf{roll}\ f_2))(\mathrm{foo}_2) + 666 \\
\downarrow^{*} & & \downarrow^{*} \\
666 + f_1(\mathrm{foo}_1) & \overline{\sim_{\mathrm{G}}} & f_2(\mathrm{foo}_2) + 666
\end{array}
$$

Case 3: Both call a function

Simple illustrating example

$$\begin{array}{rcll} \tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\ \mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\ \mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int} \end{array}$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}\,.\ \forall f_1 \overline{\sim_{\mathrm{G}}} f_2 : (\tau \to \mathsf{int}) \to \mathsf{int}.$

$$666 + (\mathsf{unroll}\,(\mathsf{roll}\ f_1))(\mathrm{foo}_1) \sim_{\mathrm{G}}^{\mathsf{exp}} (\mathsf{unroll}\,(\mathsf{roll}\ f_2))(\mathrm{foo}_2) + 666$$

$$\downarrow^* \qquad\qquad\qquad\qquad \downarrow^*$$

$$666 + f_1(\mathrm{foo}_1) \qquad\qquad\qquad\qquad f_2(\mathrm{foo}_2) + 666$$

Case 3: Both call a function

Simple illustrating example

$$\begin{array}{rcll} \tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\ \mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\ \mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int} \end{array}$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}}.\ \forall r_1\ \overline{\sim_{\mathrm{G}}}\ r_2 : \mathsf{int}.$

$$666 + r_1 \sim_{\mathrm{G}}^{\mathsf{exp}} r_2 + 666$$

Simple illustrating example

$$
\begin{array}{rcll}
\tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\
\mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\
\mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int}
\end{array}
$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}} .\ \forall n \in [\![\mathsf{int}]\!].$

$$666 + n \sim_{\mathrm{G}}^{\mathsf{exp}} n + 666$$

Simple illustrating example

$$\begin{array}{rcll} \tau & := & \mu\alpha.\,(\alpha \to \mathsf{int}) \to \mathsf{int} & \\ \mathrm{foo}_1(f) & := & 666 + (\mathsf{unroll}\ f)(\mathrm{foo}_1) & : \tau \to \mathsf{int} \\ \mathrm{foo}_2(f) & := & (\mathsf{unroll}\ f)(\mathrm{foo}_2) + 666 & : \tau \to \mathsf{int} \end{array}$$

1. Define $\sim_{\mathrm{L}}$ as $\{\, \mathrm{foo}_1 \sim_{\mathrm{L}} \mathrm{foo}_2 : \tau \to \mathsf{int} \,\}$
2. $\forall \sim_{\mathrm{G}} \supseteq \sim_{\mathrm{L}} .\ \forall n \in [\![\mathsf{int}]\!]$.

$$\begin{array}{ccc} 666 + n & \sim_{\mathrm{G}}^{\mathsf{exp}} & n + 666 \\ \downarrow^{*} & & \downarrow^{*} \\ \underline{n + 666} & \overline{\sim_{\mathrm{G}}} & \underline{n + 666} \end{array}$$

Case 2: Both terminates

Summary: The benefits of "proof parametricity"

1. Horizontal compositionality (aka congruence)
   - The less proofs about different modules assume about $\sim_{\mathrm{G}}$, 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

“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 $\sim_{\mathrm{L}}$ over $\sim_{\mathrm{G}}$
- 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