A Relational Logic for Higher-Order Programs

Alejandro Aguirre¹  Gilles Barthe¹  Marco Gaboardi²  Deepak Garg³
Pierre-Yves Strub⁴

ICFP, Sep 5 2017

¹Imdea Software; ²University at Buffalo, SUNY; ³MPI-SWS; ⁴École Polytechnique
Relational properties a.k.a. 2-properties (I)

Two runs of two programs, e.g. equivalence...
Relational properties a.k.a. 2-properties (II)

...or two runs of the same program, e.g. non-interference

X₁ and X₂ are low-equivalent

Y₁ and Y₂ are low-equivalent
Relational refinement types (I)

Refinement types extend types with logical properties:

\[ \Gamma \vdash t : \{ x : \mathbb{N} \mid \exists z. x = 2 \times z \} \]

Relational refinement types \(^1\) generalize them to a relational setting:

\[ \Gamma \vdash t_1 \sim t_2 : \{ x : \mathbb{N} \mid x_1 = x_2 \} \]

\(^1\)Gilles Barthe, Cédric Fournet, Benjamin Grégoire, Pierre-Yves Strub, Nikhil Swamy, and Santiago Zanella Béguelin. *Probabilistic relational verification for cryptographic implementations* (POPL '14)
Relational refinement types (II)

Pros:

- Very intuitive (e.g. \( \{x : \mathbb{N} \mid x_1 \leq x_2\} \rightarrow \{y : \mathbb{N} \mid y_1 \leq y_2\} \) types monotonic functions)

- Syntax directed

- Exploit structural similarities

  \[
  \text{if } b \text{ then } 2 \times x \text{ else } x + 1 \sim \text{if } b \text{ then } 2 \times x \text{ else } x - 1
  \]

- Lots of theoretical and practical developments for unary refinements could be reused
We want to prove:

\[ \text{take } n \ (\text{map } f \ l) = \text{map } f \ (\text{take } n \ l) \]
Limits of RRT

We want to prove:

\[
\text{take } n \ (\text{map } f \ l) = \text{map } f \ (\text{take } n \ l)
\]

Which type to give it?

\[
\text{take } n \ (\text{map } f \ l) \sim \text{map } f \ (\text{take } n \ l) : \{ x : \text{list}_\mathbb{N} \mid x_1 = x_2 \}\]
Limits of RRT

We want to prove:

\[ \text{take } n \ (\text{map } f \ l) = \text{map } f \ (\text{take } n \ l) \]

Which type to give it?

\[ \text{take } n \ (\text{map } f \ l) \sim \text{map } f \ (\text{take } n \ l) : \{x : \text{list}_\mathbb{N} | x_1 = x_2\} \]

We apply [APP] rule...

\[ \text{take } \sim \text{map} : \{x : ? | ?\} \]
Our contributions

- A foundational system to prove relational properties
- in a syntax directed way
- not restricted by types or structure
Base logic: HOL

- $\lambda$-terms over simple $+$ inductive types
- (Axiomatically defined) Predicates: $P(t_1, \ldots, t_n)$

$$\forall l. \text{prefix}([], l) \quad \forall xtl. \text{prefix}(t, l) \Rightarrow \text{prefix}(x :: t, x :: l)$$

- Propositional connectives: $\land, \lor, \Rightarrow$
- Quantification over simple/inductive types: $\forall(x : \tau), \exists(x : \tau)$
Base logic: HOL

- $\lambda$-terms over simple + inductive types
- (Axiomatically defined) Predicates: $P(t_1, \ldots, t_n)$

\[
\forall l. prefix([], l) \quad \forall xtl. prefix(t, l) \Rightarrow prefix(x :: t, x :: l)
\]

- Propositional connectives: $\land, \lor, \Rightarrow$
- Quantification over simple/inductive types: $\forall(x : \tau), \exists(x : \tau)$

Why not just use this?
Base logic: HOL

- $\lambda$-terms over simple + inductive types
- (Axiomatically defined) Predicates: $P(t_1, \ldots, t_n)$

$$\forall l. \text{prefix}([], l) \quad \forall xtl. \text{prefix}(t, l) \Rightarrow \text{prefix}(x :: t, x :: l)$$

- Propositional connectives: $\land, \lor, \Rightarrow$
- Quantification over simple/inductive types: $\forall (x : \tau), \exists (x : \tau)$

Why not just use this?

No syntax directedness or structural reasoning
Judgements combine **types** and **logic**:

**HOL:** \( \Gamma \mid \Psi \vdash \phi \)

**UHOL:** \( \Gamma \mid \Psi \vdash t_1 : \tau_1 \mid \phi(r) \)

**RHOL:** \( \Gamma \mid \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \mid \phi(r_1, r_2) \)
Judgements combine **types** and **logic**:

**HOL:** \[ \Gamma \vdash \psi \vdash \phi \]

**UHOL:** \[ \Gamma \vdash \psi \vdash t_1 : \tau_1 \vdash \phi(r) \]

**RHOL:** \[ \Gamma \vdash \psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \vdash \phi(r_1, r_2) \]
Relational HOL

Judgements combine **types** and **logic**:

HOL: \[ \Gamma \vdash \Psi \vdash \phi \]

 Assertions over \( \Gamma \)

 Context

UHOL: \[ \Gamma \vdash \Psi \vdash t_1 : \tau_1 \mid \phi(r) \]

RHOL: \[ \Gamma \vdash \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \mid \phi(r_1, r_2) \]
Judgements combine **types** and **logic**:

**HOL:** \[ \Gamma \vdash \Psi \vdash \phi \]  

**Context:** Assertions over \( \Gamma \)  

**Predicate:**

**UHOL:** \[ \Gamma \vdash \Psi \vdash t_1 : \tau_1 \vdash \phi(r) \]  

**RHOL:** \[ \Gamma \vdash \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \vdash \phi(r_1, r_2) \]
Relational HOL

Judgements combine types and logic:

\[
\text{HOL: } \Gamma \mid \Psi \vdash \phi
\]

\[
\text{UHOL: } \Gamma \mid \Psi \vdash t_1 : \tau_1 \mid \phi(r)
\]

\[
\text{RHOL: } \Gamma \mid \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \mid \phi(r_1, r_2)
\]
Judgements combine **types** and **logic**:

**HOL:** \( \Gamma \mid \Psi \vdash \phi \)

**UHOL:** \( \Gamma \mid \Psi \vdash t_1 : \tau_1 \mid \phi(r) \)

**RHOL:** \( \Gamma \mid \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \mid \phi(r_1, r_2) \)
Relational HOL

Judgements combine types and logic:

**HOL:** \( \Gamma \vdash \Psi \vdash \phi \)

**UHOL:** \( \Gamma \vdash \Psi \vdash t_1 : \tau_1 \vdash \phi(r) \)

**RHOL:** \( \Gamma \vdash \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \vdash \phi(r_1, r_2) \)

1st term
Judgements combine **types** and **logic**:

**HOL**: \[ \Gamma \vdash \psi \vdash \phi \]

**UHOL**: \[ \Gamma \vdash \psi \vdash t_1 : \tau_1 \vdash \phi(r) \]

**RHOL**: \[ \Gamma \vdash \psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \vdash \phi(r_1, r_2) \]

1st term 2nd term
Judgements combine types and logic:

**HOL:** \( \Gamma \mid \Psi \vdash \phi \)

**UHOL:** \( \Gamma \mid \Psi \vdash t_1 : \tau_1 \mid \phi(r) \)

**RHOL:** \( \Gamma \mid \Psi \vdash t_1 : \tau_1 \sim t_2 : \tau_2 \mid \phi(r_1, r_2) \)

1st term

2nd term

Predicate
Key Idea: separation of concerns between types and assertions

- Unary

\[\vdash t : \{x : \tau \mid \phi(x)\} \rightarrow t : \tau \mid \phi(r)\]

- Relational

\[t_1 \sim t_2 : \{x : \tau \mid \phi(x_1, x_2)\} \rightarrow t_1 : \tau \sim t_2 : \tau \mid \phi(r_1, r_2)\]
**Two-sided and one-sided rules**

**Two-sided rules** relate two terms with the same top term former

\[ \lambda x_1.t_1 \sim \lambda x_2.t_2 \]

**One-sided rules** relate two terms with *different* top term former

\[ \lambda x_1.t_1 \sim t_2 \ u_2 \]
Judgements combine **types** and **logic**:

**Abstraction**

\[
\begin{align*}
\Gamma, x_1 : \tau_1, x_2 : \tau_2 | \psi, \phi' & \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 | \phi \\
\Gamma | \psi & \vdash \lambda x_1.t_1 : \tau_1 \rightarrow \sigma_1 \sim \lambda x_2.t_2 : \tau_2 \rightarrow \sigma_2 | \forall x_1, x_2.\phi' \Rightarrow \phi[r_1 x_1/r_1][r_2 x_2/r_2]
\end{align*}
\]
Judgements combine **types** and **logic**:

Abstraction

\[
\Gamma, x_1 : \tau_1, x_2 : \tau_2 \mid \psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi
\]

\[
\Gamma \mid \psi \vdash \lambda x_1. t_1 : \tau_1 \rightarrow \sigma_1 \sim \lambda x_2. t_2 : \tau_2 \rightarrow \sigma_2 \mid \forall x_1, x_2. \phi' \Rightarrow \phi[r_1 x_1/r_1][r_2 x_2/r_2]
\]
Judgements combine types and logic:

Abstraction

\[
\Gamma, x_1 : \tau_1, x_2 : \tau_2 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi
\]

\[
\Gamma \mid \Psi \vdash \lambda x_1.t_1 : \tau_1 \rightarrow \sigma_1 \sim \lambda x_2.t_2 : \tau_2 \rightarrow \sigma_2 \mid \forall x_1, x_2.\phi' \Rightarrow \phi[r_1 x_1/r_1][r_2 x_2/r_2]
\]
Judgements combine **types** and **logic**:

**Abstraction**

\[
\frac{\Gamma, x_1 : \tau_1, x_2 : \tau_2 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi}{\Gamma \mid \Psi \vdash \lambda x_1. t_1 : \tau_1 \rightarrow \sigma_1 \sim \lambda x_2. t_2 : \tau_2 \rightarrow \sigma_2 \mid \forall x_1, x_2. \phi' \Rightarrow \phi[r_1 x_1/r_1][r_2 x_2/r_2]}
\]
Two-sided rules

Judgements combine *types* and *logic*:

**Abstraction**

\[ \begin{array}{c}
\Gamma, x_1 : \tau_1, x_2 : \tau_2 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi \\
\Gamma \mid \Psi \vdash \lambda x_1.t_1 : \tau_1 \rightarrow \sigma_1 \sim \lambda x_2.t_2 : \tau_2 \rightarrow \sigma_2 \mid \forall x_1, x_2.\phi' \Rightarrow \phi[r_1 x_1/r_1][r_2 x_2/r_2]
\end{array} \]

**Application**

\[ \begin{array}{c}
\Gamma \mid \Psi \vdash t_1 : \tau_1 \rightarrow \sigma_1 \sim t_2 : \tau_2 \rightarrow \sigma_2 \mid \forall x_1, x_2.\phi'[x_1/r_1][x_2/r_2] \Rightarrow \phi[r_1 x_1/r_1][r_2 x_2/r_2] \\
\Gamma \mid \Psi \vdash u_1 : \tau_1 \sim u_2 : \tau_2 \mid \phi' \\
\Gamma \mid \Psi \vdash t_1 u_1 : \sigma_1 \sim t_2 u_2 : \sigma_2 \mid \phi[u_1/x_1][u_2/x_2]
\end{array} \]
One-sided rules

Abstraction

\[
\frac{\Gamma, x_1 : \tau_1 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi}{\Gamma \mid \Psi \vdash \lambda x_1. t_1 : \tau \rightarrow \sigma_1 \sim t_2 : \sigma_2 \mid \forall x_1. \phi' \Rightarrow \phi[r_1 x_1/r_1]}
\]

Application

\[
\frac{\Gamma \mid \Psi \vdash t_1 : \tau_1 \rightarrow \sigma_1 \sim u_2 : \sigma_2 \mid \forall x_1. \phi'[x_1/r_1] \Rightarrow \phi[r_1 x_1/r_1]}{
\Gamma \mid \Psi \vdash u_1 : \sigma_1 \mid \phi'}
\]

\[
\frac{\Gamma \mid \Psi \vdash t_1 u_1 : \sigma_1 \sim u_2 : \sigma_2 \mid \phi[u_1/x_1]}{\Gamma \mid \Psi \vdash u_1 : \sigma_1 \mid \phi'}
\]
One-sided rules

Abstraction

\[
\Gamma, x_1 : \tau_1 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi \\
\Gamma \mid \Psi \vdash \lambda x_1. t_1 : \tau_1 \rightarrow \sigma_1 \sim t_2 : \sigma_2 \mid \forall x_1. \phi' \Rightarrow \phi[r_1 x_1/r_1]
\]

Application

\[
\Gamma \mid \Psi \vdash t_1 : \tau_1 \rightarrow \sigma_1 \sim u_2 : \sigma_2 \mid \forall x_1. \phi'[x_1/r_1] \Rightarrow \phi[r_1 x_1/r_1] \\
\Gamma \mid \Psi \vdash u_1 : \sigma_1 \mid \phi' \\
\hline
\Gamma \mid \Psi \vdash t_1 u_1 : \sigma_1 \sim u_2 : \sigma_2 \mid \phi[u_1/x_1]
\]
One-sided rules

Abstraction

\[
\frac{\Gamma, x_1 : \tau_1 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi}{\Gamma \mid \Psi \vdash \lambda x_1.t_1 : \tau_1 \rightarrow \sigma_1 \sim t_2 : \sigma_2 \mid \forall x_1.\phi' \Rightarrow \phi[r_1 x_1/r_1]}
\]

Application

\[
\frac{\Gamma \mid \Psi \vdash t_1 : \tau_1 \rightarrow \sigma_1 \sim u_2 : \sigma_2 \mid \forall x_1.\phi'[x_1/r_1] \Rightarrow \phi[r_1 x_1/r_1]}{
\Gamma \mid \Psi \vdash u_1 : \sigma_1 \mid \phi'}
\]

\[
\frac{\Gamma \mid \Psi \vdash t_1 u_1 : \sigma_1 \sim u_2 : \sigma_2 \mid \phi[u_1/x_1]}{
\Gamma \mid \Psi \vdash t_1 u_1 : \sigma_1 \sim u_2 : \sigma_2 \mid \phi[u_1/x_1]}
\]
One-sided rules

Abstraction

\[
\begin{align*}
\Gamma, x_1 : \tau_1 \mid \Psi, \phi' & \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi \\
\Gamma \mid \Psi & \vdash \lambda x_1. t_1 : \tau_1 \rightarrow \sigma_1 \sim t_2 : \sigma_2 \mid \forall x_1. \phi' \Rightarrow \phi[r_1 \ x_1 / r_1]
\end{align*}
\]

Application

\[
\begin{align*}
\Gamma \mid \Psi & \vdash t_1 : \tau_1 \rightarrow \sigma_1 \sim u_2 : \sigma_2 \mid \forall x_1. \phi'[x_1 / r_1] \Rightarrow \phi[r_1 \ x_1 / r_1] \\
\Gamma \mid \Psi & \vdash u_1 : \sigma_1 \mid \phi' \\
\Gamma \mid \Psi & \vdash t_1 u_1 : \sigma_1 \sim u_2 : \sigma_2 \mid \phi[u_1 / x_1]
\end{align*}
\]
The SUB rule

Allows us to fall back to HOL:

\[ \Gamma \mid \psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi' \quad \Gamma \mid \psi \vdash_{\text{HOL}} \phi'[t_1/r_1][t_2/r_2] \Rightarrow \phi[t_1/r_1][t_2/r_2] \]

\[ \Gamma \mid \psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi \]
Equivalence between HOL and RHOL

\[ \Gamma \mid \Psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi \]

\[ \iff \]

\[ \Gamma \mid \Psi \vdash \phi[t_1/r_1][t_2/r_2] \]

Plus: subject reduction, soundness, ...
Embeddings

RHOL is also useful as a framework in which to embed other relational typing systems:

- Relational Refinement Types
- DCC (dependency)
- RelCost (relational cost)

We get “for free” proofs of soundness.

Since RHOL is more expressive, we can verify new examples.

\[
|s| = |t|, \text{sorted}(s) \vdash \text{isort } s \sim \text{isort } t \mid \text{cost } r_1 \leq \text{cost } r_2
\]
Conclusions

- Relational refinement types have limited one-sided reasoning
- HOL is expressive but does not exploit structural similarities
- RHOL combines the best of both worlds in a lossless way: expressiveness + two-sided reasoning + 1 sided-reasoning
- This makes RHOL a foundational system
- Future work: extension with effects & implementation
Conclusions

- Relational refinement types have limited one-sided reasoning
- HOL is expressive but does not exploit structural similarities
- RHOL combines the best of both worlds in a lossless way: expressiveness + two-sided reasoning + 1 sided-reasoning
- This makes RHOL a foundational system
- Future work: extension with effects & implementation

Thanks!
Questions?
Embedding refinement types

We can embed refinement types into our system:

\[
\| \{ y : \tau \mid \phi \} \|(x_1, x_2) \triangleq \bigwedge_{i \in \{1,2\}} [\tau](x_i) \land \phi[x_i/y]
\]

\[
\| \{ y :: T \mid \phi \} \|(x_1, x_2) \triangleq \| T \|(x_1, x_2) \land \phi[x_1/y_1][x_2/y_2]
\]

\[
\| \Pi(y : \tau). \sigma \|(x) \triangleq \bigwedge_{i \in \{1,2\}} \forall y. [\tau](y) \Rightarrow [\sigma](xy)
\]

\[
\| \Pi(y :: T). U \|(x_1, x_2) \triangleq \forall y_1 y_2. \| T \|(y_1, y_2) \Rightarrow \| \sigma \|(x_1 y_1, x_2 y_2)
\]
We can implement factorial without and with accumulator:

\[
\text{fact}_1 \equiv \text{letrec } f_1 \ x_1 = \text{case } x_1 \text{ of } [0 \rightarrow 1; \text{Sy}_1 \rightarrow (\text{Sy}_1) \ast (f_1 y_1)]
\]

\[
\text{fact}_2 \equiv \text{letrec } f_2 \ x_2 = \lambda a.\text{case } x_2 \text{ of } [0 \rightarrow a; \text{Sy}_2 \rightarrow f_2 \ y_2 \ (a \ast (\text{Sy}_2))]
\]

We want to prove:

\[\emptyset \mid \emptyset \vdash \text{fact}_1 \sim \text{fact}_2 \mid \forall x_1 x_2 a. \ x_1 = x_2 \Rightarrow (r_1 \ x_1) \ast a = r_2 \ x_2 \ a\]

Notice that the two programs have different types: \(\mathbb{N} \rightarrow \mathbb{N}\) and \(\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}\)
Example: factorial (II)

Proof reduces to:

\[
\Gamma \vdash \psi \vdash \text{case } x_1 \text{ of } [0 \rightarrow 1; S y_1 \rightarrow (S y_1) \ast (f_1 y_1)] \sim \\
\text{case } x_2 \text{ of } [0 \rightarrow a; S y_2 \rightarrow f_2 y_2 (a \ast (S y_2))] | r_1 \ast a = r_2
\]

where \( \psi \) is the “inductive hypothesis”
Example: factorial (II)

Proof reduces to:

\[
\Gamma \mid \psi \vdash \text{case } x_1 \text{ of } [0 \rightarrow 1; \text{Sy}_1 \rightarrow (\text{Sy}_1) \ast (f_1 y_1)] \sim \\
\text{case } x_2 \text{ of } [0 \rightarrow a; \text{Sy}_2 \rightarrow f_2 y_2 (a \ast (\text{Sy}_2))] \mid r_1 \ast a = r_2
\]

where \(\psi\) is the “inductive hypothesis”

Proof obligations:

- \(\Gamma \mid \psi \vdash 1 \sim a \mid r_1 \ast a = r_2\)
Proof reduces to:

\[
\Gamma \vdash \psi \mid \text{case } x_1 \text{ of } [0 \rightarrow 1; Sy_1 \rightarrow (Sy_1) \ast (f_1y_1)] \sim \\
\text{case } x_2 \text{ of } [0 \rightarrow a; Sy_2 \rightarrow f_2 y_2 (a \ast (Sy_2))] \mid r_1 \ast a = r_2
\]

where \( \psi \) is the “inductive hypothesis”

Proof obligations:

- \( \Gamma \mid \psi \vdash 1 \sim a \mid r_1 \ast a = r_2 \)

Trivial
Example: factorial (II)

Proof reduces to:

\[
\begin{align*}
\Gamma \mid \psi \vdash & \text{case } x_1 \text{ of } [0 \to 1; S y_1 \to (S y_1) \ast (f_1 y_1)] \sim \\
& \text{case } x_2 \text{ of } [0 \to a; S y_2 \to f_2 y_2 (a \ast (S y_2))] \mid r_1 \ast a = r_2
\end{align*}
\]

where $\psi$ is the “inductive hypothesis”

Proof obligations:

- $\Gamma \mid \psi \vdash 1 \sim a \mid r_1 \ast a = r_2$
- $\Gamma \mid \psi, x_1 = S y_2, x_2 = S y_2 \vdash (S y_1) \ast (f_1 y_1) \sim f_2 y_2 (a \ast (S y_2)) \mid (r_1 y_1) \ast a = (r_2 y_2)$
Example: factorial (II)

Proof reduces to:

\[ \Gamma | \psi \vdash \text{case } x_1 \text{ of } [0 \rightarrow 1; Sy_1 \rightarrow (Sy_1) \ast (f_1 y_1)] \sim \]
\[ \text{case } x_2 \text{ of } [0 \rightarrow a; Sy_2 \rightarrow f_2 y_2 (a \ast (Sy_2))] | r_1 \ast a = r_2 \]

where \( \psi \) is the “inductive hypothesis”

Proof obligations:

- \( \Gamma | \psi \vdash 1 \sim a | r_1 \ast a = r_2 \)
- \( \Gamma | \psi, x_1 = Sy_2, x_2 = Sy_2 \vdash (Sy_1) \ast (f_1 y_1) \sim f_2 y_2 (a \ast (Sy_2)) | (r_1 y_1) \ast a = (r_2 y_2) \)

By instantiating \( \psi \)