Structural Logical Relations with Case Analysis and Equality Reasoning

Ulrik Rasmussen  Andrzej Filinski

Department of Computer Science
University of Copenhagen

LFMTP, Boston, MA
September 23, 2013
Motivation

• Logical relations (LR) are a powerful proof technique, but difficult to formalize in Twelf and similar systems.

• Method to do so (structural logical relations) devised by [Schürmann and Sarnat, 2008]: Formalizes weak normalization and completeness of equivalence checking for simply typed \( \lambda \)-calculus.

• Minimal, pure \( \lambda \)-calculus.

• Can we use this for “real” programming languages?
Our Contributions

- Extension of structural logical relations allowing more proofs to be formalized.
- Further insight into the structure of logical-relations based proofs.
- Demonstration of proofs of observational equivalence.

In this talk: High-level perspective; see paper for technical details.
Example 1: Termination

**Definition ($\lambda^{\text{nat}}$)**

- **Naturals**
  \[ n :: \text{Nat} ::= z \mid s \, n \]

- **Expressions**
  \[ e, v :: \text{Exp} ::= x \mid \text{lam} \, x. \, e_0 \mid \text{app} \, e_1 \, e_2 \mid \text{num} \, n \]

- **Types**
  \[ \tau :: \text{Tp} ::= \text{nat} \mid \text{arr} \, \tau_2 \, \tau_0 \]

- **CBN Eval.**
  \[ \varepsilon :: e \Downarrow v \]

- **Typing**
  \[ \mathcal{T} :: x_1 : \tau_1, \ldots, x_n : \tau_n \triangleright e : \tau \]

**Theorem (Termination)**

For any $e$ where $\triangleright e : \text{nat}$, there exists a $v$ such that $e \Downarrow v$. 
Example 1: Logical Relation

- Termination proof requires a logical relation:

**Definition (Logical Relation for Termination)**

\[
\begin{align*}
e \in \llbracket \text{nat} \rrbracket & \iff \exists n. e \downarrow \text{num } n \\
e \in \llbracket \text{arr } \tau_2 \tau_0 \rrbracket & \iff \forall e_2. e_2 \in \llbracket \tau_2 \rrbracket \supset \text{app } e \ e_2 \in \llbracket \tau_0 \rrbracket
\end{align*}
\]

- Extend to open expressions: For \( \Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n \):

\[
\Gamma \vdash e \in \llbracket \tau \rrbracket \iff \forall e_1 \in \llbracket \tau_1 \rrbracket \cdots e_n \in \llbracket \tau_n \rrbracket.
\]

\[
e[e_1 \cdots e_n/x_1 \cdots x_n] \in \llbracket \tau \rrbracket
\]

- Fundamental Theorem: If \( \Gamma \triangleright e : \tau \) then \( \Gamma \vdash e \in \llbracket \tau \rrbracket \).

- Representing LR at arrow types problematic. Twelf only supports metatheorems on \( \forall \exists \)-form.
Use an Assertion Logic

Structural Logical Relations [Schürmann and Sarnat, 2008]:

\[
\begin{align*}
\text{Definition (Assertion Logic, } \implies_{\text{eval}}) \\
\text{Propositions: } & \quad A, B :: \text{Form} ::= \forall^{\text{Exp}} \alpha. A \mid \exists^{\text{Nat}} \alpha. A \\
& \quad \quad \mid A \supset B \\
& \quad \quad \mid \text{eval}(e, v) \\
\text{Assumptions: } & \quad \Delta :: \text{Assm} ::= \{A_1, \ldots, A_n\} \text{ (Unordered)} \\
\text{Parameters: } & \quad \Xi :: \text{Ctx} ::= \cdot \mid \Xi, \alpha : \text{Nat} \mid \Xi, \alpha : \text{Exp} \\
\text{“Cut-full” sequent: } & \quad \Xi \mid \Delta \implies \bullet \Rightarrow A \\
\text{“Cut-free” sequent: } & \quad \Xi \mid \Delta \implies^{\circ} A \\
\end{align*}
\]

\( \implies_{\text{eval}} \) axiomatizes \(_, _\) \( \Downarrow \): 

Theorem (Extraction)

\[ \text{If } \cdot \mid \emptyset \implies^{\circ} \text{eval}(e, v), \text{ then } e \Downarrow v. \]
Fundamental Theorem

- LR representation: Map types to propositions w/ bound expression:
  \[ \llbracket \tau \rrbracket \colon \text{Exp} \rightarrow \text{Form} \]

### Definition (Logical Relation for Termination, Assertion-Level)

- \([\text{nat}] (e) \iff \exists \text{Nat } n. \text{eval} (e, \text{num } n)\]
- \([\text{arr } \tau_2 \tau_0] (e) \iff \forall \text{Exp } e_2. [\tau_2] (e_2) \supset [\tau_0] (\text{app } e e_2)\]

### Theorem (Fundamental Theorem)

For any e, if

\[ x_1 : \tau_1, \ldots, x_n : \tau_n \triangleright e : \tau, \]

then

\[ x_1 : \text{Exp}, \ldots, x_n : \text{Exp} \mid [\tau_1] (x_1), \ldots, [\tau_n] (x_n) \Rightarrow [\tau] (e). \]

- Note: Induction lives entirely on the meta-level!
Corollary: $\triangleright e : \text{nat}$ implies $\bullet | \emptyset \dashv \exists v. \text{eval}(e, v)$.

By extraction, termination reduced to proving cut elimination:

**Theorem (Cut Elimination)**

If $\Xi | \Delta \rightarrow A$, then $\Xi | \Delta \circ \rightarrow A$

In Twelf: Syntactic proof due to [Pfenning, 2000]. Bulk of work in:

**Lemma (Cut Admissibility)**

If $\Xi | \Delta \rightarrow A$ and $\Xi | \Delta, A \rightarrow C$ then $\Xi | \Delta \rightarrow C$. 
Languages just slightly more expressive than simply typed $\lambda$-calculus require stronger assertion logic.

Specifically, equality reasoning and case-analysis principles.

Assertion logic can only be strengthened if it retains cut-admissibility.
Example 2: $\lambda$-calculus + ifz

Definition ($\lambda^{\text{nat,ifz}}$)

Naturals \hspace{1cm} n ::= \text{Nat} ::= z \mid s\ n

Expressions \hspace{1cm} e ::= \text{Exp} ::= x \mid \text{lam} \ x. \ e_0 \mid \text{app} \ e_1 \ e_2 \mid \text{num} \ n

\hspace{1cm} | \text{ifz}(e_0, e_1, e_2)

Types \hspace{1cm} \tau ::= \text{Tp} ::= \text{nat} \mid \text{arr} \ \tau_2 \ \tau_0

CBN Eval. \hspace{1cm} \varepsilon ::= e \Downarrow v

Typing \hspace{1cm} \tau ::= \Gamma \triangleright e : \tau

Fund thm.: By IH, get $\downarrow[nat](e_0) \equiv \exists^\text{Nat} n. \ \text{eval}(e_0, \ \text{num} \ n)$. Select one of branches $e_1$ or $e_2$ based on $n$.

- Structure of terms opaque to assertion logic.
- Specify structure explicitly in LR.
Example 2: Logical Relation, Assertion Logic

Definition (Assertion Logic \((\implies eval, eq))\)

**Propositions:**

\[ A, B \quad \text{Form} \quad ::= \quad \forall \alpha. A \mid \exists \alpha. A \]
\[ \mid A \supset B \mid A \land B \mid A \lor B \mid \text{eval}(e, v) \mid \text{eq}(n, n') \]

**Assumptions:**

\[ \Delta \quad \text{Assm} \quad ::= \quad \{A_1, \ldots, A_n\} \quad \text{(Unordered)} \]

**Parameters:**

\[ \Xi \quad \text{Ctx} \quad ::= \quad \cdot \mid \Xi, \alpha : \text{Nat} \mid \Xi, \alpha : \text{Exp} \]

**Proof sequent:**

\[ \Xi | \Delta \quad \overset{c}{\longrightarrow} \quad A \quad (c \in \{\bullet, \circ\}) \]

Definition (Logical Relation for Termination, Assertion-Level)

\[
\begin{align*}
\left[\text{nat}\right](e) & \iff \exists \text{Nat } n. \text{eval}(e, \text{num } n) \\
\left[\text{arr } \tau_2 \tau_0\right](e) & \iff \forall \text{Exp } e_2. \left[\tau_2\right](e_2) \supset \left[\tau_0\right](\text{app } e e_2)
\end{align*}
\]
Equality

- \( \text{eq}(n, n') \) axiomatizes syntactic equality:

\[
\Xi \mid \Delta \xrightarrow{c} \text{eq}(n, n)
\]

- Cannot show cut-elim for logic w/general equality conversion.

- Must restrict equality reasoning to leaves of proofs, i.e., atomic formulas:

\[
\Xi \mid \Delta \xrightarrow{c} \text{eval}(e[n_1/x_1], v[n_2/x_2])
\]
### Example 3: $\lambda$-calculus + case

#### Definition ($\lambda^{\text{nat,case}}$)

<table>
<thead>
<tr>
<th>Natural</th>
<th>Exp</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n :: \text{Nat} := z \mid s \ n$</td>
<td>$e :: \text{Exp} ::= x \mid \text{lam } x. \ e_0 \mid \text{app } e_1 \ e_2 \mid \text{num } n$</td>
<td>$\text{case}(e_0, e_1, x. \ e_2)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Types</th>
<th>CBN Eval.</th>
<th>Typing</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau :: \text{Tp} ::= \text{nat} \mid \text{arr } \tau_2 \ \tau_0$</td>
<td>$\mathcal{E} ::= e \Downarrow v$</td>
<td>$\Gamma \triangleright e : \tau$</td>
</tr>
</tbody>
</table>

- Still need to select branch based on $[\text{nat}](e_0)$.
- In subcase where $\Delta \models \Xi \Rightarrow \mathsf{eval}(e_0, \text{num } (s \ n'))$: By IH, get $\Xi, x : \text{Exp} \models \Delta, [\text{nat}](x) \Rightarrow [\tau](e_2)$. *Instantiate* LR for $e_2[\text{num } n'/x]$:
  Need to show $\Downarrow \Rightarrow [\text{nat}](\text{num } n')$. 

---
Example 3: Logical Relation, Assertion Logic

Definition (Assertion Logic \(\equiv \text{eval,eq}\))

Propositions: \(A, B \quad \text{Form} \quad ::= \quad \forall^\text{Exp} \alpha. A \mid \exists^\text{Nat} \alpha. A\)
\[\mid A \supset B \mid A \land B \mid A \lor B\]
\[\mid \text{eval}(e, v) \mid \text{eq}(n, n')\]

Assumptions: \(\Delta \quad ::\quad \text{Assm} \quad ::= \quad \{A_1, \ldots, A_n\} \quad \text{(Unordered)}\)

Parameters: \(\Xi \quad ::\quad \text{Ctx} \quad ::= \quad \cdot \mid \Xi, \alpha : \text{Nat} \mid \Xi, \alpha : \text{Exp}\)

Proof sequent: \(\Xi|\Delta \xrightarrow[c]{\cdot} A \quad (c \in \{\bullet, \circ\})\)

Definition (Logical Relation for Termination, Assertion-Level)

\([\text{nat}] e \quad \iff \quad \exists^\text{Nat} n. \text{eval}(e, \text{num} n)\)
\[\land (\text{eq}(n, z) \lor (\exists^\text{Nat} n'. \text{eq}(n, s n'))\]
\[\land (\text{eq}(n', z) \lor (\exists^\text{Nat} n''. \cdots )))\]

\([\text{arr} \tau_2 \tau_0] e \quad \iff \quad \forall^\text{Exp} e_2. [\tau_2](e_2) \supset [\tau_0](\text{app} e e_2)\)
Example 3: Logical Relation, Assertion Logic

Definition (Assertion Logic ($\iff$ \text{eval,eq,nat}^+))

Propositions: \( A, B \) :: Form ::= \( \forall^{\text{Exp}} \alpha. A \mid \exists^{\text{Nat}} \alpha. A \)
\[ \mid A \supset B \mid A \land B \mid A \lor B \]
\[ \mid \text{eval}(e, v) \mid \text{eq}(n, n') \mid \text{nat}^+(n) \]

Assumptions: \( \Delta \) :: Assm ::= \{\( A_1, \ldots, A_n \)\} (Unordered)

Parameters: \( \Xi \) :: Ctx ::= \( \cdot \mid \Xi, \alpha : \text{Nat} \mid \Xi, \alpha : \text{Exp} \)

Proof sequent: \( \Xi \mid \Delta \xRightarrow{c} A \) (\( c \in \{\bullet, \circ\} \))

Definition (Logical Relation for Termination, Assertion-Level)

\( [\text{nat}](e) \iff \exists^{\text{Nat}} n. \text{eval}(e, \text{num} n) \land \text{nat}^+(n) \)

\( [\text{arr} \ \tau_2 \ \tau_0](e) \iff \forall^{\text{Exp}} e_2. [\tau_2](e_2) \supset [\tau_0](\text{app} e e_2) \)
\[ \Xi \vdash \Delta \Rightarrow \text{nat}^+(z) \]

\[ \Xi \vdash \Delta \Rightarrow \text{nat}^+(s\ n) \]

\[ \Xi \vdash \Delta, \text{eq}(n, z) \Rightarrow C \]

\[ \Xi, n' : \text{Nat}, \text{eq}(n, s\ n'), \text{nat}^+(n') \Rightarrow C \]

\[ \Xi \vdash \Delta, \text{nat}^+(n) \Rightarrow C \]

- \( \text{nat}^+(n) \) proof: *structural witness* for some \( n \).

- As-is, Pfenning’s cut-admissibility proof does not work for logic with left-rules on atomic propositions.

- Can be made to work as long as an index term always gets smaller in subderivations. For \( \text{nat}^+(n) \): \( n \) gets smaller.
Case-Analysis on Derivations?

- Required in, e.g., proofs of observational equivalence (see paper).

- Observation: For $\text{eval}(e, v)$, indices do not get smaller in sub-proofs. To be able to add left-rule, index by explicit metric, e.g.: $\text{eval}(e, v, d)$.

- Alternatively: Treat object-language derivations as terms with dependent sorts.

- In the following: Will show minimal example.
Example 4: \( \lambda \)-calculus + case + numeral constructors

**Definition (\( \lambda^{sz,\text{case}} \))**

| Expressions | Exp ::= | \( x \mid \text{lam}\ x\ .\ e_0 \mid \text{app}\ e_1\ e_2 \mid z \mid s\ e_0 \mid \text{case}(e_0, e_1, x\ .\ e_2) \) |
|-------------|---------|
| Types       | \( \tau ::= \) | nat \( \mid \text{arr}\ \tau_2\ \tau_0 \) |
| CBN Eval.   | \( \mathcal{E} ::= \) | \( e \Downarrow v \) |
| Typing      | \( \mathcal{T} ::= \) | \( \Gamma \triangleright e : \tau \) |
| Num         | \( \mathcal{N} ::= \) | \( v \# \) |

- Numerals characterized in *object-language judgment*:

  \[
  \begin{align*}
  z \# & \quad \frac{}{v \#} \\
  s\ v \# & \quad \frac{}{v \#}
  \end{align*}
  \]

- *Could* axiomatize as atomic formula, \( A ::= \ldots \mid \text{isnum}(v) \).

- Alternatively: Treat \( v \# \) as a *dependent sort*, add structural witness formula.
Example 4: Logical Relation, Assertion Logic

Definition (Assertion Logic $\prod \text{eval,eq,num}^+$)

Propositions: $A, B :: \text{Form} ::= \forall \text{Exp } \alpha. A \mid \exists \text{Exp } \alpha. A \mid \exists (\text{e }\#) \alpha. A$

| $A \supset B \mid A \land B \mid A \lor B$
| $\text{eval}(e, v) \mid \text{eq}(e, e')$
| $\text{num}^+(\mathbb{N})$

Assumptions: $\Delta ::= \text{Assm} ::= \{A_1, \ldots, A_n\} \text{ (Unordered)}$

Parameters: $\Xi ::= \text{Ctx} ::= \cdot \mid \Xi, \alpha : \text{Exp} \mid \Xi, \alpha : (\text{e }\#)$

Proof sequent: $\Xi, \Delta \xrightarrow{c} A (c \in \{\bullet, \circ\})$

---

Definition (Logical Relation for Termination, Assertion-Level)

$\llbracket \text{nat} \rrbracket(e) \iff \exists \text{Exp } v. \text{eval}(e, v) \land \exists (v \#) \mathbb{N}. \text{num}^+(\mathbb{N})$

$\llbracket \text{arr } \tau_2 \tau_0 \rrbracket(e) \iff \forall \text{Exp } e_2. \llbracket \tau_2 \rrbracket(e_2) \supset \llbracket \tau_0 \rrbracket(\text{app } e e_2)$
“Well-sortedness” must be compositional w.r.t. substitution:

**Theorem (Compositionality)**

\[
\text{If } o :: S \text{ and } \Xi_1, \alpha : S, \Xi_2 | \Delta \xrightarrow{c} A \text{ then } \Xi_1, \Xi_2[o/\alpha] | \Delta[o/\alpha] \xrightarrow{c} A[o/\alpha].
\]

“Free” theorem: everything is represented in LF, contexts \( \Xi \) in particular.

Pfenning’s cut-admissibility theorem requires no changes!
Need to take care if we want to add equality conversion axioms to judgments on which we reason by case distinction.

**Example:** Let $e \equiv e'$ be axiomatization of syntactic equality. Treat as sort.

$e \equiv e$

$se \equiv s e'$

$s e \equiv s e'$

$e \equiv e'$

$s e_0 \equiv z$

$e \equiv e'$

$e' \equiv e''$

$e \equiv e''$

... 

**Goal:** From $s \ n \equiv s \ n'$ and $n \ n'$, infer $n' \ n'$.

Quantify over alternative judgment $e \not\equiv$ equivalent to $e \not\equiv$, but with explicit equality rules.
Resulting Assertion Logic

Definition (Assertion Logic \(\leftarrow\rightarrow_{\text{eval, num}^+, \Pi, \#=, \ast}\))

Propositions: \(A, B \quad \text{Form} \quad ::= \quad \forall^{\text{Exp}} \alpha. A \mid \exists^{\text{Exp}} \alpha. A\)

| \(\exists^{(e \#=)} \alpha. A \mid \exists^{(e \ast e')} \alpha. A\)
| \(A \supset B \mid A \land B \mid A \lor B\)
| \(\text{eval}(e, v) \mid \text{num}^+(N)\)

Assumptions: \(\Delta \quad \text{Assm} \quad ::= \quad \{A_1, \ldots, A_n\} \text{ (Unordered)}\)

Parameters: \(\Xi \quad \text{Ctx} \quad ::= \quad \cdot \mid \Xi, \alpha : \text{Exp} \mid \Xi, \alpha : (e \#=)\)

| \(\Xi, \alpha : (e \ast e')\)

Proof sequent: \(\Xi | \Delta \xrightarrow{c} A\) (\(c \in \{\bullet, \circ\}\))
Retain Canonicity of Derivations

How to define rules for \( e \# = \)?

**Bad:** Add extra rule \( \Rightarrow \) extra case to handle in all proofs:

\[
\begin{align*}
& z \# = \text{nz} & v' \# = \text{ns} & v \# = v' \text{ conv} \\
& s v' \# = \text{ns} & v \# = z & v' \# = s v' \text{ ns'}
\end{align*}
\]

**Good:** Make equality intrinsic property of all rules:

\[
\begin{align*}
& v \equiv z \quad \text{nz'} & v' \# = v' \equiv s v' \quad \text{ns'}
\end{align*}
\]

Derivations still canonical. Conversions pushed to equality derivations.
Example

Given $Q :: se' \simeq se$ and $N :: e \#=$, show $e' \#=$.

$N$ must end in nz’ or ns’.

Case $N = e_0 \#=$

$\frac{Q'}{e' \simeq e}$

$\frac{e_0 \#=}{\text{result}}$

Obtain result by

$\frac{Q}{se' \simeq se}$

$\frac{e' \simeq e}{e' \simeq se_0}$

$\frac{Q'}{se' \simeq se}$

$\frac{e' \simeq e}{e' \simeq se_0}$

$\boxed{\text{result}}$
• **Summary**

  • **Results**
    - Extension of SLR method to allow reasoning by case-analysis and equality.
    - More proofs can be formalized: see paper for observational equivalence proofs.
    - Nice property: Pfenning’s cut-elim proof works for dependently-sorted logic.

  • **Future work**
    - Lots of boilerplate. Code generation or extension of Twelf?
    - Experiment with stronger logics – no termination guarantees for cut-elim though.

  • **Questions?**

  - Code, paper, slides: see http://www.utr.dk/
Summary

- **Results**
  - Extension of SLR method to allow reasoning by case-analysis and equality.
  - More proofs can be formalized: see paper for observational equivalence proofs.
  - Nice property: Pfenning’s cut-elim proof works for dependently-sorted logic.

- **Future work**
  - Lots of boilerplate. Code generation or extension of Twelf?
  - Experiment with stronger logics – no termination guarantees for cut-elim though.
Summary

- **Results**
  - Extension of SLR method to allow reasoning by case-analysis and equality.
  - More proofs can be formalized: see paper for observational equivalence proofs.
  - Nice property: Pfenning’s cut-elim proof works for dependently-sorted logic.

- **Future work**
  - Lots of boilerplate. Code generation or extension of Twelf?
  - Experiment with stronger logics – no termination guarantees for cut-elim though.

- **Questions?**