Horn-formulas as Types for Structural Resolution

Peng Fu, Ekaterina Komendantskaya

University of Dundee
School of Computing

Introduction: Background

- Logic Programming(LP) is based on first-order Horn formula
- The execution of LP is by SLD-resolution
- SLD-resolution uses Robinson's unification

Introduction: Example of SLD-resolution

- Connectivity of graph with 3 nodes:
  $\kappa_1 : \text{Connect}(x, y), \text{Connect}(y, z) \Rightarrow \text{Connect}(x, z)$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2)$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3)$

Introduction: Example of SLD-resolution

- Connectivity of graph with 3 nodes:
  $\kappa_1 : \mathrm{Connect}(x, y), \mathrm{Connect}(y, z) \Rightarrow \mathrm{Connect}(x, z)$
  $\kappa_2 : \Rightarrow \mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_2)$
  $\kappa_3 : \Rightarrow \mathrm{Connect}(\mathrm{node}_2, \mathrm{node}_3)$
- query: $\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)$?

Introduction: Example of SLD-resolution

- Connectivity of graph with 3 nodes:
  $\kappa_1 : \mathrm{Connect}(x, y), \mathrm{Connect}(y, z) \Rightarrow \mathrm{Connect}(x, z)$
  $\kappa_2 : \Rightarrow \mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_2)$
  $\kappa_3 : \Rightarrow \mathrm{Connect}(\mathrm{node}_2, \mathrm{node}_3)$
- query: $\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)$?
- Execution trace:
  $\{\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_1, [\mathrm{node}_1/x, \mathrm{node}_3/z]}$
  $\{\mathrm{Connect}(\mathrm{node}_1, y), \mathrm{Connect}(y, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_2, [\mathrm{node}_1/x, \mathrm{node}_2/y, \mathrm{node}_3/z]}$
  $\{\mathrm{Connect}(\mathrm{node}_2, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_3} \emptyset$

Introduction: Example of SLD-resolution

- Connectivity of graph with 3 nodes:
  $\kappa_1 : \mathrm{Connect}(x, y), \mathrm{Connect}(y, z) \Rightarrow \mathrm{Connect}(x, z)$
  $\kappa_2 : \Rightarrow \mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_2)$
  $\kappa_3 : \Rightarrow \mathrm{Connect}(\mathrm{node}_2, \mathrm{node}_3)$
- query: $\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)$?
- Execution trace:
  $\{\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_1, [\mathrm{node}_1/x, \mathrm{node}_3/z]}$
  $\{\mathrm{Connect}(\mathrm{node}_1, y), \mathrm{Connect}(y, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_2, [\mathrm{node}_1/x, \mathrm{node}_2/y, \mathrm{node}_3/z]}$
  $\{\mathrm{Connect}(\mathrm{node}_2, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_3} \emptyset$
- So the answer for $\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)$ is yes.

Introduction: Motivation

Assumptions of LP

- Provide answers only when a query yields terminating execution
- Answering a query as proving a formula
- The notion of proof seems to be of little use in LP

Difficulties

- Hard to model infinite data, where the execution may not terminate
- How to understand the meaning a query when the query is not terminating

Introduction: Resolution by Term-Matching

- Let's call LP by SLD-resolution *LP-Unif*
- How about resolution by term-matching(LP-TM)?
- Unifiable $t_1 \sim_\gamma t_2$, i.e. $\gamma t_1 \equiv \gamma t_2$. Matchable $t_1 \mapsto_\sigma t_2$, i.e. $\sigma t_1 \equiv t_2$.
- A use case for LP-TM: Theorem proving

Introduction: Resolution by Term-Matching

- Let's call LP by SLD-resolution *LP-Unif*
- How about resolution by term-matching(LP-TM)?
- Unifiable $t_1 \sim_\gamma t_2$, i.e. $\gamma t_1 \equiv \gamma t_2$. Matchable $t_1 \mapsto_\sigma t_2$, i.e. $\sigma t_1 \equiv t_2$.
- A use case for LP-TM: Theorem proving
- Given axioms:
  $\Rightarrow Q(x)$
  $Q(x) \Rightarrow P(x)$
  Is $P(x)$ provable?
  $P(x) \rightarrow Q(x) \rightarrow \emptyset$

Introduction: Resolution by Term-Matching

- Let's call LP by SLD-resolution *LP-Unif*
- How about resolution by term-matching(LP-TM)?
- Unifiable $t_1 \sim_\gamma t_2$, i.e. $\gamma t_1 \equiv \gamma t_2$. Matchable $t_1 \mapsto_\sigma t_2$, i.e. $\sigma t_1 \equiv t_2$.
- A use case for LP-TM: Theorem proving
- Given axioms:
  $\Rightarrow Q(x)$
  $Q(x) \Rightarrow P(x)$
  Is $P(x)$ provable?
  $P(x) \to Q(x) \to \emptyset$
- Given axioms:
  $\Rightarrow Q(\mathrm{c})$
  $Q(x) \Rightarrow P(x)$
  Is $P(x)$ provable?
  $P(x) \to Q(x) \not\to$

Execution behavior of LP-TM

- Consider following Stream predicate:
  $\kappa : \mathrm{Stream}(y) \Rightarrow \mathrm{Stream}(\mathrm{cons}(x, y))$

Execution behavior of LP-TM

- Consider following Stream predicate:
  $\kappa : \mathrm{Stream}(y) \Rightarrow \mathrm{Stream}(\mathrm{cons}(x, y))$
- For query $\mathrm{Stream}(\mathrm{cons}(x, y))$, in LP-Unif:
  $\{\mathrm{Stream}(\mathrm{cons}(x, y))\} \rightsquigarrow_{\kappa,[x/x_1,y/y_1]}$
  $\{\mathrm{Stream}(y)\} \rightsquigarrow_{\kappa,[\mathrm{cons}(x_2,y_2)/y,x/x_1,\mathrm{cons}(x_2,y_2)/y_1,]} \{\mathrm{Stream}(y_2)\} \rightsquigarrow$
  ...

Execution behavior of LP-TM

- Consider following Stream predicate:
  $\kappa : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$
- For query $\text{Stream}(\text{cons}(x, y))$, in LP-Unif:
  $\{\text{Stream}(\text{cons}(x, y))\} \rightsquigarrow_{\kappa,[x/x_1,y/y_1]}$
  $\{\text{Stream}(y)\} \rightsquigarrow_{\kappa,[\text{cons}(x_2,y_2)/y,x/x_1,\text{cons}(x_2,y_2)/y_1,]} \{\text{Stream}(y_2)\} \rightsquigarrow$
  ...
- In LP-TM:
  $\{\text{Stream}(\text{cons}(x, y))\} \rightarrow_{\kappa,[x/x_1,y/y_1]} \{\text{Stream}(y)\}$

Limitations of LP-TM

- LP-TM not quite suitable for problem solving
  - The following logic program can describe finite bit list

    $\kappa_1 : \Rightarrow \mathrm{Bit}(0)$

    $\kappa_2 : \Rightarrow \mathrm{Bit}(1)$

    $\kappa_3 : \Rightarrow \mathrm{BList}(\mathrm{nil})$

    $\kappa_4 : \mathrm{Bit}(x), \mathrm{BList}(y) \Rightarrow \mathrm{BList}(\mathrm{cons}(x, y))$

Limitations of LP-TM

- LP-TM not quite suitable for problem solving
  - The following logic program can describe finite bit list
    $\kappa_1 : \Rightarrow \mathrm{Bit}(0)$
    $\kappa_2 : \Rightarrow \mathrm{Bit}(1)$
    $\kappa_3 : \Rightarrow \mathrm{BList}(\mathrm{nil})$
    $\kappa_4 : \mathrm{Bit}(x), \mathrm{BList}(y) \Rightarrow \mathrm{BList}(\mathrm{cons}(x, y))$
  - Consider query $\mathrm{BList}(\mathrm{cons}(x, y))$:
    $\{\mathrm{BList}(\mathrm{cons}(x, y))\} \rightarrow_{\kappa_4, [x/x_1, y/y_1]} \{\mathrm{Bit}(x), \mathrm{BList}(y)\}$

Limitations of LP-TM

- LP-TM not quite suitable for problem solving
  - The following logic program can describe finite bit list
    $\kappa_1 : \Rightarrow \mathrm{Bit}(0)$
    $\kappa_2 : \Rightarrow \mathrm{Bit}(1)$
    $\kappa_3 : \Rightarrow \mathrm{BList}(\mathrm{nil})$
    $\kappa_4 : \mathrm{Bit}(x), \mathrm{BList}(y) \Rightarrow \mathrm{BList}(\mathrm{cons}(x, y))$
  - Consider query $\mathrm{BList}(\mathrm{cons}(x, y))$:
    $\{\mathrm{BList}(\mathrm{cons}(x, y))\} \rightarrow_{\kappa_4, [x/x_1, y/y_1]} \{\mathrm{Bit}(x), \mathrm{BList}(y)\}$
  - But what is the answer for x, y?

Limitations of LP-TM

- LP-TM not quite suitable for problem solving
  - The following logic program can describe finite bit list
    $\kappa_1 : \Rightarrow \mathrm{Bit}(0)$
    $\kappa_2 : \Rightarrow \mathrm{Bit}(1)$
    $\kappa_3 : \Rightarrow \mathrm{BList}(\mathrm{nil})$
    $\kappa_4 : \mathrm{Bit}(x), \mathrm{BList}(y) \Rightarrow \mathrm{BList}(\mathrm{cons}(x, y))$
  - Consider query $\mathrm{BList}(\mathrm{cons}(x, y))$:
    $\{\mathrm{BList}(\mathrm{cons}(x, y))\} \rightarrow_{\kappa_4, [x/x_1, y/y_1]} \{\mathrm{Bit}(x), \mathrm{BList}(y)\}$
  - But what is the answer for x, y?
  - We need unification to compute substitution: $x \sim 0, y \sim \mathrm{nil}$

Limitations of LP-TM

- LP-TM not quite suitable for problem solving
  - The following logic program can describe finite bit list
    $\kappa_1 : \Rightarrow \mathrm{Bit}(0)$
    $\kappa_2 : \Rightarrow \mathrm{Bit}(1)$
    $\kappa_3 : \Rightarrow \mathrm{BList}(\mathrm{nil})$
    $\kappa_4 : \mathrm{Bit}(x), \mathrm{BList}(y) \Rightarrow \mathrm{BList}(\mathrm{cons}(x, y))$
  - Consider query $\mathrm{BList}(\mathrm{cons}(x, y))$:
    $\{\mathrm{BList}(\mathrm{cons}(x, y))\} \rightarrow_{\kappa_4, [x/x_1, y/y_1]} \{\mathrm{Bit}(x), \mathrm{BList}(y)\}$
  - But what is the answer for x, y?
  - We need unification to compute substitution: $x \sim 0, y \sim \mathrm{nil}$
- The combination of LP-TM with substitution computed by unification leads to **Structural Resolution**

Formalism: LP-Unif, LP-TM and LP-Struct

- **Term-matching reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightarrow_{\kappa,\sigma} \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \mapsto_\sigma A_i$.

Formalism: LP-Unif, LP-TM and LP-Struct

- **Term-matching reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightarrow_{\kappa,\sigma} \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \mapsto_\sigma A_i$.
- **Unification reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_{\kappa,\gamma\cdot\gamma'} \{\gamma A_1, ..., \gamma B_1, ..., \gamma B_m, ..., \gamma A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \sim_\gamma A_i$.

Formalism: LP-Unif, LP-TM and LP-Struct

- **Term-matching reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightarrow_{\kappa,\sigma} \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \mapsto_\sigma A_i$.
- **Unification reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_{\kappa,\gamma\cdot\gamma'} \{\gamma A_1, ..., \gamma B_1, ..., \gamma B_m, ..., \gamma A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \sim_\gamma A_i$.
- **Substitutional reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \hookrightarrow_{\kappa,\gamma\cdot\gamma'} \{\gamma A_1, ..., \gamma A_i, ..., \gamma A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \sim_\gamma A_i$.

Formalism: LP-Unif, LP-TM and LP-Struct

- **Term-matching reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightarrow_{\kappa,\sigma} \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \mapsto_\sigma A_i$.
- **Unification reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_{\kappa,\gamma\cdot\gamma'} \{\gamma A_1, ..., \gamma B_1, ..., \gamma B_m, ..., \gamma A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \sim_\gamma A_i$.
- **Substitutional reduction:**
  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \hookrightarrow_{\kappa,\gamma\cdot\gamma'} \{\gamma A_1, ..., \gamma A_i, ..., \gamma A_n\}$, if there exists $\kappa : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \sim_\gamma A_i$.
- **LP-TM:** $(\Phi, \rightarrow)$
  LP-Unif: $(\Phi, \rightsquigarrow)$
  LP-Struct: $(\Phi, \rightarrow^\mu \cdot \hookrightarrow^1)$

LP-Struct: Stream

$\kappa : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$
For query $\text{Stream}(\text{cons}(x, y))$, in LP-Struct:

- $\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$

LP-Struct: Stream

$\kappa : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$
For query $\text{Stream}(\text{cons}(x, y))$, in LP-Struct:

- $\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$
- $\hookrightarrow_{[\text{cons}(x_1, y_1)/y]} \{\text{Stream}(\text{cons}(x_1, y_1))\} \rightarrow \{\text{Stream}(y_1)\}$

LP-Struct: Stream

$\kappa : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$
For query $\text{Stream}(\text{cons}(x, y))$, in LP-Struct:

- $\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$
- $\hookrightarrow_{[\text{cons}(x_1,y_1)/y]} \{\text{Stream}(\text{cons}(x_1, y_1))\} \rightarrow \{\text{Stream}(y_1)\}$
- $\hookrightarrow_{[\text{cons}(x_2,y_2)/y_1,\text{cons}(x_1,\text{cons}(x_2,y_2))/y]} \{\text{Stream}(\text{cons}(x_2, y_2))\} \rightarrow \{\text{Stream}(y_2)\}$

LP-Struct: Stream

$\kappa : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$
For query $\text{Stream}(\text{cons}(x, y))$, in LP-Struct:

- $\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$
- $\hookrightarrow_{[\text{cons}(x_1,y_1)/y]} \{\text{Stream}(\text{cons}(x_1, y_1))\} \rightarrow \{\text{Stream}(y_1)\}$
- $\hookrightarrow_{[\text{cons}(x_2,y_2)/y_1,\text{cons}(x_1,\text{cons}(x_2,y_2))/y]} \{\text{Stream}(\text{cons}(x_2, y_2))\} \rightarrow \{\text{Stream}(y_2)\}$
- $\hookrightarrow_{[\text{cons}(x_3,y_3)/y_2,\text{cons}(x_2,\text{cons}(x_3,y_3))/y_1,\text{cons}(x_1,\text{cons}(x_2,\text{cons}(x_3,y_3)))/y]}$ $\{\text{Stream}(\text{cons}(x_3, y_3))\} \rightarrow \{\text{Stream}(y_3)\}$

LP-Struct: Stream

$\kappa : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$
For query $\text{Stream}(\text{cons}(x, y))$, in LP-Struct:

- $\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$
- $\hookrightarrow_{[\text{cons}(x_1,y_1)/y]} \{\text{Stream}(\text{cons}(x_1, y_1))\} \rightarrow \{\text{Stream}(y_1)\}$
- $\hookrightarrow_{[\text{cons}(x_2,y_2)/y_1,\text{cons}(x_1,\text{cons}(x_2,y_2))/y]} \{\text{Stream}(\text{cons}(x_2, y_2))\} \rightarrow \{\text{Stream}(y_2)\}$
- $\hookrightarrow_{[\text{cons}(x_3,y_3)/y_2,\text{cons}(x_2,\text{cons}(x_3,y_3))/y_1,\text{cons}(x_1,\text{cons}(x_2,\text{cons}(x_3,y_3)))/y]}$ $\{\text{Stream}(\text{cons}(x_3, y_3))\} \rightarrow \{\text{Stream}(y_3)\}$
- ...
- Partial answer: $\text{cons}(x_1, \text{cons}(x_2, \text{cons}(x_3, y_3)))/y$

Question: Relation between LP-Unif and LP-Struct?

- Both LP-Unif and LP-Struct are sound w.r.t. Herbrand Model

Question: Relation between LP-Unif and LP-Struct?

- Both LP-Unif and LP-Struct are sound w.r.t. Herbrand Model
- Operationally, They seem similar but a little different

Question: Relation between LP-Unif and LP-Struct?

- Both LP-Unif and LP-Struct are sound w.r.t. Herbrand Model
- Operationally, They seem similar but a little different
- Again, the graph example

  $\kappa_1 : \text{Connect}(x, y), \text{Connect}(y, z) \Rightarrow \text{Connect}(x, z)$

  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2)$

  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3)$

Question: Relation between LP-Unif and LP-Struct?

- Both LP-Unif and LP-Struct are sound w.r.t. Herbrand Model
- Operationally, They seem similar but a little different
- Again, the graph example
  $\kappa_1 : \text{Connect}(x, y), \text{Connect}(y, z) \Rightarrow \text{Connect}(x, z)$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2)$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3)$
- $\text{Connect}(\text{node}_1, \text{node}_3)$ in LP-Unif terminates.

Question: Relation between LP-Unif and LP-Struct?

- Both LP-Unif and LP-Struct are sound w.r.t. Herbrand Model
- Operationally, They seem similar but a little different
- Again, the graph example
  $\kappa_1 : \text{Connect}(x, y), \text{Connect}(y, z) \Rightarrow \text{Connect}(x, z)$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2)$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3)$
- $\text{Connect}(\text{node}_1, \text{node}_3)$ in LP-Unif terminates.
- For LP-Struct:
  $\Phi \vdash \{\text{Connect}(\text{node}_1, \text{node}_3)\} \rightarrow_{\kappa_1, [\text{node}_1/x, \text{node}_3/z]}$
  $\{\text{Connect}(\text{node}_1, y), \text{Connect}(y, \text{node}_3)\} \rightarrow_{\kappa_1, [\text{node}_1/x, y/z]}$
  $\{\text{Connect}(\text{node}_1, y_1), \text{Connect}(y_1, y), \text{Connect}(y, \text{node}_3)\} \rightarrow_{\kappa_1}$
  ...

Formalization of a Type System

- Term $t \ ::= \ x \mid f(t_1, ..., t_n)$
  Atomic Formula $A, B, C, D \ ::= \ P(t_1, ..., t_n)$
  (Horn) Formula $F \ ::= \ A_1, ..., A_n \Rightarrow A$
  Proof Term $p, e \ ::= \ \kappa \mid a \mid \lambda a.e \mid e \ e'$

Formalization of a Type System

- Term $t \ ::= \ x \mid f(t_1, ..., t_n)$
  Atomic Formula $A, B, C, D \ ::= \ P(t_1, ..., t_n)$
  (Horn) Formula $F \ ::= \ A_1, ..., A_n \Rightarrow A$
  Proof Term $p, e \ ::= \ \kappa \mid a \mid \lambda a.e \mid e \ e'$
- Girard's observation on intuitionistic sequent calculus with atomic formulas

$$\frac{}{\underline{B} \vdash A} \ axiom \qquad \frac{\underline{B} \vdash C}{\sigma\underline{B} \vdash \sigma C} \ subst \qquad \frac{\underline{A} \vdash D \quad \underline{B}, D \vdash C}{\underline{A}, \underline{B} \vdash C} \ cut$$

Formalization of a Type System

- Term $t \ ::= \ x \mid f(t_1, ..., t_n)$
  Atomic Formula $A, B, C, D \ ::= \ P(t_1, ..., t_n)$
  (Horn) Formula $F \ ::= \ A_1, ..., A_n \Rightarrow A$
  Proof Term $p, e \ ::= \ \kappa \mid a \mid \lambda a.e \mid e \ e'$
- Girard's observation on intuitionistic sequent calculus with atomic formulas

$$\frac{}{\underline{B} \vdash A} \ axiom \qquad \frac{\underline{B} \vdash C}{\sigma\underline{B} \vdash \sigma C} \ subst \qquad \frac{\underline{A} \vdash D \quad \underline{B}, D \vdash C}{\underline{A}, \underline{B} \vdash C} \ cut$$

- Is $\vdash Q$ provable?

Formalization of a Type System

- Term $t \ ::= \ x \mid f(t_1, ..., t_n)$
  Atomic Formula $A, B, C, D \ ::= \ P(t_1, ..., t_n)$
  (Horn) Formula $F \ ::= \ A_1, ..., A_n \Rightarrow A$
  Proof Term $p, e \ ::= \ \kappa \mid a \mid \lambda a.e \mid e \ e'$
- Girard's observation on intuitionistic sequent calculus with atomic formulas

$$\frac{}{\underline{B} \vdash A} \ axiom \qquad \frac{\underline{B} \vdash C}{\sigma\underline{B} \vdash \sigma C} \ subst \qquad \frac{\underline{A} \vdash D \quad \underline{B}, D \vdash C}{\underline{A}, \underline{B} \vdash C} \ cut$$

- Is $\vdash Q$ provable?
- We internalized "$\vdash$" as "$\Rightarrow$" and add proof term annotations

$$\frac{}{\kappa : \forall \underline{x}.F} \ axiom \qquad \frac{e : F}{e : \forall \underline{x}.F} \ gen$$

$$\frac{e : \forall \underline{x}.F}{e : [\underline{t}/\underline{x}]F} \ inst \qquad \frac{e_1 : \underline{A} \Rightarrow D \quad e_2 : \underline{B}, D \Rightarrow C}{\lambda \underline{a}.\lambda \underline{b}.(e_2 \ \underline{b}) \ (e_1 \ \underline{a}) : \underline{A}, \underline{B} \Rightarrow C} \ cut$$

Soundness of LP-TM and LP-Unif

- Soundness of LP-Unif
  If $\Phi \vdash \{A\} \rightsquigarrow_{\gamma}^{*} \emptyset$, then there exists a proof $e : \forall \underline{x}.\ \Rightarrow \gamma A$ given axioms Φ.
- Soundness of LP-TM
  If $\Phi \vdash \{A\} \rightarrow^{*} \emptyset$, then there exists a proof $e : \forall \underline{x}.\ \Rightarrow A$ given axioms Φ.
- For example, the LP-Unif reductions:
  $\{\mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_1, [\mathrm{node}_1/x, \mathrm{node}_3/z]}$
  $\{\mathrm{Connect}(\mathrm{node}_1, y), \mathrm{Connect}(y, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_2, [\mathrm{node}_1/x, \mathrm{node}_2/y, \mathrm{node}_3/z]}$
  $\{\mathrm{Connect}(\mathrm{node}_2, \mathrm{node}_3)\} \rightsquigarrow_{\kappa_3} \emptyset$
- The reduction yields a proof $(\lambda b.(\kappa_1\ b)\ \kappa_3)\ \kappa_2$ for the formula $\Rightarrow \mathrm{Connect}(\mathrm{node}_1, \mathrm{node}_3)$.

Useful Properties about the Type System

- Strong Normalization
  If $e : F$, then e is strongly normalizable w.r.t. beta-reduction on proof terms.
- First Orderness
  If $e : [\forall \underline{x}.]\underline{A} \Rightarrow B$ given axioms Φ, then either e is a proof term constant or it is normalizable to the form $\lambda \underline{a}.n$, where n is first order normal proof term.
- If $e : [\forall \underline{x}.] \Rightarrow B$, then e is normalizable to a first order proof term.

Realizability Transformation

- Inspired from Kleene's realizability:
  φ realize $A \Rightarrow B$ iff for any number a realizes A and $\varphi(a)$ realizes B.

Realizability Transformation

- Inspired from Kleene's realizability:
  φ realize $A \Rightarrow B$ iff for any number a realizes A and $\varphi(a)$ realizes B.
- Representing First Order Proof Term
  Let ϕ be a mapping from proof term variables to first order terms.
  – $[\![a]\!]_\phi := \phi(a)$
  – $[\![\kappa\ p_1 ... p_n]\!]_\phi := f_\kappa([\![p_1]\!]_\phi, ..., [\![p_n]\!]_\phi)$

Realizability Transformation

- Inspired from Kleene's realizability:
  φ realize $A \Rightarrow B$ iff for any number a realizes A and $\varphi(a)$ realizes B.
- Representing First Order Proof Term
  Let ϕ be a mapping from proof term variables to first order terms.
  – $[\![a]\!]_\phi := \phi(a)$
  – $[\![\kappa\, p_1 ... p_n]\!]_\phi := f_\kappa([\![p_1]\!]_\phi, ..., [\![p_n]\!]_\phi)$
- For $A \equiv P(\underline{x})$, we write $A[y] \equiv P(\underline{x}, y)$. Similarly, $A[t] \equiv P(\underline{x}, t)$

Realizability Transformation

- Inspired from Kleene's realizability:
  φ realize $A \Rightarrow B$ iff for any number a realizes A and $\varphi(a)$ realizes B.
- Representing First Order Proof Term
  Let ϕ be a mapping from proof term variables to first order terms.
  – $[\![a]\!]_\phi := \phi(a)$
  – $[\![\kappa\, p_1 ... p_n]\!]_\phi := f_\kappa([\![p_1]\!]_\phi, ..., [\![p_n]\!]_\phi)$
- For $A \equiv P(\underline{x})$, we write $A[y] \equiv P(\underline{x}, y)$. Similarly, $A[t] \equiv P(\underline{x}, t)$
- Realizability transformation F on normal proofs

Realizability Transformation

- Inspired from Kleene's realizability:
  φ realize $A \Rightarrow B$ iff for any number a realizes A and $\varphi(a)$ realizes B.
- Representing First Order Proof Term
  Let ϕ be a mapping from proof term variables to first order terms.
  – $[\![a]\!]_\phi := \phi(a)$
  – $[\![\kappa\, p_1 ... p_n]\!]_\phi := f_\kappa([\![p_1]\!]_\phi, ..., [\![p_n]\!]_\phi)$
- For $A \equiv P(\underline{x})$, we write $A[y] \equiv P(\underline{x}, y)$. Similarly, $A[t] \equiv P(\underline{x}, t)$
- Realizability transformation F on normal proofs
  - $F(\kappa : \forall \underline{x}.A_1, ..., A_m \Rightarrow B) :=$
    $\kappa : \forall \underline{x}.\forall \underline{y}.A_1[y_1], ..., A_m[y_m] \Rightarrow B[f_\kappa(y_1, ..., y_m)]$

Realizability Transformation

- Inspired from Kleene's realizability:
  φ realize $A \Rightarrow B$ iff for any number a realizes A and $\varphi(a)$ realizes B.
- Representing First Order Proof Term
  Let ϕ be a mapping from proof term variables to first order terms.
  – $[\![a]\!]_\phi := \phi(a)$
  – $[\![\kappa\, p_1 ... p_n]\!]_\phi := f_\kappa([\![p_1]\!]_\phi, ..., [\![p_n]\!]_\phi)$
- For $A \equiv P(\underline{x})$, we write $A[y] \equiv P(\underline{x}, y)$. Similarly, $A[t] \equiv P(\underline{x}, t)$
- Realizability transformation F on normal proofs
  - $F(\kappa : \forall \underline{x}.A_1, ..., A_m \Rightarrow B) :=$
    $\kappa : \forall \underline{x}.\forall \underline{y}.A_1[y_1], ..., A_m[y_m] \Rightarrow B[f_\kappa(y_1, ..., y_m)]$
  - $F(\lambda \underline{a}.n : [\forall \underline{x}].A_1, ..., A_m \Rightarrow B) :=$
    $\lambda \underline{a}.n : [\forall \underline{x}.\forall \underline{y}].A_1[y_1], ..., A_m[y_m] \Rightarrow B[[\![n]\!]_{[\underline{y}/\underline{a}]}]$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.
  - $\{\text{Connect}(\text{node}_1, \text{node}_3, u)\}$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.
  - $\{\text{Connect}(\text{node}_1, \text{node}_3, u)\}$
  - $\hookrightarrow_{\kappa_1, [\text{node}_1/x, \text{node}_3/z, f_{\kappa_1}(u_1, u_2)/u]}$
    $\{\text{Connect}(\text{node}_1, \text{node}_3, f_{\kappa_1}(u_1, u_2))\} \rightarrow_{\kappa_1}$
    $\{\text{Connect}(\text{node}_1, y, u_1), \text{Connect}(y, \text{node}_3, u_2)\}$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.
  - $\{\text{Connect}(\text{node}_1, \text{node}_3, u)\}$
  - $\hookrightarrow_{\kappa_1, [\text{node}_1/x, \text{node}_3/z, f_{\kappa_1}(u_1, u_2)/u]}$
    $\{\text{Connect}(\text{node}_1, \text{node}_3, f_{\kappa_1}(u_1, u_2))\} \rightarrow_{\kappa_1}$
    $\{\text{Connect}(\text{node}_1, y, u_1), \text{Connect}(y, \text{node}_3, u_2)\}$
  - $\hookrightarrow_{\kappa_2, [c_{\kappa_2}/u_1, \text{node}_1/x, \text{node}_2/y, , \text{node}_3/z, f_{\kappa_1}(c_{\kappa_2}, u_2)/u]}$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.
  - $\{\text{Connect}(\text{node}_1, \text{node}_3, u)\}$
  - $\hookrightarrow_{\kappa_1, [\text{node}_1/x, \text{node}_3/z, f_{\kappa_1}(u_1, u_2)/u]}$
    $\{\text{Connect}(\text{node}_1, \text{node}_3, f_{\kappa_1}(u_1, u_2))\} \rightarrow_{\kappa_1}$
    $\{\text{Connect}(\text{node}_1, y, u_1), \text{Connect}(y, \text{node}_3, u_2)\}$
  - $\hookrightarrow_{\kappa_2, [c_{\kappa_2}/u_1, \text{node}_1/x, \text{node}_2/y, , \text{node}_3/z, f_{\kappa_1}(c_{\kappa_2}, u_2)/u]}$
  - $\{\text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2}), \text{Connect}(\text{node}_2, \text{node}_3, u_2)\} \rightarrow_{\kappa_2}$
    $\{\text{Connect}(\text{node}_2, \text{node}_3, u_2)\}$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.
  - $\{\text{Connect}(\text{node}_1, \text{node}_3, u)\}$
  - $\hookrightarrow_{\kappa_1, [\text{node}_1/x, \text{node}_3/z, f_{\kappa_1}(u_1, u_2)/u]}$
    $\{\text{Connect}(\text{node}_1, \text{node}_3, f_{\kappa_1}(u_1, u_2))\} \rightarrow_{\kappa_1}$
    $\{\text{Connect}(\text{node}_1, y, u_1), \text{Connect}(y, \text{node}_3, u_2)\}$
  - $\hookrightarrow_{\kappa_2, [c_{\kappa_2}/u_1, \text{node}_1/x, \text{node}_2/y, , \text{node}_3/z, f_{\kappa_1}(c_{\kappa_2}, u_2)/u]}$
  - $\{\text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2}), \text{Connect}(\text{node}_2, \text{node}_3, u_2)\} \rightarrow_{\kappa_2}$
    $\{\text{Connect}(\text{node}_2, \text{node}_3, u_2)\}$
  - $\hookrightarrow_{\kappa_3, [c_{\kappa_3}/u_2, c_{\kappa_2}/u_1, \text{node}_3/z, \text{node}_1/x, \text{node}_2/y, f_{\kappa_1}(c_{\kappa_2}, c_{\kappa_3})/u]}$
    $\{\text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})\} \rightarrow_{\kappa_3} \emptyset$

Realizability Transformation: Example

- Connectivity after realizability transformation:
  $\kappa_1 : \text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2) \Rightarrow \text{Connect}(x, z, f_{\kappa_1}(u_1, u_2))$
  $\kappa_2 : \Rightarrow \text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2})$
  $\kappa_3 : \Rightarrow \text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})$
- LP-Struct reduction for $\text{Connect}(\text{node}_1, \text{node}_3, u)$.
  - $\{\text{Connect}(\text{node}_1, \text{node}_3, u)\}$
  - $\hookrightarrow_{\kappa_1, [\text{node}_1/x, \text{node}_3/z, f_{\kappa_1}(u_1, u_2)/u]}$
    $\{\text{Connect}(\text{node}_1, \text{node}_3, f_{\kappa_1}(u_1, u_2))\} \rightarrow_{\kappa_1}$
    $\{\text{Connect}(\text{node}_1, y, u_1), \text{Connect}(y, \text{node}_3, u_2)\}$
  - $\hookrightarrow_{\kappa_2, [c_{\kappa_2}/u_1, \text{node}_1/x, \text{node}_2/y, , \text{node}_3/z, f_{\kappa_1}(c_{\kappa_2}, u_2)/u]}$
  - $\{\text{Connect}(\text{node}_1, \text{node}_2, c_{\kappa_2}), \text{Connect}(\text{node}_2, \text{node}_3, u_2)\} \rightarrow_{\kappa_2}$
    $\{\text{Connect}(\text{node}_2, \text{node}_3, u_2)\}$
  - $\hookrightarrow_{\kappa_3, [c_{\kappa_3}/u_2, c_{\kappa_2}/u_1, \text{node}_3/z, \text{node}_1/x, \text{node}_2/y, f_{\kappa_1}(c_{\kappa_2}, c_{\kappa_3})/u]}$
    $\{\text{Connect}(\text{node}_2, \text{node}_3, c_{\kappa_3})\} \rightarrow_{\kappa_3} \emptyset$
- Answer: $f_{\kappa_1}(c_{\kappa_2}, c_{\kappa_3})/u$

Results about Realizability Transformation

- **Termination of term-matching reduction**
  For any $(\Phi, \rightarrow^{\mu} \cdot \hookrightarrow^{1})$, we have $(F(\Phi), \rightarrow^{\nu} \cdot \hookrightarrow^{1})$
- **Preserve Provability**
  Given axioms Φ, if $e : [\forall \underline{x}].\underline{A} \Rightarrow B$ holds with e in normal form, then $F(e : [\forall \underline{x}].\underline{A} \Rightarrow B)$ holds for axioms $F(\Phi)$
- **Recording Proof**
  Suppose $F(\Phi) \vdash \{A[y]\} \rightsquigarrow_{\gamma}^{*} \emptyset$. We have $p : \forall \underline{x}. \Rightarrow \gamma A[\gamma y]$ for $F(\Phi)$, where p is in normal form and $[\![p]\!]_{\emptyset} = \gamma y$
- **Preserve Unification**
  $\Phi \vdash \{A\} \rightsquigarrow^{*} \emptyset$ iff $F(\Phi) \vdash \{A[y]\} \rightsquigarrow^{*} \emptyset$
- **Operational Equivalent of LP-Unif and LP-Struct**
  $F(\Phi) \vdash \{A[y]\} \rightsquigarrow^{*} \emptyset$ iff $F(\Phi) \vdash \{A[y]\}(\rightarrow^{\nu} \cdot \hookrightarrow^{1})^{*}\emptyset$.

Summary and Future Work

- We define a type system to model LP-TM, LP-Unif and LP-Struct
- We define a transformation called realizability transformation
- Realizability transformation preserves proof content
- We show LP-Unif and LP-Struct are operationally equivalent after the tranformation
- Future works: Apply LP-TM to analyze type class inference in functional langauges

Future Work

- In type class inference, proof has computational meaning:

```
class Eq A where
  eq :: Eq A => A -> A -> Bool

instance => Eq Int where ..
instance Eq A => Eq (List A) where ..

test = eq [] [1]
```

- `test` function will generate a query `Eq (List Int)`
- `Eq (List Int) ==> Eq Int ==> empty`
- The proof of the query `Eq (List Int)` will be passed as an input for `eq`