

# A Type Theoretic Approach to Structural Resolution

Peng Fu, Ekaterina Komendantskaya

University of Dundee  
School of Computing

# Logic programming and Proof

- ▶  $k1 : \text{Connect}(x, y), \text{Connect}(y, z) \Rightarrow \text{Connect}(x, z)$   
 $k2 : \Rightarrow \text{Connect}(N1, N2)$   
 $k3 : \Rightarrow \text{Connect}(N2, N3)$
- ▶ **Are there any proof of  $\text{Connect}(x, N3)$  for some  $x$ ?**  
**Answer 1:**  $(k1\ k2\ k3)$  with  $[N1/x]$   
**Answer 2:**  $k3$  with  $[N2/x]$

# Logic programming and Proof

## Type Class in Functional Language(e.g. Haskell)

- ▶ `k1 : Eq(x) => Eq(List(x))`  
`k2 : => Eq(Int)`

`eq : Eq(x) => x -> x -> Bool`

`test = eq d [1,3] [1,2, 3]`

- ▶ **What is the proof of `Eq(List(Int))`?**
- ▶ **`d = (k1 k2)` is a proof of `Eq(List(Int))`!**

# Resolutions

$k : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$

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

► SLD-resolution:

$\{\text{Stream}(\text{cons}(x, y))\} \rightsquigarrow \{\text{Stream}(y)\} \rightsquigarrow$   
 $\{\text{Stream}(y2)\} \rightsquigarrow \dots$

Matcher:  $\sigma t_1 \equiv t_2$ , Unifier:  $\sigma t_1 \equiv \sigma t_2$

# Resolutions

$k : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$

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

► SLD-resolution:

$\{\text{Stream}(\text{cons}(x, y))\} \rightsquigarrow \{\text{Stream}(y)\} \rightsquigarrow$   
 $\{\text{Stream}(y2)\} \rightsquigarrow \dots$

► Resolution by Term matching:

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

Matcher:  $\sigma t_1 \equiv t_2$ , Unifier:  $\sigma t_1 \equiv \sigma t_2$

# Resolutions

$k : \text{Stream}(y) \Rightarrow \text{Stream}(\text{cons}(x, y))$

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

► **SLD-resolution:**

$\{\text{Stream}(\text{cons}(x, y))\} \rightsquigarrow \{\text{Stream}(y)\} \rightsquigarrow$   
 $\{\text{Stream}(y_2)\} \rightsquigarrow \dots$

► **Resolution by Term matching:**

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

► **Structural Resolution(Matching + Unification):**

$\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$   
 $\hookrightarrow \{\text{Stream}(\text{cons}(x_1, y_1))\} \rightarrow \{\text{Stream}(y_1)\}$   
 $\hookrightarrow \{\text{Stream}(\text{cons}(x_2, y_2))\} \rightarrow \{\text{Stream}(y_2)\}$   
 $\hookrightarrow \{\text{Stream}(\text{cons}(x_3, y_3))\} \rightarrow \{\text{Stream}(y_3)\} \dots$

**Matcher:**  $\sigma t_1 \equiv t_2$ , **Unifier:**  $\sigma t_1 \equiv \sigma t_2$

## A Few Definitions

► **Term-matching reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightarrow_{\kappa} \{A_1, \dots, \sigma B_1, \dots, \sigma B_m, \dots, A_n\}$ , if  
there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\sigma C \equiv A_i$ .

## A Few Definitions

► **Term-matching reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightarrow_{\kappa} \{A_1, \dots, \sigma B_1, \dots, \sigma B_m, \dots, A_n\}$ , if  
there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\sigma C \equiv A_i$ .

► **Unification reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightsquigarrow_{\kappa, \gamma, \gamma'} \{\gamma A_1, \dots, \gamma B_1, \dots, \gamma B_m, \dots, \gamma A_n\}$ ,  
if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\gamma C \equiv \gamma A_i$ .



## A Few Definitions

► **Term-matching reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightarrow_{\kappa} \{A_1, \dots, \sigma B_1, \dots, \sigma B_m, \dots, A_n\}$ , if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\sigma C \equiv A_i$ .

► **Unification reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightsquigarrow_{\kappa, \gamma, \gamma'} \{\gamma A_1, \dots, \gamma B_1, \dots, \gamma B_m, \dots, \gamma A_n\}$ , if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\gamma C \equiv \gamma A_i$ .

► **Substitutional reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \hookrightarrow_{\kappa, \gamma, \gamma'} \{\gamma A_1, \dots, \gamma A_i, \dots, \gamma A_n\}$ , if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\gamma C \equiv \gamma A_i$ .

## A Few Definitions

► **Term-matching reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightarrow_{\kappa} \{A_1, \dots, \sigma B_1, \dots, \sigma B_m, \dots, A_n\}$ , if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\sigma C \equiv A_i$ .

► **Unification reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \rightsquigarrow_{\kappa, \gamma, \gamma'} \{\gamma A_1, \dots, \gamma B_1, \dots, \gamma B_m, \dots, \gamma A_n\}$ , if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\gamma C \equiv \gamma A_i$ .

► **Substitutional reduction:**

$\Phi \vdash \{A_1, \dots, A_i, \dots, A_n\} \hookrightarrow_{\kappa, \gamma, \gamma'} \{\gamma A_1, \dots, \gamma A_i, \dots, \gamma A_n\}$ , if there exists  $\kappa : B_1, \dots, B_m \Rightarrow C \in \Phi$  such that  $\gamma C \equiv \gamma A_i$ .

► **LP-TM:**  $(\Phi, \rightarrow)$

**LP-Unif:**  $(\Phi, \rightsquigarrow)$

**LP-Struct:**  $(\Phi, \rightarrow^{\mu} \cdot \hookrightarrow^1)$

# LP-Unif and LP-Struct

**Question 1.** What is the relation between LP-Unif and LP-Struct?

- ▶ Again, the graph example

k1 :  $\text{Connect}(x, y), \text{Connect}(y, z) \Rightarrow \text{Connect}(x, z)$

k2 :  $\Rightarrow \text{Connect}(N1, N2)$

k3 :  $\Rightarrow \text{Connect}(N2, N3)$

- ▶  $\text{Connect}(N1, N3)$  in LP-Unif has a finite path.

- ▶ For LP-Struct:

$\{\text{Connect}(N1, N3)\} \rightarrow_{\kappa_1, [N1/x, N3/z]}$

$\{\text{Connect}(N1, y), \text{Connect}(y, N3)\} \rightarrow_{\kappa_1, [N1/x, y/z]}$

$\{\text{Connect}(N1, y1), \text{Connect}(y1, y), \text{Connect}(y, N3)\}$

$\rightarrow_{\kappa_1} \dots$

## A Notion of Productivity

- ▶ We say a logic program is *productive* if  $\rightarrow$  is terminating
- ▶ Productive programs allow finite observation, e.g. stream
$$\{\text{Stream}(\text{cons}(x, y))\} \rightarrow \{\text{Stream}(y)\}$$
$$\hookrightarrow \{\text{Stream}(\text{cons}(x_1, y_1))\} \rightarrow \{\text{Stream}(y_1)\}$$
$$\hookrightarrow \{\text{Stream}(\text{cons}(x_2, y_2))\} \rightarrow \{\text{Stream}(y_2)\}$$
$$\hookrightarrow \{\text{Stream}(\text{cons}(x_3, y_3))\} \rightarrow \{\text{Stream}(y_3)\} \dots$$
- ▶ There are nonproductive programs, e.g. graph

# LP-Unif and LP-Struct

**Question 2.** Given  $\rightarrow$  is terminating, what is the relation between LP-Unif and LP-Struct?

- ▶  $k1 : \Rightarrow P(c)$   
     $k2 : Q(x) \Rightarrow P(x)$
- ▶ LP-Unif:  $P(x) \rightsquigarrow \emptyset$
- ▶ LP-Struct:  $P(x) \rightarrow Q(x)$

# Realizability Transformation

## Reflecting proofs into formulas

$k_1$  :  $\text{Connect}(x, y, u_1), \text{Connect}(y, z, u_2)$   
           $\Rightarrow \text{Connect}(x, z, k_1(u_1, u_2))$   
 $k_2$  :  $\Rightarrow \text{Connect}(N_1, N_2, k_2)$   
 $k_3$  :  $\Rightarrow \text{Connect}(N_2, N_3, k_3)$

## LP-Struct:

$\{\text{Connect}(N_1, N_3, u)\} \leftrightarrow \{\text{Connect}(N_1, N_3, k_1(u_1, u_2))\} \rightarrow$   
 $\{\text{Connect}(N_1, y, u_1), \text{Connect}(y, N_3, u_2)\} \leftrightarrow$   
 $\{\text{Connect}(N_1, N_2, k_2), \text{Connect}(N_2, N_3, u_2)\} \rightarrow$   
 $\{\text{Connect}(N_2, N_3, u_2)\} \leftrightarrow \{\text{Connect}(N_2, N_3, k_3)\} \rightarrow \emptyset$

Final Answer:  $[k_1(k_2, k_3)/u]$

# Realizability Transformation

$k1 : \Rightarrow P(c, k1)$

$k2 : Q(x, u1) \Rightarrow P(x, k2(u1))$

LP-Struct:  $P(x, u) \Leftrightarrow P(c, k1) \rightarrow \emptyset$

**Question 3:** How to justify the realizability transformation?

# Use a Type System

- ▶ Girard's observation on atomic intuitionistic sequent calculus

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

- ▶  $\vdash Q$  ?
- ▶ Internalized “ $\vdash$ ” as “ $\Rightarrow$ ”

$$\frac{(\kappa : \forall \underline{x}. F) \in \Phi}{\kappa : \forall \underline{x}. F} \textit{axiom} \quad \frac{e : F}{e : \forall \underline{x}. F} \textit{gen}$$

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



# Some Results

- ▶ **Soundness of LP-Unif**

If  $\Phi \vdash \{A\} \rightsquigarrow_{\gamma}^* \emptyset$ , then there exists a proof  $e : \Rightarrow \gamma A$  given axioms  $\Phi$ .

- ▶ **Soundness of LP-TM**

If  $\Phi \vdash \{A\} \rightarrow^* \emptyset$ , then there exists a proof  $e : \Rightarrow A$  given axioms  $\Phi$ .

**New!** **Completeness for LP-Unif**

If there exists a proof  $e : \Rightarrow A$  given axioms  $\Phi$ , then  $\Phi \vdash \{A\} \rightsquigarrow_{\gamma}^* \emptyset$  for some  $\gamma$ .

# Realizability Transformation

## Realizability transformation $F$ on normal proofs

- ▶  $F(\kappa : A_1, \dots, A_m \Rightarrow B) :=$   
 $\kappa : A_1[y_1], \dots, A_m[y_m] \Rightarrow B[f_\kappa(y_1, \dots, y_m)]$
- ▶  $F(\lambda \underline{a}.n : A_1, \dots, A_m \Rightarrow B) :=$   
 $\lambda \underline{a}.n : A_1[y_1], \dots, A_m[y_m] \Rightarrow B[[n]_{[y/a]}]$

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

- ▶ **Preserve Provability**

$\Phi \vdash n : \underline{A} \Rightarrow B$  implies  $F(\Phi) \vdash F(n : \underline{A} \Rightarrow B)$

- ▶ **Preserve Behavior of LP-Unif**

$\Phi \vdash \{A\} \rightsquigarrow^* \emptyset$  iff  $F(\Phi) \vdash \{A[y]\} \rightsquigarrow^* \emptyset$

- ▶ **Operational Equivalence of LP-Unif and LP-Struct**

$F(\Phi) \vdash \{A[y]\} \rightsquigarrow^* \emptyset$  iff  $F(\Phi) \vdash \{A[y]\} (\rightarrow^\mu \cdot \hookrightarrow^1)^* \emptyset$ .

- ▶ **Helps to identify productive and non-overlapping programs**

## Summary and Future Work

- ▶ We define a type system to model LP-TM, LP-Unif and LP-Struct
- ▶ We formalize realizability transformation and show it preserves the proof content
- ▶ We show that LP-Unif and LP-Struct are operationally equivalent after the transformation
- ▶ Future work: towards analyzing type class inference in Haskell
- ▶ Thank you!