

# A proof-theoretic foundation for tabled higher-order logic programming

Brigitte Pientka

Department of Computer Science

Carnegie Mellon University

Pittsburgh, PA, 15217, USA

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - Characterization based on uniform proofs
  - Soundness proof
- Related and future work

# Higher-order logic programming

What means higher-order?

- Terms: (dependently) typed  $\lambda$ -calculus
- Clauses: implication, universal quantification

# Higher-order logic programming

What means higher-order?

- Terms: (dependently) typed  $\lambda$ -calculus
- Clauses: implication, universal quantification

Framework for specifying and implementing

- logical systems
- proofs about them

# Higher-order logic programming

What means higher-order?

- Terms: (dependently) typed  $\lambda$ -calculus
- Clauses: implication, universal quantification

Framework for specifying and implementing

- logical systems (safety logics, type system . . .)
- proofs about them

# Higher-order logic programming

What means higher-order?

- Terms: (dependently) typed  $\lambda$ -calculus
- Clauses: implication, universal quantification

Framework for specifying and implementing

- logical systems (safety logics, type system ...)
- proofs about them (correctness, soundness ...)

# Higher-order logic programming

What means higher-order?

- Terms: (dependently) typed  $\lambda$ -calculus
- Clauses: implication, universal quantification

Framework for specifying and implementing

- logical systems (safety logics, type system ...)
- proofs about them (correctness, soundness ...)

Languages:

- $\lambda$ Prolog[Miller91], Isabelle[Paulson86]
- *Elf* [Pfenning91]

# Proof search via logic programming

Generic proof search over logical systems

- factor effort for each particular logical system

Infinite computation leads to non-termination.

- Many specifications are not executable.

Redundant computation hampers performance.

- Sub-proofs may be repeated.
- There may be *many* ways to prove a query.



# First-order tabled computation

- Resolution with memoization [Tamaki, Sato86]
- Memoize atomic subgoals and re-use results
- Finds all possible answers to a query
- Terminates for programs in a finite domain
- Combine tabled and non-tabled execution
- Very successful: XSB system [Warren *et.al.*]

# This talk

1. Tabled higher-order logic programming
  - Term: (dependently) typed  $\lambda$ -calculus
  - Clauses: universal quantification, implication
2. High-level description based on uniform proofs
3. Soundness proof

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - Characterization based on uniform proofs
  - Soundness proof
- Related and future work

# Declarative description of subtyping

types  $\tau ::= \text{zero} \mid \text{pos} \mid \text{nat} \mid \text{bit} \mid \tau_1 \Rightarrow \tau_2 \mid \dots$

Example:  $6 = 110$       and       $110 \in \text{nat}$

# Declarative description of subtyping

types  $\tau ::= \text{zero} \mid \text{pos} \mid \text{nat} \mid \text{bit} \mid \tau_1 \Rightarrow \tau_2 \mid \dots$

Example:  $6 = 110$       and       $110 \in \text{nat}$

$\frac{}{\text{zero} \preceq \text{nat}}$       zn

$\frac{}{\text{pos} \preceq \text{nat}}$       pn

$\frac{}{\text{nat} \preceq \text{bit}}$       nb

# Declarative description of subtyping

types  $\tau ::= \text{zero} \mid \text{pos} \mid \text{nat} \mid \text{bit} \mid \tau_1 \Rightarrow \tau_2 \mid \dots$

Example:  $6 = 110$  and  $110 \in \text{nat}$

$$\frac{}{\text{zero} \preceq \text{nat}} \text{zn}$$
$$\frac{}{\text{pos} \preceq \text{nat}} \text{pn}$$
$$\frac{}{\text{nat} \preceq \text{bit}} \text{nb}$$
$$\frac{}{T \preceq T} \text{refl}$$
$$\frac{T \preceq R \quad R \preceq S}{T \preceq S} \text{tr}$$

# Typing rules for Mini-ML

expressions  $e ::= \epsilon \mid e\ 0 \mid e\ 1 \mid \text{lam } x.e \mid \text{app } e_1\ e_2$

$$\frac{\Gamma \vdash e : \tau' \quad \tau' \preceq \tau}{\Gamma \vdash e : \tau} \text{tp-sub}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \text{lam } x.e : \tau_1 \Rightarrow \tau_2} \text{tp-lam}^x$$

# Implementation of subtyping

zn: sub zero nat.

pn: sub pos nat.

nb: sub nat bit.

refl: sub T T.

tr: sub T S

<- sub T R

<- sub R S.



# Implementation of subtyping

zn: sub zero nat.

pn: sub pos nat.

nb: sub nat bit.

refl: sub T T.

tr: sub T S

<- sub T R

<- sub R S.

**Not executable!**

# Implementation of typing rules

```
tp_sub:   of E T
         <- of E T'
         <- sub T' T.
```

```
tp_lam:   of (lam  $\lambda x.E x$ ) (T1 => T2)
         <- ( $\Pi x:\text{exp}.$  of x T1 -> of (E x) T2).
```

“forall  $x:\text{exp}$ , assume of x T1  
and show of (E x) T2”

# Implementation of typing rules

```
tp_sub:   of E T
         <- of E T'
         <- sub T' T.
```

```
tp_lam:   of (lam  $\lambda x.E x$ ) (T1 => T2)
         <- ( $\Pi x:\text{exp}.$  of x T1 -> of (E x) T2).
           “forall  $x:\text{exp}$ , assume of x T1
             and show of (E x) T2”
```

Redundancy: `tp_sub` is always applicable!

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - Characterization based on uniform proofs
  - Soundness proof
- Related and future work

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - Characterization based on uniform proofs
  - Soundness proof
- Related and future work

# Tabled higher-order logic programming

- Eliminate redundant and infinite paths from proof search using a memo-table
- Table entry:  $(\Gamma \rightarrow a, \mathcal{A})$ 
  - $\Gamma$  : context of assumptions (i.e.  $x:\text{exp}, u:\text{of } x \text{ T1}$ )
  - $a$  : atomic goal (i.e. of  $(\text{lam } \lambda x. x) \text{ T}$ )
  - $\mathcal{A}$  : list of answer substitutions for all free variables in  $\Gamma$  and  $a$
- Depth-first multi-stage strategy adopted from first-order strategy [Tamaki, Sato89]

# How higher-order tabling works...

Stage 1

---

- $\rightarrow$  of  $(\text{lam } \lambda x.x) \text{ T}$

Entry	Answers

# How higher-order tabling works...

Stage 1

---

•  $\rightarrow$  of  $(\text{lam } \lambda x.x) \top$

Entry	Answers
• $\rightarrow$ of $(\text{lam } \lambda x.x) \top$	



# How higher-order tabling works...

## Stage 1

---

•  $\rightarrow$  of  $(\text{lam } \lambda x.x) T$

$\xrightarrow{\text{tp\_sub}}$  •  $\rightarrow$  of  $(\text{lam } \lambda x.x) R,$   
sub  $R T$

Entry	Answers
• $\rightarrow$ of $(\text{lam } \lambda x.x) T$	

# How higher-order tabling works...

Stage 1

---

•  $\rightarrow$  of (lam  $\lambda x.x$ ) T

$\xrightarrow{\text{tp\_sub}}$  •  $\rightarrow$  of (lam  $\lambda x.x$ ) R,  
sub R T **Suspend**

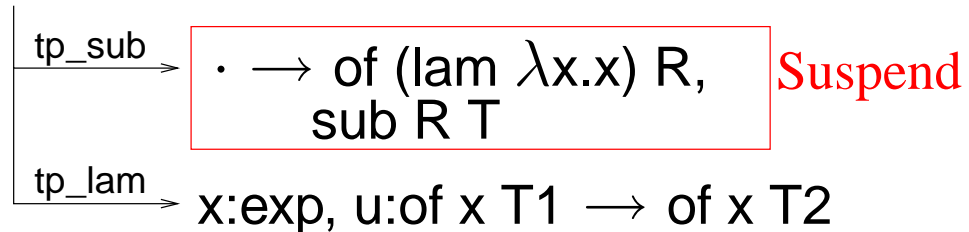
Entry	Answers
• $\rightarrow$ of (lam $\lambda x.x$ ) T	

# How higher-order tabling works...

## Stage 1

---

•  $\rightarrow$  of (lam  $\lambda x.x$ ) T



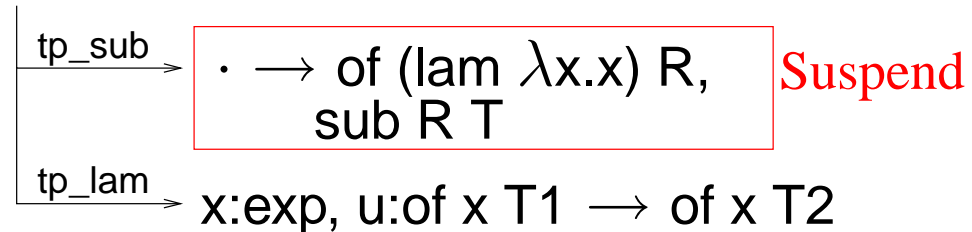
Entry	Answers
$\cdot \rightarrow$ of (lam $\lambda x.x$ ) T	

# How higher-order tabling works...

## Stage 1

---

- $\rightarrow$  of  $(\text{lam } \lambda x.x) T$

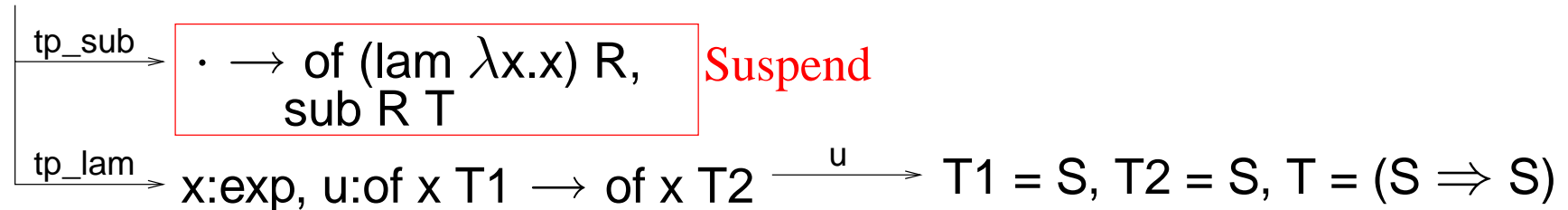


Entry	Answers
$\cdot \rightarrow$ of $(\text{lam } \lambda x.x) T$	
$x:\text{exp}, u:\text{of } x T1 \rightarrow$ of $x T2$	

# How higher-order tabling works...

## Stage 1

- $\rightarrow$  of (lam  $\lambda x.x$ ) T

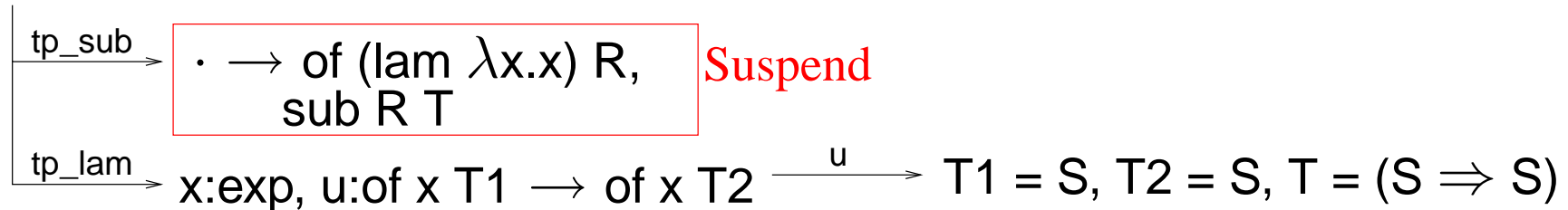


Entry	Answers
$\cdot \rightarrow$ of (lam $\lambda x.x$ ) T	
$x:\text{exp}, u:\text{of } x \text{ T1} \rightarrow \text{of } x \text{ T2}$	

# How higher-order tabling works...

## Stage 1

- $\rightarrow$  of  $(\text{lam } \lambda x.x) T$

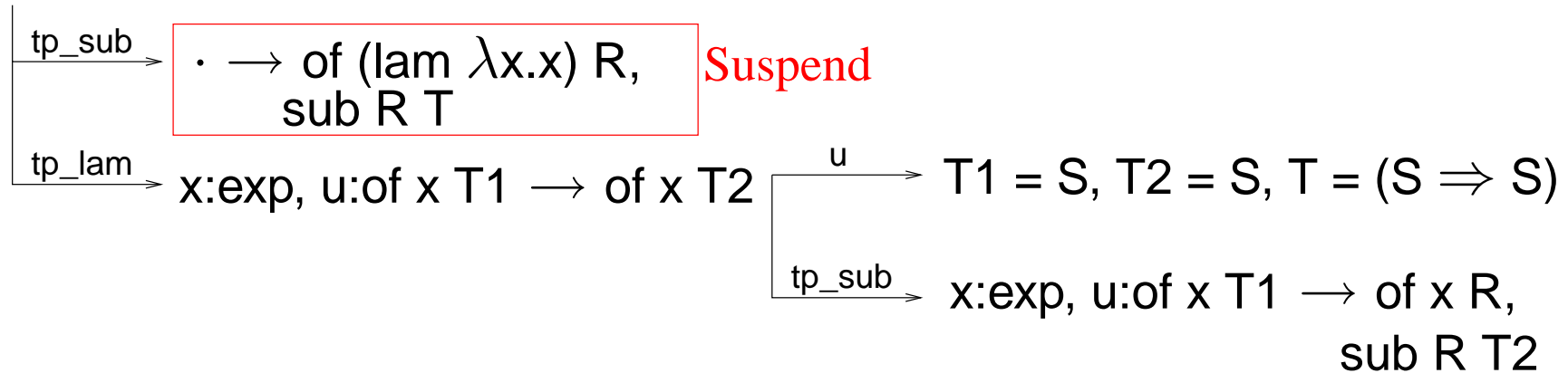


Entry	Answers
$\cdot \rightarrow$ of $(\text{lam } \lambda x.x) T$	$T = (S \Rightarrow S)$
$x:\text{exp}, u:\text{of } x T1 \rightarrow \text{of } x T2$	$T1 = S, T2 = S$

# How higher-order tabling works...

## Stage 1

- $\rightarrow$  of  $(\text{lam } \lambda x.x) T$

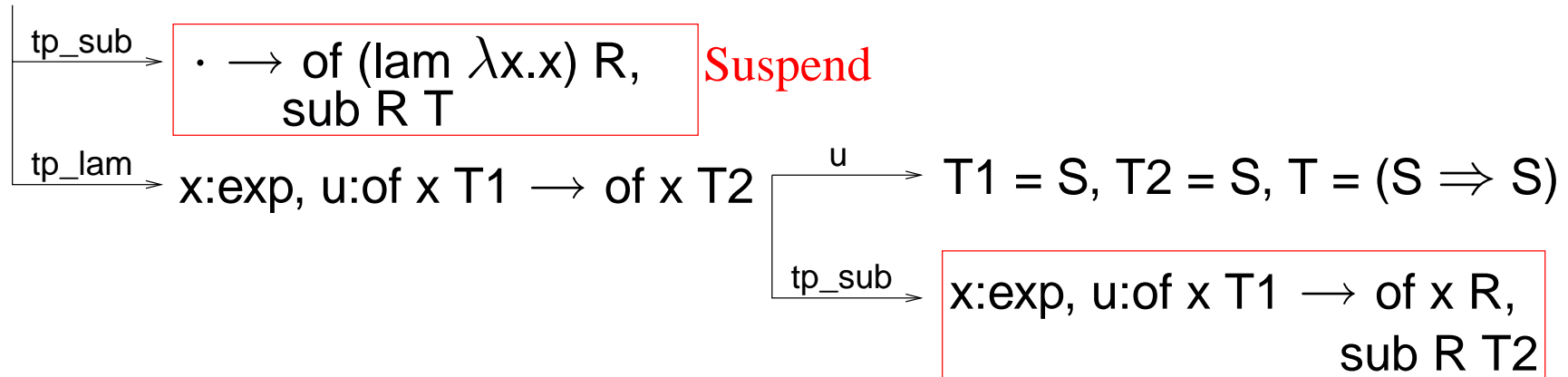


Entry	Answers
$\cdot \rightarrow \text{of } (\text{lam } \lambda x.x) T$	$T = (S \Rightarrow S)$
$x:\text{exp}, u:\text{of } x T1 \rightarrow \text{of } x T2$	$T1 = S, T2 = S$

# How higher-order tabling works...

## Stage 1

•  $\rightarrow$  of  $(\text{lam } \lambda x.x) T$



Entry	Answers
$\cdot \rightarrow$ of $(\text{lam } \lambda x.x) T$	$T = (S \Rightarrow S)$
$x:\text{exp}, u:\text{of } x T1 \rightarrow$ of $x T2$	$T1 = S, T2 = S$

Suspend

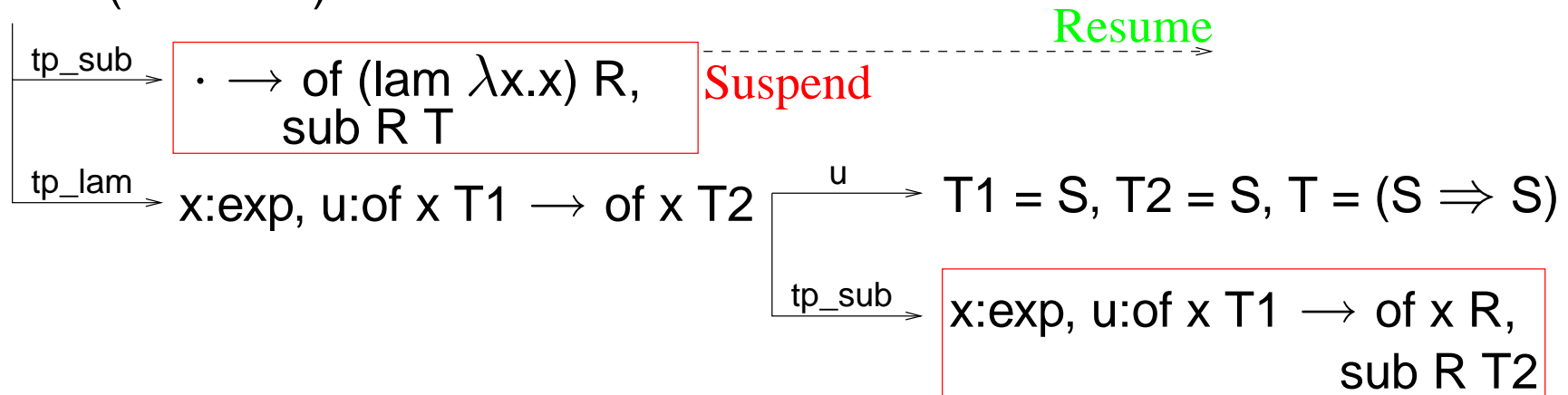
**Stage 1 finished**



# How higher-order tabling works...

## Stage 1

•  $\rightarrow$  of  $(\text{lam } \lambda x.x) T$



Entry	Answers
$\cdot \rightarrow$ of $(\text{lam } \lambda x.x) T$	$T = (S \Rightarrow S)$
$x:\text{exp}, u:\text{of } x T1 \rightarrow$ of $x T2$	$T1 = S, T2 = S$

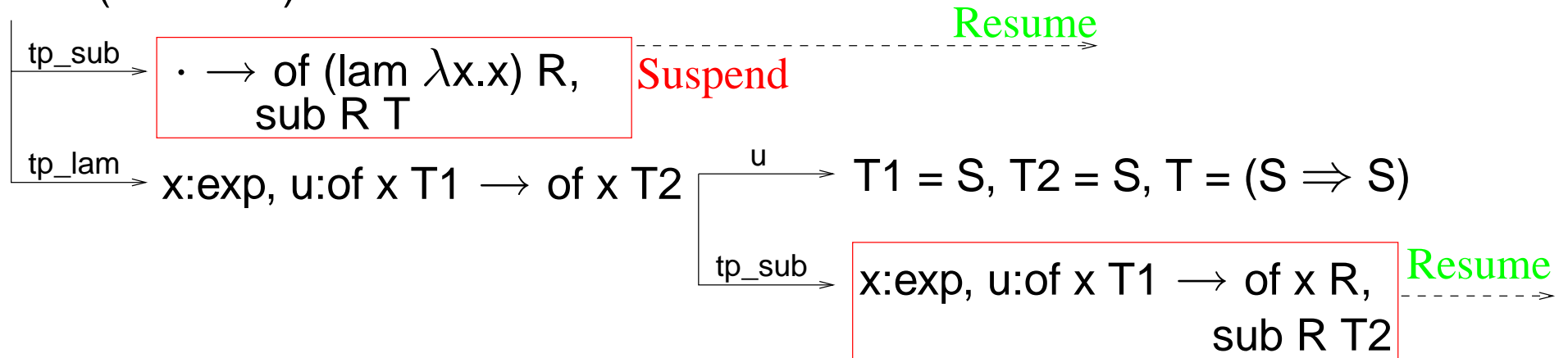
Suspend

**Stage 1 finished**

# How higher-order tabling works...

Stage 1

•  $\rightarrow$  of  $(\text{lam } \lambda x.x) T$



Entry	Answers
$\cdot \rightarrow$ of $(\text{lam } \lambda x.x) T$	$T = (S \Rightarrow S)$
$x:\text{exp}, u:\text{of } x T1 \rightarrow$ of $x T2$	$T1 = S, T2 = S$

**Suspend**

**Stage 1 finished**

# Higher-order issues

- Dependencies among propositions

$x:\text{exp}, u:\text{of } x \text{ } P \rightarrow \text{sub } P \text{ } R$

# Higher-order issues

- Dependencies among propositions

$x:\text{exp}, u:\text{of } x \ P \longrightarrow \text{sub } P \ R,$

strengthen:  $\longrightarrow \text{sub } P \ R$

# Higher-order issues

- Dependencies among propositions

$x:\text{exp}, u:\text{of } x \text{ } P \rightarrow \text{sub } P \text{ } R,$

strengthen:  $\rightarrow \text{sub } P \text{ } R$

- Dependencies among terms

$x:\text{exp}, u:\text{of } x \text{ } T1 \rightarrow \text{of } x \text{ } (R \text{ } x \text{ } u),$

# Higher-order issues

- Dependencies among propositions

$x:\text{exp}, u:\text{of } x \text{ } P \rightarrow \text{sub } P \text{ } R,$

strengthen:  $\rightarrow \text{sub } P \text{ } R$

- Dependencies among terms

$x:\text{exp}, u:\text{of } x \text{ } T1 \rightarrow \text{of } x \text{ } (R \text{ } x \text{ } u),$

strengthen  $x:\text{exp}, u:\text{of } x \text{ } T1 \rightarrow \text{of } x \text{ } R$

# Higher-order issues

- Dependencies among propositions

$x:\text{exp}, u:\text{of } x \text{ } P \rightarrow \text{sub } P \text{ } R,$

strengthen:  $\rightarrow \text{sub } P \text{ } R$

- Dependencies among terms

$x:\text{exp}, u:\text{of } x \text{ } T1 \rightarrow \text{of } x \text{ } (R \text{ } x \text{ } u),$

strengthen  $x:\text{exp}, u:\text{of } x \text{ } T1 \rightarrow \text{of } x \text{ } R$

- Subordination analysis [Virga99]

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - **Characterization based on uniform proofs**
  - Soundness proof
- Related and future work



# Types and Programs

Types  $A ::= a \mid A_1 \rightarrow A_2 \mid \Pi x : A_1. A_2$

Programs  $\Gamma ::= \cdot \mid \Gamma, x : A$

Logic programming view:

$\text{tr} : \text{sub } T \ S \leftarrow \text{sub } T \ R \leftarrow \text{sub } R \ S.$

Type-theoretic view:

$\text{tr} : \Pi T : \text{tp}. \Pi S : \text{tp}. \Pi R : \text{tp}. \text{sub } R \ S \rightarrow (\text{sub } T \ R \rightarrow \text{sub } T \ S)$

# Uniform Proofs[Miller *et al.*91]

Two judgements

$\Gamma \xrightarrow{u} A$       *uniform proof*

decompose goal  $A$  until atomic

$\Gamma \gg A \xrightarrow{f} a$       *focused proof*

pick a program clause  $A$  and  
decompose  $A$  until atomic

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1 . A_2} \text{u}\forall^x$$

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1 . A_2} \text{u}\forall^x$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u}\rightarrow^u$$

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1 . A_2} \text{u}\forall^x$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u}\rightarrow^u$$

$$\frac{\Gamma, u : A, \Gamma' \gg A \xrightarrow{f} a}{\Gamma, u : A, \Gamma' \xrightarrow{u} a} \text{uAtom}$$

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2} \text{u}\forall^x \quad \frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u} \rightarrow^u$$

$$\frac{\Gamma, u : A, \Gamma' \gg A \xrightarrow{f} a}{\Gamma, u : A, \Gamma' \xrightarrow{u} a} \text{uAtom}$$

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2} \text{u}\forall^x$$

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u}\rightarrow^u$$

$$\frac{\Gamma \gg A_1 \xrightarrow{f} a \quad \Gamma \xrightarrow{u} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a} \text{f}\rightarrow$$

$$\frac{\Gamma, u : A, \Gamma' \gg A \xrightarrow{f} a}{\Gamma, u : A, \Gamma' \xrightarrow{u} a} \text{uAtom}$$

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2} \text{u}\forall^x$$

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u}\rightarrow^u$$

$$\frac{\Gamma \gg A_1 \xrightarrow{f} a \quad \Gamma \xrightarrow{u} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a} \text{f}\rightarrow$$

$$\frac{\Gamma, u : A, \Gamma' \gg A \xrightarrow{f} a}{\Gamma, u : A, \Gamma' \xrightarrow{u} a} \text{uAtom}$$

$$\frac{}{\Gamma \gg a \xrightarrow{f} a} \text{fAtom}$$



# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2} \text{u}\forall^x$$

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u}\rightarrow^u$$

$$\frac{\Gamma \gg A_1 \xrightarrow{f} a \quad \Gamma \xrightarrow{u} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a} \text{f}\rightarrow$$

$$\frac{\Gamma, u : A, \Gamma' \gg A \xrightarrow{f} a}{\Gamma, u : A, \Gamma' \xrightarrow{u} a} \text{uAtom}$$

$$\frac{}{\Gamma \gg a \xrightarrow{f} a} \text{fAtom}$$

# Uniform Proofs

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2} \text{u}\forall^x$$

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2} \text{u}\rightarrow^u$$

$$\frac{\Gamma \gg A_1 \xrightarrow{f} a \quad \Gamma \xrightarrow{u} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a} \text{f}\rightarrow$$

$$\frac{\Gamma, u : A, \Gamma' \gg A \xrightarrow{f} a}{\Gamma, u : A, \Gamma' \xrightarrow{u} a} \text{uAtom}$$

$$\frac{}{\Gamma \gg a \xrightarrow{f} a} \text{fAtom}$$

# Computing answer substitutions

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

- Idea: replace  $M$  with an existential variable  $X$ , which is instantiated using unification

# Computing answer substitutions

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

- Idea: replace  $M$  with an existential variable  $X$ , which is instantiated using unification
- Problem
  - Higher-order unification is undecidable  
restriction to higher-order patterns  
[Miller92,Pfenning91]
  - Instantiation for  $X$  may only depend on  $\Gamma$

# Computing answer substitutions

$$\frac{\Gamma \gg [M/x]A_2 \xrightarrow{f} a \quad M \text{ has type } A_1 \text{ in } \Gamma}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a} \text{f}\forall$$

## 1. Raise $M$ [Miller92,Pfenning91]

- replace  $M$  with  $(\lambda\Gamma.M) \Gamma$
- $(\lambda\Gamma.M)$  has type  $\Pi\Gamma.A_1$

## 2. Replace $(\lambda\Gamma.M)$ with existential variable $X_{\Pi\Gamma.A_1}$

# Computing answer substitutions

$$\Gamma \gg [X_{\Pi\Gamma.A_1} \Gamma/x] A_2 \xrightarrow{f} a/\theta \quad X_{\Pi\Gamma.A_1} \text{ is new}$$

---

$$\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a/\theta$$

$$\text{Unify}(\Gamma, a', a) = \theta$$

---

$$\Gamma \gg a' \xrightarrow{f} a/\theta$$

- Annotate existential variables  $X$  with its type  $A$
- Compute answer substitution  $\theta$  as a result
- Substitution:  $\theta ::= \cdot \mid \theta, X_A = M$

# Uniform Proofs with substitutions

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2/\theta}$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2/\theta}$$

$$\frac{\Gamma, x : A, \Gamma' \gg A \xrightarrow{f} a/\theta}{\Gamma, x : A, \Gamma' \xrightarrow{u} a/\theta}$$

# Uniform Proofs with substitutions

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2/\theta} \quad \frac{\Gamma \gg [X_{\Pi\Gamma.A_1} \cdot \Gamma/x]A_2 \xrightarrow{f} a/\theta \quad X_{\Pi\Gamma.A_1} \text{ is new}}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a/\theta}$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2/\theta}$$

$$\frac{\Gamma, x : A, \Gamma' \gg A \xrightarrow{f} a/\theta}{\Gamma, x : A, \Gamma' \xrightarrow{u} a/\theta}$$



# Uniform Proofs with substitutions

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2/\theta} \quad \frac{\Gamma \gg [X_{\Pi\Gamma.A_1} \cdot \Gamma/x]A_2 \xrightarrow{f} a/\theta \quad X_{\Pi\Gamma.A_1} \text{ is new}}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a/\theta}$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2/\theta} \quad \frac{\Gamma \gg A_1 \xrightarrow{f} a/\theta_1 \quad \Gamma[\theta_1] \xrightarrow{u} A_2[\theta_1]/\theta_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a/\theta_1 \circ \theta_2}$$

$$\frac{\Gamma, x : A, \Gamma' \gg A \xrightarrow{f} a/\theta}{\Gamma, x : A, \Gamma' \xrightarrow{u} a/\theta}$$

# Uniform Proofs with substitutions

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2/\theta} \quad \frac{\Gamma \gg [X_{\Pi\Gamma.A_1} \cdot \Gamma/x]A_2 \xrightarrow{f} a/\theta \quad X_{\Pi\Gamma.A_1} \text{ is new}}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a/\theta}$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2/\theta} \quad \frac{\Gamma \gg A_1 \xrightarrow{f} a/\theta_1 \quad \Gamma[\theta_1] \xrightarrow{u} A_2[\theta_1]/\theta_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a/\theta_1 \circ \theta_2}$$

$$\frac{\Gamma, x : A, \Gamma' \gg A \xrightarrow{f} a/\theta}{\Gamma, x : A, \Gamma' \xrightarrow{u} a/\theta} \quad \frac{\text{Unify}(\Gamma, a', a) = \theta}{\Gamma \gg a' \xrightarrow{f} a/\theta}$$

# Uniform Proofs with substitutions

$$\frac{\Gamma, x : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} \Pi x : A_1.A_2/\theta} \quad \frac{\Gamma \gg [X_{\Pi\Gamma.A_1} \cdot \Gamma/x]A_2 \xrightarrow{f} a/\theta \quad X_{\Pi\Gamma.A_1} \text{ is new}}{\Gamma \gg \Pi x : A_1.A_2 \xrightarrow{f} a/\theta}$$

$$\frac{\Gamma, u : A_1 \xrightarrow{u} A_2/\theta}{\Gamma \xrightarrow{u} A_1 \rightarrow A_2/\theta} \quad \frac{\Gamma \gg A_1 \xrightarrow{f} a/\theta_1 \quad \Gamma[\theta_1] \xrightarrow{u} A_2[\theta_1]/\theta_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{f} a/\theta_1 \circ \theta_2}$$

$$\frac{\Gamma, x : A, \Gamma' \gg A \xrightarrow{f} a/\theta}{\Gamma, x : A, \Gamma' \xrightarrow{u} a/\theta} \quad \frac{\text{Unify}(\Gamma, a', a) = \theta}{\Gamma \gg a' \xrightarrow{f} a/\theta}$$

# Uniform Proofs with Tables

- Table  $\mathcal{T}$  to store conjectures and their answers
- Main judgments:
  1.  $\mathcal{T}; \Gamma \xrightarrow{u} A / (\theta, \mathcal{T}')$
  2.  $\mathcal{T}; \Gamma \gg A \xrightarrow{f} a / (\theta, \mathcal{T}')$ .
- To prove:  $\mathcal{T}; (\Gamma, x : A) \xrightarrow{u} a / (\theta, \mathcal{T}')$ 
  - Pick program clause  $A$  from  $\Gamma$
  - Retrieve answers from  $\mathcal{T}$ , if there are any

# Operations

**extend** add  $\Gamma \xrightarrow{u} a$  to  $\mathcal{T}$ ,  
if it is not already in  $\mathcal{T}$

**insert** insert answer substitution  $\theta$  to  $\mathcal{A}$  of  $\Gamma \xrightarrow{u} a$ ,  
if  $\theta$  is not already in  $\mathcal{A}$ .

**retrieve** : retrieve an answer substitution  $\theta$  for  $\Gamma \xrightarrow{u} a$   
from its answer list  $\mathcal{A}$  in  $\mathcal{T}$

# Extensions

$$\frac{\begin{array}{l} \text{extend}(\mathcal{T}, (\Gamma, u : A, \Gamma') \xrightarrow{u} a) = \mathcal{T}_1 \\ \mathcal{T}_1; (\Gamma, u : A, \Gamma') \ggg A \xrightarrow{f} a / (\theta, \mathcal{T}_2) \\ \text{insert}(\mathcal{T}_2, (\Gamma, u : A, \Gamma') \xrightarrow{u} a, \theta) = \mathcal{T}_3 \end{array}}{\mathcal{T}; (\Gamma, u : A, \Gamma') \xrightarrow{u} a / (\theta, \mathcal{T}_3)} \text{extend}$$

$$\frac{\text{retrieve}(\mathcal{T}; \Gamma \xrightarrow{u} a) = \theta}{\mathcal{T}; \Gamma \xrightarrow{u} a / (\theta, \mathcal{T})} \text{retrieve}$$

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - Characterization based on uniform proofs
  - **Soundness proof**
- Related and future work

# Main results

**Soundness** Any uniform proof *with answer substitution* has a uniform proof.

**Completeness** Any uniform proofs has a uniform proofs *with answer substitution*.

**Soundness** Any *tabled* uniform proof with an answer substitution has a uniform proof with the same answer substitution.



# Contributions

- Tabled higher-order logic programming  
Memoize and retrieve goals together with context
- High-level description of tabling  
based on uniform proofs
- Soundness of higher-order tabled search
- Implementation of a prototype
  - Tabling offers a more efficient and flexible proof search engine (see experiments [Pientka02])

# Outline

- What is higher-order logic programming?
- Example: Type-system using subtyping
- Tabled higher-order logic programming
  - How higher-order tabling works
  - Characterization based on uniform proofs
  - Soundness proof
- Related and future work

# Related work

- Tabled search is incomplete:
  - With tabelling we find only one proof for  $\Gamma \xrightarrow{u} A$
  - Proof irrelevance[Pfenning01]: all proofs for  $\Gamma \xrightarrow{u} a$  are considered equivalent
- Other higher-order logic programming languages:
  - $\lambda$ Prolog[Miller91]
  - Isabelle[Paulson86]

# Future work

- Implementation issues:
  - Higher-order indexing
  - Different tabled search strategies
- Apply labelling to linear logic programming:
  - Lolli[Miller,Hodas91]
  - LLF[Cervesato,Pfenning96]

# Finally ...

Acknowledgements: Frank Pfenning

if you want to find out more:

<http://www.cs.cmu.edu/~bp>  
email: [bp@cs.cmu.edu](mailto:bp@cs.cmu.edu)