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 λ-calculus
- Clauses: implication, universal quantification

Higher-order logic programming

What means higher-order?

- Terms: (dependently) typed λ-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 λ-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 λ-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 λ-calculus
- Clauses: implication, universal quantification

Framework for specifying and implementing

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

Languages:

- λProlog[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 λ-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 τ ::= zero | pos | nat | bit | $\tau_1 \Rightarrow \tau_2$ | $\ldots$

Example: $6 = 110$ and $110 \in$ nat

Declarative description of subtyping

types τ ::= zero | pos | nat | bit | $\tau_1 \Rightarrow \tau_2$ | $\ldots$

Example: $6 = 110$ and $110 \in$ nat

$$\frac{}{\text{zero} \preceq \text{nat}}\ \text{zn} \qquad \frac{}{\text{pos} \preceq \text{nat}}\ \text{pn} \qquad \frac{}{\text{nat} \preceq \text{bit}}\ \text{nb}$$

Declarative description of subtyping

types τ ::= zero | pos | nat | bit | $\tau_1 \Rightarrow \tau_2$ | $\ldots$

Example: $6 = 110$ and $110 \in$ nat

$$\frac{}{\text{zero} \preceq \text{nat}}\ \text{zn} \qquad \frac{}{\text{pos} \preceq \text{nat}}\ \text{pn} \qquad \frac{}{\text{nat} \preceq \text{bit}}\ \text{nb}$$

$$\frac{}{T \preceq T}\ \text{refl} \qquad \frac{T \preceq R \qquad R \preceq S}{T \preceq S}\ \text{tr}$$

Typing rules for Mini-ML

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

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

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 λx.E x) (T1 => T2)
      <-(Πx:exp.of x T1 -> of (E x) T2).
```

“forall `x: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 λx.E x) (T1 => T2)
      <-(Πx:exp.of x T1 -> of (E x) T2).
```

"forall `x: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: ($\Gamma \rightarrow a$, $\mathcal{A}$)
  - Γ : context of assumptions (i.e. x:exp, u:of x T1)
  - a : atomic goal (i.e. of (lam λx. x) T)
  - $\mathcal{A}$: list of answer substitutions for all free variables in Γ and a

- Depth-first multi-stage strategy adopted from first-order strategy [Tamaki,Sato89]

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

Entry	Answers

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\cdot \longrightarrow$ of (lam λx.x) R,
sub R T

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\begin{array}{l}\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R,} \\ \quad \text{sub R T}\end{array}}$ Suspend

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend

$\xrightarrow{\text{tp\_lam}}$ x:exp, u:of x T1 $\longrightarrow$ of x T2

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

tp_sub $\rightarrow$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend

tp_lam $\rightarrow$ x:exp, u:of x T1 $\longrightarrow$ of x T2

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	
x:exp, u:of x T1 $\longrightarrow$ of x T2	

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend

$\xrightarrow{\text{tp\_lam}}$ x:exp, u:of x T1 $\longrightarrow$ of x T2 $\xrightarrow{\text{u}}$ T1 = S, T2 = S, T = (S $\Rightarrow$ S)

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	
x:exp, u:of x T1 $\longrightarrow$ of x T2	

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

tp_sub $\rightarrow$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend

tp_lam $\rightarrow$ x:exp, u:of x T1 $\longrightarrow$ of x T2 $\xrightarrow{u}$ T1 = S, T2 = S, T = (S $\Rightarrow$ S)

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	T = (S $\Rightarrow$ S)
x:exp, u:of x T1 $\longrightarrow$ of x T2	T1 = S, T2 = S

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend

$\xrightarrow{\text{tp\_lam}}$ x:exp, u:of x T1 $\longrightarrow$ of x T2

- $\xrightarrow{\text{u}}$ T1 = S, T2 = S, T = (S $\Rightarrow$ S)
- $\xrightarrow{\text{tp\_sub}}$ x:exp, u:of x T1 $\longrightarrow$ of x R, sub R T2

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	T = (S $\Rightarrow$ S)
x:exp, u:of x T1 $\longrightarrow$ of x T2	T1 = S, T2 = S

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend

$\xrightarrow{\text{tp\_lam}}$ x:exp, u:of x T1 $\longrightarrow$ of x T2

$\xrightarrow{\text{u}}$ T1 = S, T2 = S, T = (S $\Rightarrow$ S)

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\text{x:exp, u:of x T1} \longrightarrow \text{of x R, sub R T2}}$

Suspend

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	T = (S $\Rightarrow$ S)
x:exp, u:of x T1 $\longrightarrow$ of x T2	T1 = S, T2 = S

Stage 1 finished

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda x.x) \text{ R, sub R T}}$ Suspend - - - - - → Resume

$\xrightarrow{\text{tp\_lam}}$ x:exp, u:of x T1 $\longrightarrow$ of x T2

$\xrightarrow{\text{u}}$ T1 = S, T2 = S, T = (S $\Rightarrow$ S)

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\text{x:exp, u:of x T1} \longrightarrow \text{of x R, sub R T2}}$

Suspend

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	T = (S $\Rightarrow$ S)
x:exp, u:of x T1 $\longrightarrow$ of x T2	T1 = S, T2 = S

Stage 1 finished

How higher-order tabling works...

Stage 1

$\cdot \longrightarrow$ of (lam λx.x) T

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\cdot \longrightarrow \text{of (lam } \lambda\text{x.x) R, sub R T}}$ Suspend Resume

$\xrightarrow{\text{tp\_lam}}$ x:exp, u:of x T1 $\longrightarrow$ of x T2

$\xrightarrow{\text{u}}$ T1 = S, T2 = S, T = (S $\Rightarrow$ S)

$\xrightarrow{\text{tp\_sub}}$ $\boxed{\text{x:exp, u:of x T1} \longrightarrow \text{of x R, sub R T2}}$ Resume

Suspend

Entry	Answers
$\cdot \longrightarrow$ of (lam λx.x) T	T = (S $\Rightarrow$ S)
x:exp, u:of x T1 $\longrightarrow$ of x T2	T1 = S, T2 = S

Stage 1 finished

Higher-order issues

- Dependencies among propositions

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

Higher-order issues

- Dependencies among propositions

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

Higher-order issues

- Dependencies among propositions

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

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

- Dependencies among terms

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

Higher-order issues

- Dependencies among propositions

$$\begin{array}{ll} & \text{x:exp, u:of x P} \longrightarrow \text{sub P R,} \\ \text{strengthen:} & \longrightarrow \text{sub P R} \end{array}$$

- Dependencies among terms

$$\begin{array}{ll} & \text{x:exp, u:of x T1} \longrightarrow \text{of x (R x u),} \\ \text{strengthen} & \text{x:exp, u:of x T1} \longrightarrow \text{of x R} \end{array}$$

Higher-order issues

- Dependencies among propositions

$$
\begin{array}{lr}
 & \text{x:exp, u:of x P} \longrightarrow \text{sub P R,} \\
\text{strengthen:} & \longrightarrow \text{sub P R}
\end{array}
$$

- Dependencies among terms

$$
\begin{array}{lr}
 & \text{x:exp, u:of x T1} \longrightarrow \text{of x (R x u),} \\
\text{strengthen} & \text{x:exp, u:of x T1} \longrightarrow \text{of x R}
\end{array}
$$

- 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

$$\text{Types } A \quad ::= \quad a \mid A_1 \rightarrow A_2 \mid \Pi x : A_1.A_2$$
$$\text{Programs } \Gamma \quad ::= \quad \cdot \mid \Gamma, x : A$$

Logic programming view:

```
tr:sub T S <- sub T R <- sub R S.
```

Type-theoretic view:

tr:ΠT:tp.ΠS:tp.ΠR:tp. sub R $S \rightarrow ($sub T $R \rightarrow$ sub T $S)$

Uniform Proofs[Miller *et al.*91]

Two judgements

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

decompose goal A until atomic

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

pick a program clause A and

decompose A until atomic

Uniform Proofs

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

Uniform Proofs

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

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

Uniform Proofs

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

$$\frac{\Gamma, u : A_1 \xrightarrow{\mathsf{u}} A_2}{\Gamma \xrightarrow{\mathsf{u}} A_1 \to A_2}\ \mathsf{u} \to^u$$

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

Uniform Proofs

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

$$\frac{\Gamma, u : A_1 \xrightarrow{\mathsf{u}} A_2}{\Gamma \xrightarrow{\mathsf{u}} A_1 \to A_2}\ \mathsf{u} \to^u$$

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

Uniform Proofs

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

$$\frac{\Gamma, u : A_1 \xrightarrow{\mathsf{u}} A_2}{\Gamma \xrightarrow{\mathsf{u}} A_1 \rightarrow A_2}\,\mathsf{u} \rightarrow^u \qquad \frac{\Gamma \gg A_1 \xrightarrow{\mathsf{f}} a \qquad \Gamma \xrightarrow{\mathsf{u}} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{\mathsf{f}} a}\,\mathsf{f} \rightarrow$$

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

Uniform Proofs

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

$$\frac{\Gamma, u : A_1 \xrightarrow{\mathsf{u}} A_2}{\Gamma \xrightarrow{\mathsf{u}} A_1 \rightarrow A_2}\ \mathsf{u} \rightarrow^u \qquad \frac{\Gamma \gg A_1 \xrightarrow{\mathsf{f}} a \qquad \Gamma \xrightarrow{\mathsf{u}} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{\mathsf{f}} a}\ \mathsf{f} \rightarrow$$

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

Uniform Proofs

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

$$\frac{\Gamma, u : A_1 \xrightarrow{\mathsf{u}} A_2}{\Gamma \xrightarrow{\mathsf{u}} A_1 \to A_2}\,\mathsf{u} \to^u \qquad \frac{\Gamma \gg A_1 \xrightarrow{\mathsf{f}} a \qquad \Gamma \xrightarrow{\mathsf{u}} A_2}{\Gamma \gg A_2 \to A_1 \xrightarrow{\mathsf{f}} a}\,\mathsf{f} \to$$

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

Uniform Proofs

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

$$\frac{\Gamma, u : A_1 \xrightarrow{\textsf{u}} A_2}{\Gamma \xrightarrow{\textsf{u}} A_1 \rightarrow A_2}\ \textsf{u} \rightarrow^u \qquad \frac{\Gamma \gg A_1 \xrightarrow{\textsf{f}} a \qquad \Gamma \xrightarrow{\textsf{u}} A_2}{\Gamma \gg A_2 \rightarrow A_1 \xrightarrow{\textsf{f}} a}\ \textsf{f} \rightarrow$$

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

Computing answer substitutions

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

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

Computing answer substitutions

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

- 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 Γ

Computing answer substitutions

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

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

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

$$\frac{\mathsf{Unify}(\Gamma,\ a',\ a) = \theta}{\Gamma \gg a' \xrightarrow{\mathsf{f}} a/\theta}$$

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

Uniform Proofs with substitutions

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

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

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

Uniform Proofs with substitutions

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

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

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

Uniform Proofs with substitutions

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

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

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

Uniform Proofs with substitutions

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

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

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

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

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

$$\frac{\text{Unify}(\Gamma,\ a',\ a) = \theta}{\Gamma \gg a' \xrightarrow{\text{f}} a/\theta}$$

Uniform Proofs with substitutions

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

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

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

Uniform Proofs with Tables

- Table $\mathcal{T}$ to store conjectures and their answers
- Main judgments:

1. $\mathcal{T}; \Gamma \xrightarrow{\mathsf{u}} A/(\theta, \mathcal{T}')$
2. $\mathcal{T}; \Gamma \gg A \xrightarrow{\mathsf{f}} a/(\theta, \mathcal{T}')$.

- To prove: $\mathcal{T}; (\Gamma, x : A) \xrightarrow{\mathsf{u}} a/(\theta, \mathcal{T}')$
  - Pick program clause A from Γ
  - Retrieve answers from $\mathcal{T}$, if there are any

Operations

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

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

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

Extensions

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

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

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
  - Tabeling 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 $\Gamma \xrightarrow{\text{u}} A$
  - Proof irrelevance[Pfenning01]: all proofs for $\Gamma \xrightarrow{\text{u}} a$ are considered equivalent

- Other higher-order logic programming languages:
  - λProlog[Miller91]
  - Isabelle[Paulson86]

Future work

- Implementation issues:
  - Higher-order indexing
  - Different tabled search strategies
- Apply tabelling 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