Unification of Hypergraph λ-Terms

Alimjan Yasin and Kazunori Ueda

Waseda University, Tokyo

November 24, 2017

Overview

- Unification of higher-order terms are essential parts of logic programming languages and proof systems in which terms involve with name binding.
- We present a simple unification algorithm for unifying higher-order terms represented by hypergraphs.

Name Binding

Example

$\lambda x.M$, where λ is the binder, x is a bound variable with possible occurrences in M.

- We face variable name binding in programming languages, type systems and proof systems.
- There are many existing techniques for name binding (next slide), and each has pros and cons.
- Previously, we proposed hypergrpah λ-terms[1], which is the hypergraph representation of untyped λ-terms.

[1] Alimujiang Yasen and Kazunori Ueda: Hypergraph Representation of Lambda-Terms. In Proc. TASE 2016,

Techniques for Name Binding

- De Bruijn representation
  - Use numbers to represent variables
  - Poor readability, shifting operations in substitutions
- Higher-order abstract syntax
  - λ-calculus as meta-language
  - Implementation cost is high
- Nominal logic
  - First-order representation, equips *names* with swapping and freshness
  - Sometimes difficult to reason with
- Hypergraph based approach
  - Representation is straightforward
  - Allows intuitive definition of substitutions
  - Very close to theory

Motivation

Modeling type inferences of programming languages with terms involving name binding requires unification of terms with name binding.

System $F_{<:}$ Type Inference

Finding **T** in the following.

$$\vdash \mathit{sone} : \mathbf{T}$$

where *sone* is Church numeral one in System $F_{<:}$.

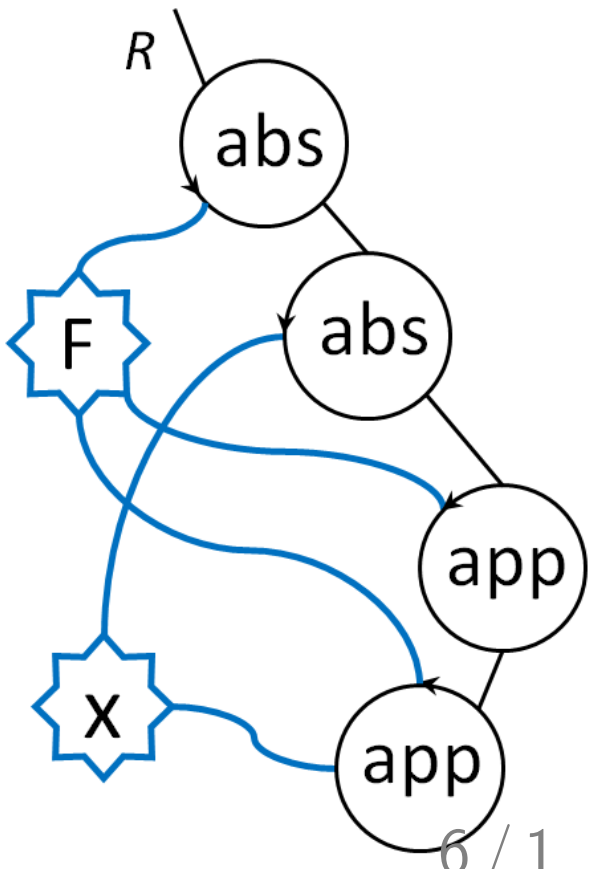
$$\mathit{sone} = \lambda X <: \mathit{top} \,.\, \lambda S <: X \,.\, \lambda Z <: X \,.\, \lambda s : X \to S \,.\, \lambda z : Z \,.\, s\, z$$

Hypergraph λ-Terms: Representation

R=abs(F,abs(X,app(F,app(F,X)))) represents $\lambda f.\lambda x.f\ (f\ x)$.

- Atoms represent constructors.
  - R=abs(X,Y) as $\lambda(x, y)$
  - R=app(M,N) as $(M\ N)$
  - R=sub(X,N,M) as $M[x := N]$
- Hyperlinks represent variables.
- Regular links represent subterm-superterm relations.
- Rewrite rules implement substitutions (subsequent slide).

These are implemented in HyperLMNtal.

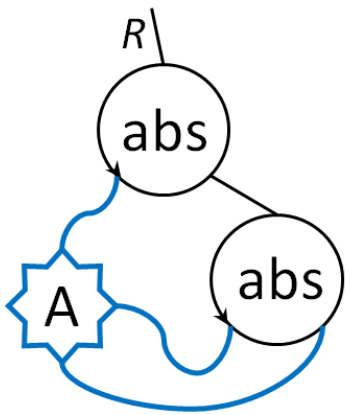


Advantage of Hypergraph Representation

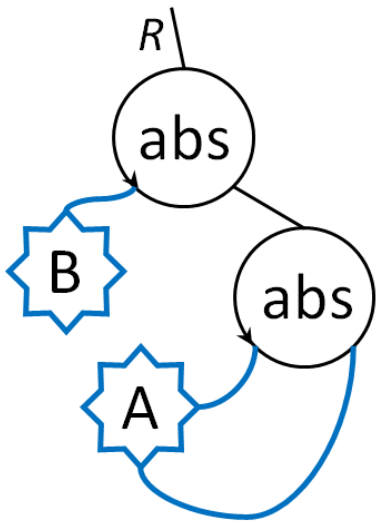
Variable Convention for Graph Representation

- Bound variables and free variables are distinct.
- All bound variables are distinct.

Example: $\lambda a.\lambda a.a$ should be written as R=abs(B,abs(A,A)).



(a) Incorrect representation of $\lambda a.\lambda a.a$



(b) Correct representation of $\lambda a.\lambda a.a$

The Encoding of λ-Calculus

HyperLMNtal implementation of λ-Calculus is as follows.

```
beta@@ R=app(abs(X,M),N)      :- R=sub(X,N,M).
var1@@ R=sub(X,N,X)           :- hlink(X) | R=N.
var2@@ R=sub(X,N,Y)           :- X\=Y, hlground(N,1) | R=Y.
abs@@  R=sub(X,N,abs(Y,M))    :- R=abs(Y,sub(X,N,M)).
app@@  R=sub(X,N,app(M1,M2)):- hlink(X), hlground(N,1) |
                               R=app(sub(X,N,M1), sub(X,N,M2)).
```

- Substitution is capture free.
- Carries no side conditions.
- Intuitive.

Unification: Examples

Example 1

$$\texttt{abs(A,}X\texttt{)} = \texttt{abs(B,B)}$$
$$X = \texttt{[A/B]B}$$
$$X = \texttt{A}$$

Unifier: $[X := \texttt{A}]$.

Note: `A` replacing X leads to **variable capture**.

Example 2

$$\texttt{abs(A,}X\texttt{)} = \texttt{abs(B,}X\texttt{)}$$
$$X = \texttt{[A/B]}X$$

Unifier: $\{\texttt{A \# }X\texttt{, B \# }X\}$.

Note: it seems we need **freshness** #.

Unification of Hypergraph λ-Terms

Unification Problem

Classic definition: For terms M and N containing unknown variables $X, Y, \ldots,$ finding $\delta{=}[X{:=}t_1, Y{:=}t_2, \ldots]$, so $\delta(\mathrm{M}){=}\delta(\mathrm{N})$.

Definition in our case: A unification problem P is a finite set consisting of equations over hypergraph λ-terms and freshness constraints, where equations, such as $M = N$, may contain $X, Y, \ldots$.

Notations

- $X, Y, \ldots,$ are unknown variables.
- A,B,C,D,... are hyperlinks
- M,N,P are hypergraph λ-terms

Unification: Swapping and Freshness

- Swapping: $[A \leftrightarrow B]M$ means swap hyperlinks A and B in term M.
- Freshness: $A\#M$ means hyperlink A does not occur in M.

Note

Swapping and freshness are borrowed from nominal logic[1].

[1] Andrew M. Pitts: A First Order Theory of Names and Bindings. Information and Computation, **186**, pp.165–193, 2003.

Unification: A lemma

Lemma

For two α-equivalent hypergraph λ-terms

$$\mathtt{abs}(A, M) = \mathtt{abs}(B, N) \ ,$$

the following holds,

- $A \# N$ and $B \# M$,
- $M = [A \leftrightarrow B]N$ and $[A \leftrightarrow B]M = N$.

The lemma follows from variable convention for graph representation.

Unification: Inductive Definition of Swappings

$\pi \cdot M$ means applying a list of swappings π to M

$$\pi @ [A \leftrightarrow C] \cdot B \stackrel{\text{def}}{=} \pi \cdot B \qquad (A \neq B, B \neq C)$$

$$\pi @ [A \leftrightarrow C] \cdot A \stackrel{\text{def}}{=} \pi \cdot C$$

$$\pi @ [C \leftrightarrow A] \cdot A \stackrel{\text{def}}{=} \pi \cdot C$$

$$\pi \cdot \mathtt{abs}(A, M) \stackrel{\text{def}}{=} \mathtt{abs}(A, \pi \cdot M)$$

$$\pi \cdot \mathtt{app}(M, N) \stackrel{\text{def}}{=} \mathtt{app}(\pi \cdot M, \pi \cdot N)$$

$$\pi \cdot (\pi' \cdot M) \stackrel{\text{def}}{=} \pi @ \pi' \cdot M$$

$$[] \cdot M \stackrel{\text{def}}{=} M$$

Unification: Judgments

Equality Under Freshness Environment

$$\theta \vdash M = N$$

θ is a freshness environment.
For example:

$$\{\mathtt{A}\#X, \mathtt{B}\#X\} \vdash \mathtt{abs(A},X\mathtt{)} = \mathtt{abs(B},X\mathtt{)}$$

Freshness Under Freshness Environment

$$\theta \vdash A\#M$$

For example:

$$\mathtt{A}\#X \vdash \mathtt{A}\#\mathtt{app(}X\mathtt{,B)}$$

Unification: Inductive Definition of Judgments

$$\frac{}{\theta \vdash A = A} \quad \texttt{=hlink}$$

$$\frac{\theta \vdash M = [A \leftrightarrow B] \cdot N \quad \theta \vdash A \# N \quad \theta \vdash B \# M}{\theta \vdash \mathtt{abs}(A, M) \;=\; \mathtt{abs}(B, N)} \quad \texttt{=abs}$$

$$\frac{\theta \vdash M_1 = M_2 \quad \theta \vdash N_1 = N_2}{\theta \vdash \mathtt{app}(M_1, N_1) \;=\; \mathtt{app}(M_2, N_2)} \quad \texttt{=app}$$

$$\frac{(A \# X) \in \theta \text{ for all } A \in \mathit{var}(\pi @ \pi')}{\theta \vdash \pi \cdot X = \pi' \cdot X} \quad \texttt{=susp}$$

$$\frac{A \neq B}{\theta \vdash A \# B} \quad \texttt{\#hlink}$$

$$\frac{\theta \vdash A \# N}{\theta \vdash A \# \mathtt{abs}(B, N)} \quad \texttt{\#abs}$$

$$\frac{\theta \vdash A \# M \quad \theta \vdash A \# N}{\theta \vdash A \# \mathtt{app}(M, N)} \quad \texttt{\#app}$$

$$\frac{(\pi^{-1} \cdot A \# X) \in \theta}{\theta \vdash A \# \pi \cdot X} \quad \texttt{\#susp}$$

Unification: Lemmas

Lemma

$\theta \vdash M = \pi \cdot N$ holds if only if $\theta \vdash \pi^{-1} \cdot M = N$ holds.
We can move swappings to other side of $=$.

Lemma

$\theta \vdash A \# \pi \cdot M$ holds if only if $\theta \vdash \pi^{-1} \cdot A \# M$ holds.
We can move swappings to other side of $\#$.

Unification: The Unification Problem

A unification problem P is a finite set consisting of equations over hypergraph λ-terms and freshness constraints, where equations, such as $M = N$, may contain $X, Y, \ldots$.

Unifier

A solution (θ, δ).

- θ is a set of freshness constraints
- δ is a substitution, i.e, $[X := M_1, Y := M_2, \ldots]$

Unification: Most General Unifier

$\mathcal{U}(P)$ is the set of all solutions for a problem P.

- $(\theta, \delta) \in \mathcal{U}(P)$ is a *most general unifier* if for any $(\theta', \delta') \in \mathcal{U}(P)$, there is a substitution δ'' such that $\theta' \vdash \delta''(\theta)$ and $\theta' \vdash \delta'' \circ \delta = \delta'$.
- $(\theta, \delta) \in \mathcal{U}(P)$ is idempotent if $\theta \vdash \delta \circ \delta = \delta$

Unification: The Unification Algorithm

- A unification starts with $\{M = N\}$ and ε.
- The following rules are applied in an arbitrary order.

=hln	$\{A = A\} \cup P, \delta$	$\Longrightarrow$	P, δ
=abs	$\{\mathtt{abs}(A, M) = \mathtt{abs}(B, N)\} \cup P, \delta$	$\Longrightarrow$	$\{M = [B \leftrightarrow A]N, A\#N, B\#M\} \cup P, \delta$
=app	$\{\mathtt{app}(M_1, N_1) = \mathtt{app}(M_2, N_2)\} \cup P, \delta$	$\Longrightarrow$	$\{M_1 = M_2\,,\ N_1 = N_2\} \cup P, \delta$
=rm	$\{\pi \cdot X = \pi' \cdot X\} \cup P, \delta$	$\Longrightarrow$	P, δ
=var	$\left.\begin{array}{c}\{M = \pi \cdot X\} \\ \{\pi \cdot X = M\}\end{array}\right\} \cup P, \delta$	$\Longrightarrow$	$\delta'(P), \delta' \circ \delta,$ where $\delta' = [X := \pi^{-1} \cdot M]$ provided X does not occur in M
#hln	$\{A\#B\} \cup P, \delta,$	$\Longrightarrow$	P, δ
#abs	$\{A\#\mathtt{abs}(B, N)\} \cup P, \delta$	$\Longrightarrow$	$\{A\#N\} \cup P, \delta$
#app	$\{A\#\mathtt{app}(M, N)\} \cup P, \delta$	$\Longrightarrow$	$\{A\#M, A\#N\} \cup P, \delta$
#sus	$\{A\#\pi \cdot X\} \cup P, \delta$	$\Longrightarrow$	$\{\pi^{-1} \cdot A\#X\} \cup P, \delta$

Unification: Example(1)

$\mathtt{abs(A,abs(B,}X\mathtt{))} = \mathtt{abs(C,abs(D,}X\mathtt{))}$
has a unifier $(\{\mathtt{A}\#X, \mathtt{C}\#X, \mathtt{B}\#X, \mathtt{D}\#X\}, \varepsilon)$.

$\{\mathtt{abs(A,abs(B,}X\mathtt{))} = \mathtt{abs(C,abs(D,}X\mathtt{))}\}, \varepsilon$
$\{\mathtt{abs(B,}X\mathtt{)} = [\mathtt{C} \leftrightarrow \mathtt{A}] \cdot \mathtt{abs(D,}X\mathtt{)}, \mathtt{A}\#\mathtt{abs(D,}X\mathtt{)}, \mathtt{C}\#\mathtt{abs(B,}X\mathtt{)}\}, \varepsilon$ (=abs)
$\{\mathtt{abs(B,}X\mathtt{)} = \mathtt{abs(D,[C} \leftrightarrow \mathtt{A]} \cdot X\mathtt{)}, \mathtt{A}\#\mathtt{abs(D,}X\mathtt{)}, \mathtt{C}\#\mathtt{abs(B,}X\mathtt{)}\}, \varepsilon$ (swapping)
$\{X = [\mathtt{D} \leftrightarrow \mathtt{B}, \mathtt{C} \leftrightarrow \mathtt{A}] \cdot X, \mathtt{A}\#\mathtt{abs(D,}X\mathtt{)}, \mathtt{C}\#\mathtt{abs(B,}X\mathtt{)},$ (=abs)
$\mathtt{D}\#X, \mathtt{B}\#[\mathtt{C} \leftrightarrow \mathtt{A}] \cdot X\ \}, \varepsilon$
$\{\mathtt{A}\#\mathtt{abs(D,}X\mathtt{)}, \mathtt{C}\#\mathtt{abs(B,}X\mathtt{)}, \mathtt{D}\#X, \mathtt{B}\#[\mathtt{C} \leftrightarrow \mathtt{A}] \cdot X\ \}, \varepsilon$ (=rm)
$\{\mathtt{A}\#\mathtt{D}, \mathtt{A}\#X, \mathtt{C}\#\mathtt{B}, \mathtt{C}\#X, \mathtt{D}\#X, \mathtt{B}\#[\mathtt{C} \leftrightarrow \mathtt{A}] \cdot X\ \}, \varepsilon$ (#abs)
$\{\mathtt{A}\#X, \mathtt{C}\#X, \mathtt{D}\#X, \mathtt{B}\#[\mathtt{C} \leftrightarrow \mathtt{A}] \cdot X\ \}, \varepsilon$ (#hln)
$\{\mathtt{A}\#X, \mathtt{C}\#X, \mathtt{D}\#X, \mathtt{B}\#X\ \}, \varepsilon$ (#sus)
Success

Unification: Example(2)

$\mathtt{abs(A,abs(B,app(}X\mathtt{,B)))} \quad = \quad \mathtt{abs(C,abs(D,app(D,}X\mathtt{)))}$
has no unifiers.

$\{\mathtt{abs(A,abs(B,app(}X\mathtt{,B)))} = \mathtt{abs(C,abs(D,app(D,}X\mathtt{)))}\}, \varepsilon$
$\{\mathtt{abs(B,app(}X\mathtt{,B))} = [\mathtt{C} \leftrightarrow \mathtt{A}] \cdot \mathtt{abs(D,app(D,}X\mathtt{))},$ (=abs)
$\mathtt{A\#abs(D,app(D,}X\mathtt{))}, \mathtt{C\#abs(B,app(}X\mathtt{,B))}\}, \varepsilon$
$\{\mathtt{app(}X\mathtt{,B)} = [\mathtt{D} \leftrightarrow \mathtt{B}] \cdot \mathtt{app(D,}[\mathtt{C} \leftrightarrow \mathtt{A}] \cdot X\mathtt{)},$ (=abs,#abs,#app,#hln)
$\mathtt{A\#}X\mathtt{, C\#}X\mathtt{, B\#\ app(D,}[\mathtt{C} \leftrightarrow \mathtt{A}] \cdot X\mathtt{), D\#app(}X\mathtt{,B)}\}, \varepsilon$
$\{X = \mathtt{B}, \mathtt{B} = [\mathtt{D} \leftrightarrow \mathtt{B}, \mathtt{C} \leftrightarrow \mathtt{A}] \cdot X, \mathtt{A\#}X, \mathtt{C\#}X, \mathtt{D\#}X, \mathtt{B\#}X\}, \varepsilon$ (=app,#app,#hln,#sus)
$\{\mathtt{B} = \mathtt{D}, \mathtt{B\#B}\}, [\mathtt{X} := \mathtt{B}]$ (=var,#hln)
Failure

Unification: Correctness

Theorem

For a given unification problem P, the unification algorithm either fails if P has no unifier or successfully produces an idempotent most general unifier.

It fails in the following cases.

- $A = B$ where A and B are different hyperlinks.
- $M = N$ where M and N are starting with different constructors such as $\mathtt{abs}$ and $\mathtt{app}$.
- One of M and N is a hyperlink and the other is a constructor.
- $\pi \cdot X = M$ where M is either $\mathtt{abs}(A, M_1)$ or $\mathtt{app}(M_2, N)$ with X occurring in M_1, M_2 and N.
- $A \# A$.

Unification: Implementation

- Implemented in HyperLMNtal.
- Total 52 rewrite rules.
- These rules are very similar to rules of the algorithm.
- A number of examples.

It can be found in https://gitlab.com/alimjanyasin.

Related Work

Higher-order Pattern Unification[1]

- Typed λ-calculus as meta-language
- Unifies modulo $=_{\alpha\beta_0\eta}$
- Substitution is capture free
- Requires higher-order patterns

Nominal Unification[2]

- First-order nominal terms as meta-language
- Unifies modulo $=_{\alpha}$
- Substitution allows variable capture
- No requirement for patterns or variables

[1] Dale Miller: A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification. J. Logic and Comput, **1**, 497–536 (1991).

[2] C. Urban, A.M. Pitts, M.J. Gabbay: Nominal unification. J. Theoretical Computer Science, **323**(1–3), 473-497 (2004).

Conclusion

Our Algorithm

- First-order graph terms as meta-language
- Unifies modulo $=_\alpha$
- Substitution allows variable capture only during unification
- Requires distinct bound and free variables

Features

- Close to nominal unification
  - Use first-order terms, only consider α-equality and allows variable capture.
- Our proofs are much simpler
  - We easily proved lemmas and equivalence relation.

Future Work

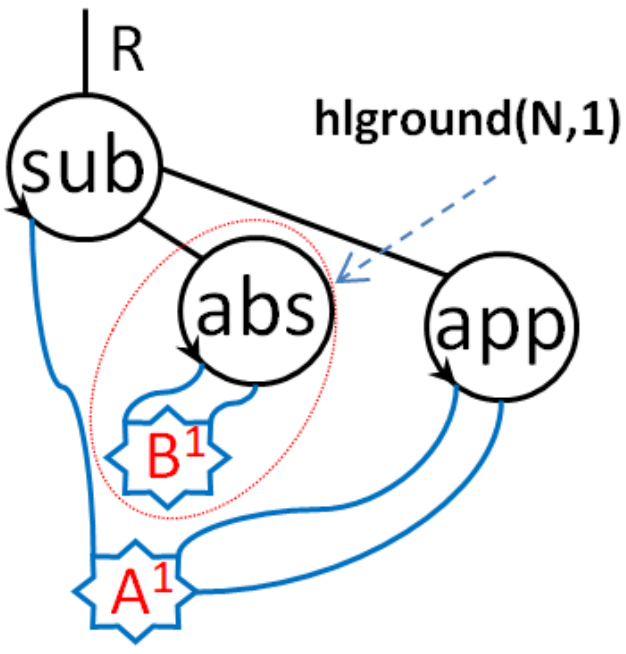
HyperLMNtal Implementations in Future

- System $F_{<:}$ type inference
- αProlog

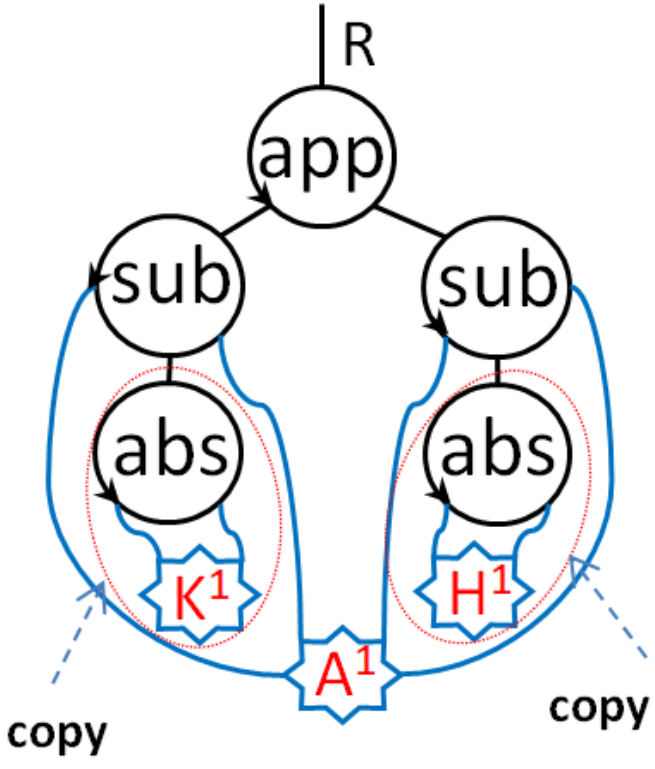
Thanks for your attention!

Hypergraph λ-terms: An example

- The rule app rewrites R=sub(A,abs(B,B),app(A,A)) into R=app(sub(A,abs(K,K),A),sub(A,abs(H,H),A)).



(a)



(b)

The untyped λ-calculus

Syntax	$M, N ::= x \mid \lambda x.M \mid M\,N$		
β-reduction	$(\lambda x.M)N \rightarrow M[x := N]$		
Substitution	$x[x := N]$	$\equiv$	N
	$y[x := N]$	$\equiv$	y, if $x \neq y$
	$(\lambda y.M)[x := N]$	$\equiv$	$\lambda y.(M[x := N])$, if $x \neq y$
	$(M_1 M_2)[x := N]$	$\equiv$	$(M_1[x := N])(M_2[x := N])$

Figure: The untyped λ-calculus

- Barendregt's variable convention is assumed in substitution definition.