Type reconstruction

constraint-based, unification and a little interpreter

Massimo Nocentini
massimo.nocentini@gmail.com

Prof. Battistina Venneri

Università degli Studi di Firenze

Firenze, May 11, 2013

Road map

1. Introduction
   - Type variables and substitutions
   - Parametric polymorphism and type inference
   - Definition of solution for (Γ, t)
2. Constrait-based typing
   - Constraint set and relations
   - Constraint typing relation
   - Definition of solution for $(\Gamma, t, S, \mathcal{C})$
3. Unification
   - Algorithm
   - Properties
   - Definition of principal solution for $(\Gamma, t, S, \mathcal{C})$

Road map

1. Introduction
   - Type variables and substitutions
   - Parametric polymorphism and type inference
   - Definition of solution for (Γ, t)
2. Constrait-based typing
   - Constraint set and relations
   - Constraint typing relation
   - Definition of solution for $(\Gamma, t, S, \mathcal{C})$
3. Unification
   - Algorithm
   - Properties
   - Definition of principal solution for $(\Gamma, t, S, \mathcal{C})$

Road map

1. Introduction
   - Type variables and substitutions
   - Parametric polymorphism and type inference
   - Definition of solution for (Γ, t)
2. Constrait-based typing
   - Constraint set and relations
   - Constraint typing relation
   - Definition of solution for $(\Gamma, t, S, \mathcal{C})$
3. Unification
   - Algorithm
   - Properties
   - Definition of principal solution for $(\Gamma, t, S, \mathcal{C})$

Syntax review

Terms

$$
\begin{array}{rlr}
t ::= & x & \textit{variable} \\
& \lambda x : T.t & \textit{abstraction} \\
& t\, t & \textit{application}
\end{array}
$$

Values

$$
\begin{array}{rlr}
v ::= & \lambda x : T.t & \textit{abstraction value}
\end{array}
$$

Types

$$
\begin{array}{rlr}
T ::= & T \to T & \textit{function type}
\end{array}
$$

Contexts

$$
\begin{array}{rlr}
\Gamma ::= & \emptyset & \textit{empty context} \\
& \Gamma, x : T & \textit{variable binding}
\end{array}
$$

Evaluation rules review

Evaluation rules

$$\frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2} \qquad (E\text{-}APP1)$$

$$\frac{t_2 \to t_2'}{v_1\ t_2 \to v_1\ t_2'} \qquad (E\text{-}APP2)$$

$$(\lambda x : T_{11}.t_{12})v_2 \to [x \mapsto v_2]t_{12} \qquad (E\text{-}APPABS)$$

Typing rules review and Inversion lemma

Typing rules

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad (T\text{-}VAR)$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1 . t_2 : T_1 \to T_2} \qquad (T\text{-}ABS)$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \qquad (T\text{-}APP)$$

Inversion Lemma

- if $\Gamma \vdash x : S_1$ then $x : S_1 \in \Gamma$
- if $\Gamma \vdash \lambda x : S_1 . t_2 : S_3$ then $S_3 = S_1 \to S_2$ for some S_2 such that $\Gamma, x : S_1 \vdash t_2 : S_2$
- if $\Gamma \vdash t_1\ t_2 : S_2$ then there exists S_1 such that $\Gamma \vdash t_1 : S_1 \to S_2$ and $\Gamma \vdash t_2 : S_1$

Contents

Introduction

Up to now we've worked with terms which all depends on *explicit* type annotation, that is every λ- abstraction have to declare the *concrete* type for the variable it introduce.

It should be interesting to be "lazy" (in the fisical sense, not because we like to be some instance of a *lazy* structures ☺) and say:

> "I don't care to declare a *concrete* type for my variable now,
> suspend the decision and call this type X, I'll specify it later"

Suppose we're deeply "lazy", why not apply the previous though to *all* variables we introduce in our program? When are our terms well typed if no concrete type is declared at all?

In this lecture we'll see a method which allow us to be "deeply lazy" while been able to type our meaningful terms and discard divergent ones (`lambda x.(x x)`, remember?)

Augment type set with type variables

In order to leave unspecified the type for a variable we've to augment our language with a new type category, which we'll call *type variables*, written as $\mathcal{A}$:

Types - augmented with type variables

$T ::=$	$T \to T$	*function type*
	$\mathcal{A}$	*type variable*
$\mathcal{A} ::=$	$\{A, B, \ldots, X_{i \in \mathbb{N}}, \ldots\}$	*type variables*

Let $X \in \mathcal{A}$:

- there's no typing rule that uses the category $\mathcal{A}$
- X can represent a *basic* type (ie. `Bool`, `Nat`, ...) or another unspecified type
- X, being a type, can be used by the defined typing rules
- $X \neq Y, \forall Y \in \mathcal{A} \setminus \{X\}$, in other words, $\mathcal{A}$ is infinite and types represented by different type variables are different too

Example

$$\Gamma \vdash \lambda x : X.x : X \to X$$

$$\Gamma \vdash \lambda x : A.x : A \to A$$

$$\Gamma \vdash \lambda s : Z \to Z.\lambda z : Z.(s\,(s\,z)) : (Z \to Z) \to Z \to Z$$

Substitutions

"I'd like that type variable X in my term t to stands for `Nat` type.
Is it possible to do a *substitution*?"

Definition of substitution

A *type substitution* σ is a mapping $\sigma : \mathcal{A} \rightarrow T$

A *substitution application* σS_1 of a type substitution σ to a type S_1 is defined inductively on the structure of S_1:

$$
\begin{aligned}
X \in \mathcal{A} \wedge \exists T_1 : (X \mapsto T_1) \in \sigma &\rightarrow \sigma X = T_1 \\
X \in \mathcal{A} \wedge \forall T_1 : (X \mapsto T_1) \notin \sigma &\rightarrow \sigma X = X \\
T_1 \text{ is a concrete type } &\rightarrow \sigma T_1 = T_1 \\
T_1, T_2 \in T &\rightarrow \sigma(T_1 \rightarrow T_2) = \sigma T_1 \rightarrow \sigma T_2
\end{aligned}
$$

Typing relation is closed under substitution application

For the following it is useful to introduce two combinations (let $\mathcal{C}$ is a set of contexts):

- $\Gamma = \{x_1 : T_1, \ldots, x_n : T_n\} \in \mathcal{C} \to \sigma\Gamma = (x_1 : \sigma T_1, \ldots, x_n : \sigma T_n) \in \mathcal{C}$
- let σ, γ be two type substitutions, their composition $\sigma \circ \gamma$ is defined as follows:

$$\begin{aligned}\sigma \circ \rho = \{X \mapsto \sigma T_1 | \forall T_1 : (X \mapsto T_1) \in \rho\} \cup \\ \{X \mapsto T_1 | \forall T_1 : ((X \mapsto T_1) \in \sigma \wedge (X \mapsto T_1) \notin \rho)\}\end{aligned}$$

Theorem

Let σ be a type substitution, Γ a context, $T_1 \in T$ and t a term. Then:

$$\Gamma \vdash t : T_1 \to \sigma\Gamma \vdash \sigma t : \sigma T_1$$

Contents

Parametric Polymorphism

Let Γ be a context and t be a term containing free variables. A first question arises:

- $\forall \sigma . \exists T_1 : \sigma\Gamma \vdash \sigma t : T_1$
  Do all type sustitution σ makes the term σt typeable under assumption $\sigma\Gamma$?

  If this is the case, why not write *any* concrete type using a type variable and use a substitution to obtain the original one?

Example

Write $\lambda s : Z \to Z . \lambda z : Z . (s\,(s\,z))$ instead of $\lambda s : Nat \to Nat . \lambda z : Nat . (s\,(s\,z))$

For example, if we work with concrete type *String* just use $\sigma = \{Z \mapsto String\}$

Holding type variables abstract during type checking is called *parametric polymorphism*: type variables are used to allow a term in which they appear usable in many concrete contexts

Type inference

Let Γ be a context and t be a term containing free variables. A second question arises:

- $\exists \sigma . \exists T_1 : \sigma\Gamma \vdash \sigma t : T_1$

  Is it always possible to find a type sustitution σ such that the term σt is typeable under assumption $\sigma\Gamma$?

  If this is the case, suppose $\nexists T_1 . \Gamma \vdash t : T_1$, using type substitution σ we're able to give a type T_2 to σt, formally $\sigma\Gamma \vdash \sigma t : T_2$

Example

$\lambda s : S . \lambda z : Z . (s\,(s\,z))$ has no type, $\forall S, Z \in \mathcal{A}$

but it has type if we use $\sigma = \{S \mapsto Nat \to Nat, Z \mapsto Nat\}$ or $\sigma = \{S \mapsto Z \to Z\}$

Looking for valid "instantiations" of type variables is called *type inference*: the compiler fill in type information wherever the user don't specify them

Contents

Let Γ be a context and t be a term containing free variables.

Definition of *solution* for (Γ, t)

Let σ be a type substitution and $T_1 \in T$.
A *solution* for (Γ, t) is a pair (σ, T_1) such that $\sigma\Gamma \vdash \sigma t : T_1$

Example

Let $\Gamma = \{f : X, a : Y\}$ and $t = f\, a$ then:

$$([X \mapsto Nat \to Nat, Y \mapsto Nat], Nat) \quad ([X \mapsto Y \to Z], Z)$$

are both solutions for (Γ, t).

Contents

Questions:

- During an execution of a type-checking algorithm, why not record constraints of the form $S_i = T_i$ instead to actually perform a type comparison?
- As we've seen in the last example, there exists countable solutions for a pair (Γ, t), are they related in some way?

Definition of *constraint set* and *unify* relation

A *constraint set* $\mathcal{C}$ is a set of equations $\{S_i \triangleq T_i\}_{i \in \mathbb{N}}$ such that $\mathcal{C} \subseteq T \times T$

A type substitution σ *unify* an equation $S \triangleq T$, written as $\sigma \bowtie S \triangleq T$, if $\sigma S = \sigma T$

Extending the unification relation to constraint sets, a type substitution σ *unify* a constraint set $\mathcal{C}$, written as $\sigma \bowtie \mathcal{C}$, if $\forall (S \triangleq T) \in \mathcal{C} : \sigma \bowtie (S \triangleq T)$

Answers:

- The modified type checking algorithm prove that a term t has type T_1 under assumptions Γ whenever there exists a type substitution σ such that $\sigma \bowtie \mathcal{C}$
- let $\mathcal{C}$ be a constraint set for a pair (Γ, t). Two solutions (σ_1, T_1) and (σ_2, T_2) are related if $\sigma_1 \bowtie \mathcal{C}$ and $\sigma_2 \bowtie \mathcal{C}$

Let's see an example of how we collect a constraint $S \triangleq T$

Example

Suppose to have an application term $t_1\, t_2$ with $\Gamma \vdash t_1 : T_1$ and $\Gamma \vdash t_2 : T_2$

Instead of checking:

- T_1 has the form $T_2 \rightarrow T_{12}$, for some T_{12}
- $t_1\, t_2$ has type *exactly* T_{12}

We suspend a decision for the type T_{12}, abstracting it with a *fresh* type variable X, creating the constraint $T_1 \triangleq T_2 \rightarrow X$ (hence $t_1\, t_2$ has type X from now on!)

The following questions drive what follow:

- given a pair (Γ, t) there always exist a constraint set?
- suppose a constraint set exists, is it unique?
- assume the existence is enough, how is its construction defined for all cases?
- how can we use the constraint set to build a solution for (Γ, t)?

Contents

Definition of *constraint typing relation* $\Gamma \vdash t : T_1 \mid_{\mathcal{X}} \mathcal{C}$

The *constraint typing relation* where a term t has type T_1 under assumptions Γ whenever exists a type substitution σ such that $\sigma \bowtie \mathcal{C}$, written as $\Gamma \vdash t : T_1 |_{\mathcal{X}} \mathcal{C}$, is defined inductively as follow:

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T \mid_{\emptyset} \emptyset} \qquad (CT\text{-}VAR)$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2 \mid_{\mathcal{X}} \mathcal{C}}{\Gamma \vdash \lambda x : T_1 . t_2 : T_1 \to T_2 \mid_{\mathcal{X}} \mathcal{C}} \qquad (CT\text{-}ABS)$$

$$\frac{\begin{array}{rl} \text{let} & X \text{ be fresh variable} \\ \Gamma \vdash & t_1 : T_1 \mid_{\mathcal{X}_1} \mathcal{C}_1 \\ \Gamma \vdash & t_2 : T_2 \mid_{\mathcal{X}_2} \mathcal{C}_2 \\ \mathcal{C}' = & \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{T_1 \triangleq T_2 \to X\} \end{array}}{\Gamma \vdash t_1\ t_2 : X \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{X\}} \mathcal{C}'} \qquad (CT\text{-}APP)$$

The set $\mathcal{X}$ is used to track type variables introduced by applications of rule *CT-APP*

Extended example

$$\frac{}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : S_1 \mid_{\mathcal{X}} \mathcal{C}}$$

Extended example

$$\cfrac{\cfrac{}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : S_2 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \to S_2 \mid_{\mathcal{X}} \mathcal{C}}$$

Extended example

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\,z)\,(y\,z)) : S_3 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : Y \to S_3 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \to Y \to S_3 \mid_{\mathcal{X}} \mathcal{C}}$$

Extended example

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x : X, y : Y, z : Z \vdash (x\, z)\,(y\, z) : S_4 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\, z)\,(y\, z)) : Z \to S_4 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : Y \to Z \to S_4 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to S_4 \mid_{\mathcal{X}} \mathcal{C}}$$

Extended example

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x : X, z : Z \vdash x\, z : S_1 \,|_{\mathcal{X}_1}\, \mathcal{C}_1} \qquad \dfrac{}{\Gamma, y : Y, z : Z \vdash y\, z : S_2 \,|_{\mathcal{X}_2}\, \mathcal{C}_2}}{\Gamma, x : X, y : Y, z : Z \vdash (x\, z)\,(y\, z) : A \,|_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S_1 \triangleq S_2 \to A\}}}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\, z)\,(y\, z)) : Z \to A \,|_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S_1 \triangleq S_2 \to A\}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : Y \to Z \to A \,|_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S_1 \triangleq S_2 \to A\}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A \,|_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S_1 \triangleq S_2 \to A\}}
$$

Extended example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x : X \vdash x : S_3 \mid_{\mathcal{X}_3} \mathcal{C}_3} \qquad \dfrac{}{\Gamma, z : Z \vdash z : S_4 \mid_{\mathcal{X}_4} \mathcal{C}_4}}{\Gamma, x : X, z : Z \vdash x\, z : B \mid_{\mathcal{X}_3 \cup \mathcal{X}_4 \cup \{B\}} \mathcal{C}_3 \cup \mathcal{C}_4 \cup \{S_3 \triangleq S_4 \rightarrow B\}} \qquad \dfrac{}{\Gamma, y : Y, z : Z \vdash y\, z : S_2 \mid_{\mathcal{X}_2} \mathcal{C}_2}}{\Gamma, x : X, y : Y, z : Z \vdash (x\, z)\,(y\, z) : A \mid_{\mathcal{X}_3 \cup \mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A, B\}} \mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, S_3 \triangleq S_4 \rightarrow B\}}}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\, z)\,(y\, z)) : Z \rightarrow A \mid_{\mathcal{X}_3 \cup \mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A, B\}} \mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, S_3 \triangleq S_4 \rightarrow B\}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : Y \rightarrow Z \rightarrow A \mid_{\mathcal{X}_3 \cup \mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A, B\}} \mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, S_3 \triangleq S_4 \rightarrow}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \rightarrow Y \rightarrow Z \rightarrow A \mid_{\mathcal{X}_3 \cup \mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A, B\}} \mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, S_3 \triangleq S_4}$$

Extended example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{x : X \in \Gamma, x : X}}{\Gamma, x : X \vdash x : X \,|_{\emptyset}\, \emptyset} \quad \overline{\Gamma, z : Z \vdash z : S_4 \,|_{\mathcal{X}_4}\, \mathcal{C}_4}}{\Gamma, x : X, z : Z \vdash x\, z : B \,|_{\mathcal{X}_4 \cup \{B\}}\, \mathcal{C}_4 \cup \{X \triangleq S_4 \rightarrow B\}} \quad \overline{\Gamma, y : Y, z : Z \vdash y\, z : S_2 \,|_{\mathcal{X}_2}\, \mathcal{C}_2}}{\Gamma, x : X, y : Y, z : Z \vdash (x\, z)\, (y\, z) : A \,|_{\mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, X \triangleq S_4 \rightarrow B\}}}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\, z)\, (y\, z)) : Z \rightarrow A \,|_{\mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, X \triangleq S_4 \rightarrow B\}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\, z)\, (y\, z)) : Y \rightarrow Z \rightarrow A \,|_{\mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, X \triangleq S_4 \rightarrow B\}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\, (y\, z)) : X \rightarrow Y \rightarrow Z \rightarrow A \,|_{\mathcal{X}_4 \cup \mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_4 \cup \mathcal{C}_2 \cup \{B \triangleq S_2 \rightarrow A, X \triangleq S_4 \rightarrow B\}}$$

Extended example

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : X \in \Gamma, x : X}}{\Gamma, x : X \vdash x : X \,|_{\emptyset}\, \emptyset} \qquad \dfrac{\dfrac{}{z : Z \in \Gamma, z : Z}}{\Gamma, z : Z \vdash z : Z \,|_{\emptyset}\, \emptyset}}{\Gamma, x : X, z : Z \vdash x\,z : B \,|_{\{B\}}\, \{X \triangleq Z \to B\}} \qquad \dfrac{}{\Gamma, y : Y, z : Z \vdash y\,z : S_2 \,|_{\mathcal{X}_2}\, \mathcal{C}_2}}{\Gamma, x : X, y : Y, z : Z \vdash (x\,z)\,(y\,z) : A \,|_{\mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_2 \cup \{B \triangleq S_2 \to A, X \triangleq Z \to B\}}}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\,z)\,(y\,z)) : Z \to A \,|_{\mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_2 \cup \{B \triangleq S_2 \to A, X \triangleq Z \to B\}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : Y \to Z \to A \,|_{\mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_2 \cup \{B \triangleq S_2 \to A, X \triangleq Z \to B\}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \to Y \to Z \to A \,|_{\mathcal{X}_2 \cup \{A,B\}}\, \mathcal{C}_2 \cup \{B \triangleq S_2 \to A, X \triangleq Z \to B\}}
$$

Extended example

$$\dfrac{}{\Gamma, y : Y \vdash y : S_5 \,|_{\mathcal{X}_5}\, \mathcal{C}_5} \qquad \dfrac{}{\Gamma, z : Z \vdash z : S_6 \,|_{\mathcal{X}_6}\, \mathcal{C}_6}$$

$$\dfrac{}{\Gamma, x : X, z : Z \vdash x\,z : B \,|_{\{B\}}\, \{X \triangleq Z \to B\}} \qquad \Gamma, y : Y, z : Z \vdash y\,z : C \,|_{\mathcal{X}_5 \cup \mathcal{X}_6 \cup \{C\}}\, \mathcal{C}_5 \cup \mathcal{C}_6 \cup \{S_5 \triangleq S_6 \to$$

$$\Gamma, x : X, y : Y, z : Z \vdash (x\,z)\,(y\,z) : A \,|_{\mathcal{X}_5 \cup \mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_5 \cup \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, S_5 \triangleq S_6 \to C\}$$

$$\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\,z)\,(y\,z)) : Z \to A \,|_{\mathcal{X}_5 \cup \mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_5 \cup \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, S_5 \triangleq S_6 \to$$

$$\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : Y \to Z \to A \,|_{\mathcal{X}_5 \cup \mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_5 \cup \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, S_5 \triangleq S$$

$$\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \to Y \to Z \to A \,|_{\mathcal{X}_5 \cup \mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_5 \cup \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, S_5 \triangleq$$

Extended example

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\Gamma, x : X, z : Z \vdash x\, z : B \,|_{\{B\}}\, \{X \triangleq Z \to B\}}
\qquad
\dfrac{
\dfrac{\dfrac{}{y : Y \in \Gamma, y : Y}}{\Gamma, y : Y \vdash y : Y \,|_{\emptyset}\, \emptyset}
\quad
\dfrac{}{\Gamma, z : Z \vdash z : S_6 \,|_{\mathcal{X}_6}\, \mathcal{C}_6}
}{\Gamma, y : Y, z : Z \vdash y\, z : C \,|_{\mathcal{X}_6 \cup \{C\}}\, \mathcal{C}_6 \cup \{Y \triangleq S_6 \to C\}}
}{\Gamma, x : X, y : Y, z : Z \vdash (x\, z)\,(y\, z) : A \,|_{\mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq S_6 \to C\}}
}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\, z)\,(y\, z)) : Z \to A \,|_{\mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq S_6 \to C\}}
}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : Y \to Z \to A \,|_{\mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq S_6 \to C\}}
}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A \,|_{\mathcal{X}_6 \cup \{A,B,C\}}\, \mathcal{C}_6 \cup \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq S_6 \to C}
$$

Extended example

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\Gamma, x : X, z : Z \vdash x\, z : B\,|_{\{B\}}\ \{X \triangleq Z \to B\}}
\qquad
\dfrac{
\dfrac{\dfrac{}{y : Y \in \Gamma, y : Y}}{\Gamma, y : Y \vdash y : Y\,|_{\emptyset}\ \emptyset}
\qquad
\dfrac{\dfrac{}{z : Z \in \Gamma, z : Z}}{\Gamma, z : Z \vdash z : Z\,|_{\emptyset}\ \emptyset}
}{\Gamma, y : Y, z : Z \vdash y\, z : C\,|_{\{C\}}\ \{Y \triangleq Z \to C\}}
}{\Gamma, x : X, y : Y, z : Z \vdash (x\, z)\,(y\, z) : A\,|_{\{A,B,C\}}\ \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}}
}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\, z)\,(y\, z)) : Z \to A\,|_{\{A,B,C\}}\ \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}}
}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : Y \to Z \to A\,|_{\{A,B,C\}}\ \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}}
}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A\,|_{\{A,B,C\}}\ \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}}
$$

Extended example

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{\Gamma, x : X, z : Z \vdash x\,z : B\,|_{\{B\}}\ \{X \triangleq Z \rightarrow B\}} \qquad \overline{\Gamma, y : Y, z : Z \vdash y\,z : C\,|_{\{C\}}\ \{Y \triangleq Z \rightarrow C\}}}{\Gamma, x : X, y : Y, z : Z \vdash (x\,z)\,(y\,z) : A\,|_{\{A,B,C\}}\ \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}}}{\Gamma, x : X, y : Y \vdash \lambda z : Z.((x\,z)\,(y\,z)) : Z \rightarrow A\,|_{\{A,B,C\}}\ \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}}}{\Gamma, x : X \vdash \lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : Y \rightarrow Z \rightarrow A\,|_{\{A,B,C\}}\ \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}}}{\Gamma \vdash \lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \rightarrow Y \rightarrow Z \rightarrow A\,|_{\{A,B,C\}}\ \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}}$$

Conclusion

We conclude that the term $\lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z))$ has *abstract type* $X \rightarrow Y \rightarrow Z \rightarrow A$ if it is possible to find a type substitution σ such that $\sigma \bowtie \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$

Extended example

$$\frac{}{\Gamma \vdash \lambda x : X.(x\, x) : S_1 \mid_{\mathcal{X}} \mathcal{C}}$$

Extended example

$$\dfrac{\dfrac{}{\Gamma, x : X \vdash x\,x : S_2 \mid_{\mathcal{X}} \mathcal{C}}}{\Gamma \vdash \lambda x : X.(x\,x) : X \rightarrow S_2 \mid_{\mathcal{X}} \mathcal{C}}$$

Extended example

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : X \vdash x : S_1 \,|_{\mathcal{X}_1}\, \mathcal{C}_1} \qquad \dfrac{}{\Gamma, x : X \vdash x : S_2 \,|_{\mathcal{X}_2}\, \mathcal{C}_2}}{\Gamma, x : X \vdash x\,x : A \,|_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S_1 \triangleq S_2 \to A\}}}{\Gamma \vdash \lambda x : X.(x\,x) : X \to A \,|_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S_1 \triangleq S_2 \to A\}}$$

Extended example

$$\dfrac{\dfrac{\overline{x : X \in \Gamma, x : X}}{\Gamma, x : X \vdash x : X \,|_{\emptyset}\, \emptyset} \qquad \overline{\Gamma, x : X \vdash x : S_2 \,|_{\mathcal{X}_2}\, \mathcal{C}_2}}{\dfrac{\Gamma, x : X \vdash x\,x : A \,|_{\mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_2 \cup \{X \triangleq S_2 \to A\}}{\Gamma \vdash \lambda x : X.(x\,x) : X \to A \,|_{\mathcal{X}_2 \cup \{A\}}\, \mathcal{C}_2 \cup \{X \triangleq S_2 \to A\}}}$$

Extended example

$$
\dfrac{\dfrac{\dfrac{}{x : X \in \Gamma, x : X}}{\Gamma, x : X \vdash x : X \,|_{\emptyset}\, \emptyset} \qquad \dfrac{\dfrac{}{x : X \in \Gamma, x : X}}{\Gamma, x : X \vdash x : X \,|_{\emptyset}\, \emptyset}}{\dfrac{\Gamma, x : X \vdash x\,x : A \,|_{\{A\}}\, \{X \triangleq X \rightarrow A\}}{\Gamma \vdash \lambda x : X.(x\,x) : X \rightarrow A \,|_{\{A\}}\, \{X \triangleq X \rightarrow A\}}}
$$

Conclusion

We conclude that the term $\lambda x : X.(x\,x))$ has *abstract type* $X \rightarrow A$ if it is possible to find a type substitution σ such that $\sigma \bowtie \{X \triangleq X \rightarrow A\}$

Contents

It is helpful make the following observations:

- when a type variable X is choosen by a final rule which has some premises, then X is different from all other type variables introduced by premises' subderivations
- for any given pair (Γ, t) the rules provide a procedure to build sets $\mathcal{X}, \mathcal{C}$ and find type T_1 such that $\Gamma \vdash t : T_1 \mid_{\mathcal{X}} \mathcal{C}$
- if we consider the relation $\Gamma \vdash t : T_1 \mid_{\mathcal{X}} \mathcal{C}$ modulo the choice of fresh variables, then the constraint set $\mathcal{C}$ and type T_1 are *uniquely* determined
- in order to find solutions for (Γ, t) we use the given rules to:
  1. build the constraint set $\mathcal{C}$, that must be satisfied in order for t to have a type
  2. determine a type S possibly containing type variables (which are subjects under constraints in $\mathcal{C}$), which caracterizes the types of t in terms of these variables
  3. find a type sustitution σ such that $\sigma \bowtie \mathcal{C}$: for each such σ the type σS is a possible type for t, hence $(\sigma, \sigma S)$ is a solution for (Γ, t)
  4. if no type sustitution $\sigma \bowtie \mathcal{C}$ exists then there is no way to instantiate type variables in t in order for t to have a type.

Let Γ be a context and t be a term containing free variables.

Definition of *solution* for $(\Gamma, t, S, \mathcal{C})$

Let σ be a type substitution, $T_1 \in T$ and suppose $\Gamma \vdash t : S \mid_{\mathcal{X}} \mathcal{C}$
A *solution* for $(\Gamma, t, S, \mathcal{C})$ is a pair (σ, T_1) such that $\sigma \bowtie \mathcal{C} \wedge \sigma S = T_1$

Example

Let $t = \lambda x : X \rightarrow Y.x\,0$ then:

$$\begin{aligned} S &= (X \rightarrow Y) \rightarrow Z \\ \mathcal{C} &= \{Nat \rightarrow Z \triangleq X \rightarrow Y\} \\ \sigma &= \{X \mapsto Nat, Z \mapsto Bool, Y \mapsto Bool\} \end{aligned}$$

hence $(\sigma, (Nat \rightarrow Bool) \rightarrow Bool)$ is a solution for $(\Gamma, t, S, \mathcal{C})$.

Let Γ be a context and t be a term containing free variables

We have two different ways of instantiating type variables appearing in t to produce a typeable term. In the next definitions σ is a type sustitution and $T_1 \in T$ as usual:

Declarative approach

$\Omega = \{\omega = (\sigma, T_1) : \omega \text{ is solution of } (\Gamma, t)\}$

Algorithmic approach

$\Theta = \{\theta = (\sigma, T_1) : \theta \text{ is solution of } (\Gamma, t, S, \mathcal{C})\}$

Theorem

$\Omega = \Theta$

Contents

Unification algorithm

By pattern matching on the structure of constraint set $\mathcal{C}$ given as argument:

$$\begin{aligned}
unify(\emptyset) &= [\,] \\
unify(\{T_1 \triangleq T_2\} \cup \mathcal{C}') &= unify(\mathcal{C}') \quad \text{if } T_1 = T_2 \\
unify(\{X \triangleq T_1\} \cup \mathcal{C}') &= unify([X \mapsto T_1]\mathcal{C}') \circ [X \mapsto T_1] \quad \text{if } X \notin FV(T_1) \\
unify(\{T_1 \triangleq X\} \cup \mathcal{C}') &= unify([X \mapsto T_1]\mathcal{C}') \circ [X \mapsto T_1] \quad \text{if } X \notin FV(T_1) \\
unify(\{T_1 \to T_2 \triangleq S_1 \to S_2\} \cup \mathcal{C}') &= unify(\mathcal{C}' \cup \{T_1 \triangleq S_1, T_2 \triangleq S_2\}) \\
unify(_) &= \text{raise failure}
\end{aligned}$$

For the properties' discussion are useful the following concepts:

- A type substitution σ is *less specific* (or *more general*) than a type substitution ρ, written as $\sigma \sqsubseteq \rho$, if $\rho = \gamma \circ \sigma$, for some type substitution γ
- A *principal unifier* for a constraint set $\mathcal{C}$ is a type substitution σ such that $\sigma \bowtie \mathcal{C}$ and $\forall \rho \bowtie \mathcal{C} : \sigma \sqsubseteq \rho$

Example

Let $\rho = \{S \mapsto Nat \to Nat, Z \mapsto Nat\}$ and $\sigma = \{S \mapsto Z \to Z\}$. We have $\sigma \sqsubseteq \rho$ because $\exists \gamma = \{Z \mapsto Nat\}$ such that $\rho = \gamma \circ \sigma$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$unify(\{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\})$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$\mathit{unify}(\{B \triangleq C \rightarrow A\} \cup \{X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\})$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$\mathit{unify}([B \mapsto C \rightarrow A]\{X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}) \circ [B \mapsto C \rightarrow A]$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$\mathit{unify}(\{X \triangleq Z \rightarrow C \rightarrow A, Y \triangleq Z \rightarrow C\}) \circ [B \mapsto C \rightarrow A]$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$unify(\{X \triangleq Z \to C \to A\} \cup \{Y \triangleq Z \to C\}) \circ [B \mapsto C \to A]$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$\textit{unify}([X \mapsto Z \to C \to A]\{Y \triangleq Z \to C\}) \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$\mathit{unify}(\{Y \triangleq Z \rightarrow C\}) \circ [X \mapsto Z \rightarrow C \rightarrow A] \circ [B \mapsto C \rightarrow A]$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \rightarrow A, X \triangleq Z \rightarrow B, Y \triangleq Z \rightarrow C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$\mathit{unify}(\emptyset) \circ [Y \mapsto Z \rightarrow C] \circ [X \mapsto Z \rightarrow C \rightarrow A] \circ [B \mapsto C \rightarrow A]$$

We return to the extended example: we built the constraint set $\mathcal{C} = \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\}$, now we apply the unification in order to build a type substitution σ (hoping such that $\sigma \bowtie \mathcal{C}$)

$$[] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

Contents

Theorem

unify$(\mathcal{C})$ *halts, either by failing or by returning a type substitution* σ

Theorem

unify$(\mathcal{C}) = \sigma \rightarrow \sigma \bowtie \mathcal{C}$

Theorem

$\rho \bowtie \mathcal{C} \rightarrow$ *unify*$(\mathcal{C}) = \sigma \wedge \sigma \sqsubseteq \rho$

Contents

Let Γ be a context and t be a term containing free variables.

Definition of *principal solution* for $(\Gamma, t, S, \mathcal{C})$

A *principal solution* for $(\Gamma, t, S, \mathcal{C})$ is a solution (σ, T_1) such that for any other solution (ρ, T_2) for $(\Gamma, t, S, \mathcal{C})$ we have $\sigma \sqsubseteq \rho$.

Theorem

if $(\Gamma, t, S, \mathcal{C})$ has a solution, then it has a principal one too. The unification algorithm can be used to decide if $(\Gamma, t, S, \mathcal{C})$ has solutions and if it is the case, it compute the principal one.

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$\sigma(X \to Y \to Z \to A)$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$\sigma X \to \sigma(Y \to Z \to A)$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$\sigma X \to \sigma Y \to \sigma(Z \to A)$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$\sigma X \to \sigma Y \to \sigma Z \to \sigma A$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$(Z \to C \to A) \to \sigma Y \to \sigma Z \to \sigma A$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\, z)\,(y\, z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$(Z \to C \to A) \to (Z \to C) \to \sigma Z \to \sigma A$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$(Z \to C \to A) \to (Z \to C) \to Z \to \sigma A$$

We finish our extended example to show a principal solution. Using unification algorithm we found the type substitution:

$$\sigma = [] \circ [Y \mapsto Z \to C] \circ [X \mapsto Z \to C \to A] \circ [B \mapsto C \to A]$$

And, using the deduction for constraint typing relation, we found that:

$$\lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)) : X \to Y \to Z \to A$$

In order to find a principal solution we apply σ to the abstract type above to have a *concrete* type for the term (possibly containing free variables, heart of parametric polymorphism, remember?):

$$(Z \to C \to A) \to (Z \to C) \to Z \to A$$

Hence $(\sigma, (Z \to C \to A) \to (Z \to C) \to Z \to A), \forall A, C, Z \in T$, is the principal solution for

$$(\emptyset, \lambda x : X.\lambda y : Y.\lambda z : Z.((x\,z)\,(y\,z)), X \to Y \to Z \to A, \{B \triangleq C \to A, X \triangleq Z \to B, Y \triangleq Z \to C\})$$