Typage
Master 2: Languages et Programmation

Lecture 8

Delia Kesner, Giovanni Bernardi

IRIF

université PARIS DIDEROT PARIS 7

Questions questions questions . . .

- What is a partial order over a set S?
- What is an equivalence relation over a set S?
- What is the greatest lower bound of a set S?
- When a system of equations is in solved form?

État Projet

?

Consider the famous fixed-point combinator

$$\mathcal{Y} \equiv \lambda x.(\lambda y.x(yy))(\lambda y.x(yy))$$

[...] It is not typable [...] But $\mathcal{Y}$ has considerable practical importance [...], and a theory that excludes it seems rather over restrictive. Can we find a more generous type-theory, one that assigns a type to $\mathcal{Y}$?

Types with Intersection: An Introduction

J.R. Hindley, 1992

1979

Intuition

Is term $\lambda x.xx$ typeable in the simply typed λ-calculus ?

Intuition

Is term $\lambda x.xx$ typeable in the simply typed λ-calculus ?

simple types not enough

$$\boxed{\dfrac{\dfrac{}{x : A \vdash x : A \to B} \qquad \dfrac{}{x : A \vdash x : A}}{\dfrac{x : A \vdash xx : B}{\vdash \lambda x.xx : A \to B}}}$$

Possible approaches,

- Have a type A such that $A \approx A \to B$ involved equivalence
- Have a type C that is A <u>and</u> $A \to B$

$$\boxed{C = A \wedge (A \to B)}$$

Formally

- $M, N ::= x \mid MN \mid \lambda x.M$ — λ-calculus
- $A, B ::= a \mid \omega \mid A \to A \mid A \wedge A$ — types

Typing rules,

$$\frac{}{\Gamma, x : A \vdash x : A} \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \quad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$\frac{}{\Gamma \vdash M : \omega} \qquad \frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \wedge B}$$

$$\frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash M : A} \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash M : B}$$

$$\frac{\Gamma \vdash \lambda x.Mx : A}{\Gamma \vdash M : A}\ x \notin FV(M)$$

Examples

Let $C = (A \to B) \wedge A$ for some A, B, and $\Gamma = \{f : B \to D\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : C \vdash x : (A \to B) \wedge A}}{x : C \vdash x : A \to B} \qquad \dfrac{\dfrac{}{x : C \vdash x : (A \to B) \wedge A}}{x : C \vdash x : A}}{x : C \vdash xx : B}}{\vdash \lambda x.xx : C \to B}$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : C \vdash f : B \to D} \quad \dfrac{\vdots}{\Gamma, x : C \vdash xx : B}}{\Gamma, x : C \vdash f(xx) : D}}{\Gamma \vdash \lambda x.f(xx) : C \to D}$$

What about $\mathcal{Y}$??

Typing $\mathcal{Y}$

Intuitively

Recall $\mathcal{Y} = \lambda f.ZZ$ where $Z = \lambda x.f(xx)$

for every term F

$$\boxed{F(\mathcal{Y}F) =_\beta \mathcal{Y}F}$$

(1) Suppose F function with codomain A

(2) FM has type A whenever M “outputs” at all

$$F : \omega \to A$$

(3) As $\mathcal{Y}F$ in range of F we have $\mathcal{Y}F : A$

(4) Because of (2) and (3)

$$\mathcal{Y} : (\omega \to A) \to A$$

Typing $\mathcal{Y}$

Formally

Let $Z = \lambda x.f(xx)$ and $B = \omega \to A$ for some A

2 $$\dfrac{\dfrac{\overline{f : B, x : \omega \vdash f : \omega \to A} \quad \overline{f : B, x : \omega \vdash xx : \omega}}{f : B, x : \omega \vdash f(xx) : A}}{f : B \vdash Z : B}$$

1 $$\dfrac{\dfrac{\overline{f : B, x : B \vdash f : \omega \to A} \quad \overline{f : \omega \to A, x : \omega \to A \vdash xx : \omega}}{f : B, x : B \vdash f(xx) : A}}{f : B \vdash Z : B \to A}$$

$$\dfrac{\dfrac{\dfrac{1}{f : \omega \to A \vdash Z : (\omega \to A) \to A} \quad \dfrac{2}{f : \omega \to A \vdash Z : \omega \to A}}{f : \omega \to A \vdash ZZ : A}}{\vdash \lambda f.ZZ : (\omega \to A) \to A}$$

Properties

Uniqueness false:

$$\vdash \mathcal{Y} : \omega \qquad \vdash \mathcal{Y} : (\omega \to A) \to A$$

Theorem (Characterisation terms with NF)

A λ-term M has a NF
if and only if
$\Gamma \vdash M : A$ *for some context* Γ *and type* A*,*
neither of which contains ω*.*

Towards practice

Suppose we have a function

$$plus : int \to int \to int \wedge string \to string \to string$$

What is the type of the following function ?

$$mult\ x\ y = \ \texttt{if}\ y == 0\ \texttt{then}\ 0\ \texttt{else}\ plus\ x\ (mult\ x\ (y-1))$$

Towards practice

Suppose we have a function

$$plus : int \to int \to int \wedge string \to string \to string$$

What is the type of the following function ?

$$mult\ x\ y = \ \texttt{if}\ y == 0\ \texttt{then}\ 0\ \texttt{else}\ plus\ x\ (mult\ x\ (y-1))$$

> A compiler for a language with intersection types might even provide two different object-code sequences for the different versions of *plus* [. . .]
> – B.C. Pierce, Intersection Types and Bounded Polymorphism

$\mathbb{C}$Duce
http://www.cduce.org/

A working programming language

So why recursive types?

Recursive types are not necessary to type $\mathcal{Y}$...

So why recursive types?

Recursive types are not necessary to type $\mathcal{Y}$...

but we still have

$$\boxed{\text{Circular definitions}}$$

$$\texttt{IntList} \quad = \quad [\,] \quad | \quad \text{int} \quad : \quad \texttt{IntList} \tag{1}$$

It is convenient to have a

- finitary object A
- that satisfies equations as (1).

So why recursive types?

Recursive types are not necessary to type $\mathcal{Y}$...

but we still have

$$\boxed{\text{Circular definitions}}$$

$$\begin{array}{ccccccc} \texttt{IntList} & = & [] & | & \texttt{int} & : & \texttt{IntList} \\ \downarrow & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ F(X) & = & \{\varepsilon\} & \cup & \mathbb{Z} & \times & X \end{array} \tag{1}$$

It is convenient to have a

- finitary object A $\qquad \mu X.F(X), \nu X.F(X)$
- that satisfies equations as (1).

Pour le TP

Projet projet projet!!

implement type inference for recursive types