
Typage monomorphe à la Curry

Quelques exemples

let $x : int = 3$ **in** $x + 1$

s'écrit maintenant

let $x = 3$ **in** $x + 1$

let $x : int = 4$ **in** (**let** $y : int = x + 1$ **in** $x * y$)

s'écrit maintenant

let $x = 4$ **in** (**let** $y = x + 1$ **in** $x * y$)

Typage monomorphe

Expressions à la Curry

$M ::= x$ |
 cte |
 $\langle M, M \rangle$ |
 $M M$ |
 $\lambda x.M$ |
let $x = M$ **in** M

2

Règles du typage monomorphe à la Curry

$\Gamma \vdash x : \Gamma(x)$ $\Gamma \vdash cte : TC(cte)$

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

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B} \quad \frac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash \text{let } x = M \text{ in } N : B}$$

3

4

Propriétés ?

L'**unicité** n'est plus vraie :

$$\vdash \lambda x.x : int \rightarrow int \quad \vdash \lambda x.x : bool \rightarrow bool$$

Mais les deux fonctions sont une **instance** d'une même fonction identité qui se comporte de la même façon :

Polymorphisme

5

Types à la Curry : difficultés de l'algorithme de typage

$$Type(\Gamma, \lambda x.M) = A \rightarrow B \quad \text{s'il existe } A \text{ t.q.} \\ Type((\Gamma, x : A), M) = B$$

$$Type(\Gamma, \text{let } x = M \text{ in } N) = B \quad \text{s'il existe } A \text{ t.q.} \\ Type(\Gamma, M) = A \text{ et} \\ Type((\Gamma, x : A), N) = B$$

7

Vers un algorithme de typage

- Les substitutions
- Les unificateurs d'un système d'équations
- Algorithme d'unification d'un système d'équations
- La construction d'un système d'équations à partir d'un terme
- Algorithme de typage d'un terme à l'aide de l'unification

6

Vers un algorithme de typage

- Soit M un terme à typer. Pour chaque variable x de M on introduit une **variable de type** α_x et pour chaque sous-expression N de M on introduit une variable de type α_N .
- On introduit un tableau de **schémas de types** pour les constantes, appelé **STC**. Ainsi par exemple $STC(fst) = \alpha \times \beta \rightarrow \alpha$.

8

On associe à M un système d'équations $SE(M)$ comme suit :

M	$SE(M)$
x	$\{\alpha_M \doteq \alpha_x\}$
cte	$\{\alpha_M \doteq STC(cte)\}$
$\langle N, L \rangle$	$\{\alpha_M \doteq \alpha_N \times \alpha_L\} \cup SE(N) \cup SE(L)$
$N L$	$\{\alpha_N \doteq \alpha_L \rightarrow \alpha_M\} \cup SE(N) \cup SE(L)$
$\lambda x.N$	$\{\alpha_M \doteq \alpha_x \rightarrow \alpha_N\} \cup SE(N)$
let $x = N$ in L	$\{\alpha_M \doteq \alpha_L; \alpha_x \doteq \alpha_N\} \cup SE(N) \cup SE(L)$

9

Correction et complétude de l'algorithme de typage

Théorème : (Correction) Si σ est une solution de $SE(M)$, alors $\Delta \vdash M : \tau(\alpha_M)$, où $\Delta = \{x : \tau(\alpha_x) \mid x \in FV(M)\}$ et τ est une instance de σ .

Théorème : (Complétude) S'il existent Δ et A t.q. $\Delta \vdash M : A$, alors $SE(M)$ admet une solution.

10

Principauté de l'algorithme de typage

Théorème : (Principauté) Si M est typable, i.e., $\Delta \vdash M : A$, alors A est une instance du type principal $\sigma(\alpha_M)$, où σ est une solution principale du système $SE(M)$.