# Types

# Motivations

- Spécifier partiellement un programme
- Éviter de programmes absurdes $(1 + \mathit{true})$
- Éviter la violation de la mémoire
- Ne pas écrire un programme ayant des comportements indéfinis
- Ne pas mélanger ou confondre les valeurs $(1 + 1.2)$
- Éviter la sémantique qui dépend du modèle de la machine

# Plan

- Types monomorphes
  - À la Church
  - À la Curry
    - Unification
    - Inférence de types
- Types polymorphes
  - À la Church
  - À la Curry (inférence de types)

# Typage monomorphe à la Church

# Typage monomorphe

**Types :**

$$A ::= \mathcal{T} \mid A \times A \mid A \to A$$

**Exemple :**

$$int \to bool \qquad bool \times bool \qquad bool \to (bool \to int)$$

$$bool \times (bool \to int) \qquad (bool \to bool) \to int$$

# Expressions à la Church

$$
\begin{array}{rll}
M ::= & x & | \\
 & cte & | \\
 & \langle M, M \rangle & | \\
 & M\ M & | \\
 & \lambda x : A.M & | \\
 & \textbf{let } x : A = M \textbf{ in } M &
\end{array}
$$

# Quelques exemples

$$\textbf{let } x : \mathit{int} = 3 \textbf{ in } x + 1$$

$$\textbf{let } x : \mathit{int} = (\textbf{if } \mathit{true} \textbf{ then } 1 \textbf{ else } 2) \textbf{ in } x + 1$$

$$\textbf{let } x : \mathit{int} = 4 \textbf{ in } (\textbf{let } y : \mathit{int} = x + 1 \textbf{ in } x * y)$$

$$\textbf{let } f : \mathit{int} \to \mathit{int} = (\lambda x : \mathit{int}.x + 1) \textbf{ in } f\ (f\ x)$$

$$\mathit{fix}(\lambda \mathit{fact} : \mathit{int} \to \mathit{int}.\lambda x : \mathit{int}.\textbf{if } x \textbf{ then } 1 \textbf{ else } (x * \mathit{fact}\ (x - 1)))$$

# Règles de réduction

$$
\begin{array}{lcl}
(\lambda x : A.M)\ N & \Rightarrow & M\{x/N\} \\
\textbf{let } x : A = N \textbf{ in } M & \Rightarrow & M\{x/N\} \\
\mathit{fix}\ M & \Rightarrow & M\ (\mathit{fix}\ M) \\
\mathit{fst}\langle M, N\rangle & \Rightarrow & M \\
\mathit{snd}\langle M, N\rangle & \Rightarrow & N \\
\textbf{if } \mathit{true} \textbf{ then } M \textbf{ else } N & \Rightarrow & M \\
\textbf{if } \mathit{false} \textbf{ then } M \textbf{ else } N & \Rightarrow & N \\
\textbf{if } 0 \textbf{ then } M \textbf{ else } N & \Rightarrow & M \\
\textbf{if } n \textbf{ then } M \textbf{ else } N & \Rightarrow & N, \quad n \neq 0
\end{array}
$$

# Règles de typage monomorphe à la Church

Pour chaque *cte* il existe un type $A$, qu'on note $TC(cte) : A$. Un environnement de typage $\Gamma$ est un ensemble de la forme $x_1 : A_1, \ldots, x_n : A_n$. Dans ce cas on écrit $\Gamma(x_i)$ pour dénoter $A_i$.

$$\Gamma \vdash x_i : \Gamma(x_i) \qquad \Gamma \vdash cte : TC(cte)$$

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

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

# Exemples de dérivations

$$\dfrac{\dfrac{x : int \vdash + : int \times int \to int \qquad \dfrac{x : int \vdash x : int \qquad x : int \vdash 1 : int}{x : int \vdash \langle x, 1\rangle : int \times int}}{x : int \vdash +\langle x, 1\rangle : int}}{\emptyset \vdash \lambda x : int.\ +\langle x, 1\rangle : int \to int}$$

Soit $(*) = f : int \to int \vdash f : int \to int$

$$\dfrac{(*) \qquad \dfrac{(*) \qquad f : int \to int \vdash 3 : int}{f : int \to int \vdash f\ 3 : int}}{f : int \to int \vdash f\ (f\ 3) : int}$$

$$\frac{\emptyset \vdash \lambda x : int.\ +\ \langle x, 1\rangle : int \to int \qquad f : int \to int \vdash f\ (f\ 3) : int}{\emptyset \vdash \textbf{let}\ f : int \to int = (\lambda x : int.\ +\ \langle x, 1\rangle)\ \textbf{in}\ f\ (f\ 3) : int}$$

# Propriétés du typage

- (Unicité) : Si $\Gamma \vdash M : A$ et $\Gamma \vdash M : B$, alors $A \equiv B$
- (Affaiblissement) : Soit $\Gamma = \{x : B \mid x \in FV(M)\}$ et soit $\Gamma \subseteq \Delta$. Alors $\Gamma \vdash M : A$ ssi $\Delta \vdash M : A$.
- (Préservation) : Si $\Gamma \vdash M : A$ et $M \Rightarrow M'$, alors $\Gamma \vdash M' : A$

# Algorithme de typage

$$
\begin{array}{lll}
\mathit{Type}(\Gamma, cte) & = TC(cte) & \\
\mathit{Type}(\Gamma, x) & = A & \text{si } x : A \in \Gamma \\
\mathit{Type}(\Gamma, \lambda x : A.M) & = A \to B & \text{si } \mathit{Type}((\Gamma, x : A), M) = B \\
\mathit{Type}(\Gamma, \langle M, N \rangle) & = A \times B & \text{si } \mathit{Type}(\Gamma, M) = A \text{ et} \\
 & & \quad \mathit{Type}(\Gamma, N) = B \\
\mathit{Type}(\Gamma, M\ N) & = B & \text{si } \mathit{Type}(\Gamma, M) = A \to B \text{ et} \\
 & & \quad \mathit{Type}(\Gamma, N) = A \\
\mathit{Type}(\Gamma, \textbf{let } x : A = M \textbf{ in } N) & = B & \text{si } \mathit{Type}(\Gamma, M) = A \text{ et} \\
 & & \quad \mathit{Type}((\Gamma, x : A), N) = B \\
\mathit{Type}(\Gamma, M) & = \mathit{erreur} & \text{sinon}
\end{array}
$$

# Propriétés de l'algorithme de typage

- (Terminaison) : Pour tout terme $M$ et tout environnement $\Gamma$, l'appel $\mathit{Type}(\Gamma, M)$ termine.
- (Correction) : Si $\mathit{Type}(\Gamma, M) = A$, alors $\Gamma \vdash M : A$.
- (Complétude) : Si $\Gamma \vdash M : A$, alors $\mathit{Type}(\Gamma, M) = A$.

Autrement dit,
Si $\mathit{Type}(\Gamma, M) = \mathit{erreur}$, alors $M$ n'est pas typable dans $\Gamma$.