Types

Motivations

- Spécifier partiellement un programme
- Éviter de programmes absurdes $(1 + true)$
- 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 :

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

Exemple :

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

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

Expressions à la Church

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

Quelques exemples

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

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

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

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

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

Règles de réduction

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

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 Γ 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.

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

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

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

Exemples de dérivations

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

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

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

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

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

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

Propriétés de l'algorithme de typage

- (Terminaison): Pour tout terme M et tout environnement Γ, l'appel $Type(\Gamma, M)$ termine.

- (Correction): Si $Type(\Gamma, M) = A$, alors $\Gamma \vdash M : A$.

- (Complétude): Si $\Gamma \vdash M : A$, alors $Type(\Gamma, M) = A$.

  Autrement dit,

  Si $Type(\Gamma, M) = erreur$, alors M n'est pas typable dans Γ.

La théorie de l'unification

Les Σ-algèbres

Σ: Ensemble de symboles de fonction ayant une arité $n \in \mathbb{N}$.

$\mathcal{X}$: Ensemble de variables.

$\mathcal{T}(\mathcal{X}, \Sigma)$: Ensemble de termes sur $\mathcal{X}$ et Σ:

f est d'arité n

$$\frac{x \in \mathcal{X}}{x \in \mathcal{T}(\mathcal{X}, \Sigma)} \qquad \frac{t_1, \ldots, t_n \in \mathcal{T}(\mathcal{X}, \Sigma) \quad f/n \in \Sigma}{f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{X}, \Sigma)}$$

On écrit $Var(t)$ l'ensemble de toutes les variables de t. Un terme t est clos si $Var(t) = \emptyset$.

Les substitutions

Définition :

- Une substitution est une fonction $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$.
- Le domaine d'une substitution σ est l'ensemble $Dom(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$.
- Le codomaine d'une substitution σ est l'ensemble $Codom(\sigma) = \{Var(\sigma(x)) \mid x \in Dom(\sigma)\}$.
- Un renommage est une substitution injective σ t.q. $\sigma(x) = y$ $\forall x \in Dom(\sigma)$.
- Si le domaine d'une substitution σ est fini on note $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ si $\sigma(x_i) = t_i$ et $x_i \in Dom(\sigma)$.

- L'application d'une substitution à un terme est l'extension de σ aux termes donnée par $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.

Comparer deux substitutions

Soient σ et τ deux substitution. La composition de σ avec τ est donnée par $(\sigma \circ \tau)(x) = \sigma(\tau(x))$.

Exemple :
$\{y/b, z/h(c)\} \circ \{x/f(y), y/z\} = \{x/f(b), y/h(c), z/h(c)\}$

La substitution σ est une instance de la substitution τ (ou τ est plus générale que σ), ce que l'on écrit $\sigma \leq \tau$, ss'il existe une substitution ρ t.q. pour toute variable $x \in \mathcal{X}$, $\sigma(x) = (\rho \circ \tau)(x)$.

Exemple : $\{x/f(y), y/z\}$ est plus générale que $\{x/f(b), y/h(c), z/h(c)\}$

Identifier deux substitutions

Remarque : La relation $\leq$ n'est pas antysimmetrique.

Exemple : Soient $\sigma_1 = \{x/y\}$ et $\sigma_2 = \{y/x\}$.

On a $\sigma_1 \leq \sigma_2$ car $\sigma_1 = \{x/y\} \circ \sigma_2$.

On a $\sigma_2 \leq \sigma_1$ car $\sigma_2 = \{y/x\} \circ \sigma_1$.

Mais $\sigma_1 \neq \sigma_2$.

Exemple : Soient $\sigma_1 = \{x/y\}$ et $\sigma_3 = \{x/y, z/w, w/z\}$.

On a $\sigma_1 \leq \sigma_3$ car $\sigma_1 = \{z/w, w/z\} \circ \sigma_3$.

On a $\sigma_3 \leq \sigma_1$ car $\sigma_3 = \{z/w, w/z\} \circ \sigma_1$.

Mais $\sigma_1 \neq \sigma_3$.

Lemme : $\sigma \sim \sigma'$ ssi $\exists$ un renommage ρ t.q. $\sigma = \rho \circ \sigma'$.

Donc $\sigma_1 \sim \sigma_2 \sim \sigma_3$ dans l'exemple précédent.

Substitution(s) principale(s)

Soit $\mathcal{S}$ en ensemble de substitutions et $\tau \in \mathcal{S}$. On dit que τ est principale ssi toute substitution $\sigma \in \mathcal{S}$ est une instance de τ.

Exemple : Soit $\mathcal{S} = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$, où $\sigma_1 = \{x/y\}$, $\sigma_2 = \{y/x\}$, $\sigma_3 = \{x/y, w/z, z/w\}$, $\sigma_4 = \{x/u, y/u\}$ et $\sigma_5 = \{x/a, y/a\}$.

Alors σ_1, σ_2 et σ_3 sont principales pour $\mathcal{S}$. En effet,

$\sigma_2, \sigma_3, \sigma_4, \sigma_5 \leq \sigma_1$ et $\sigma_1, \sigma_3, \sigma_4, \sigma_5 \leq \sigma_2$ et $\sigma_1, \sigma_2, \sigma_4, \sigma_5 \leq \sigma_3$

Mais $\sigma_1 \not\leq \sigma_4$ et $\sigma_1 \not\leq \sigma_5$ (entre autres).

Unification comme solution d'un système d'équations

Deux termes A et B sont unifiables ss'il existe une substitution σ t.q. $\sigma(A) = \sigma(B)$ (σ est donc un unificateur de A et B).

Une équation est une paire de termes de la forme $A \doteq B$. On dit qu'elle est unifiable ssi les termes A et B le sont.

Un système/problème d'équations E est un ensemble d'équations. On dit qu'il est unifiable ssi il existe une substitution qui est unificateur de toutes les équations de E. Cette substitution est appelée solution de E.

On s'intéresse aux systèmes d'équations finis.

L'unicité

1. On identifie deux unificateurs σ et σ' d'un problème $\mathcal{P}$ s'ils ne diffèrent que par des renommage de variables, c'est à dire, si $\sigma \sim \sigma'$.

2. On considère uniquement comme unificateurs de $\mathcal{P}$ les substitutions σ t.q. $Dom(\sigma) \subseteq Var(\mathcal{P})$.

Exemple : Soit $\mathcal{S} = \{x \doteq y\}$. Prenons trois unificateurs principaux de $\mathcal{S}$: $\sigma_1 = \{x/y\}$,$\sigma_2 = \{y/x\}$ et $\sigma_3 = \{x/y, z/w, w/z\}$.

Alors $\sigma_1 = \sigma_2$ (car $[\sigma_1]_\sim = [\sigma_2]_\sim$) et σ_3 n'est plus considéré comme un unificateurs de $\mathcal{S}$.

L'unicité

Module ces considérations, l'unificateur pricipal d'un problème $\mathcal{P}$ est unique modulo renommage, c'est à dire :

Si σ et σ' sont deux unificateurs principaux de $\mathcal{P}$, alors $\sigma \sim \sigma'$.

Les formes résolues

Définition : Un système d'équations E est en forme résolue ssi il est de la forme $\{\alpha_1 \doteq t_1, \ldots, \alpha_n \doteq t_n\}$, où

- toutes les variables α_i sont distinctes ($i \neq j$ implique $\alpha_i \neq \alpha_j$)
- aucune α_i n'apparaît dans un t_j ($\forall i, \alpha_i \notin \bigcup_{1 \leq j \leq n} Var(t_j)$)

Notation : Si E est un système en forme résolue $\{\alpha_1 \doteq t_1, \ldots, \alpha_n \doteq t_n\}$ on note $\vec{E}$ la substitution $\{\alpha_1/t_1, \ldots, \alpha_n/t_n\}$.

Les règles de transformation

$$\frac{E \cup \{s \doteq s\}}{E} \quad \text{(effacer)} \qquad \frac{E \cup \{t \doteq \alpha\} \quad t \notin \mathcal{X}}{E \cup \{\alpha \doteq t\}} \quad \text{(orienter)}$$

$$\frac{E \cup \{f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)\}}{E \cup \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}} \quad \text{(décomposer)}$$

$$\frac{E \cup \{\alpha \doteq s\} \quad \alpha \in Var(E) \quad \alpha \notin Var(s)}{E\{\alpha/s\} \cup \{\alpha \doteq s\}} \quad \text{(remplacer)}$$

Algorithme d'unification d'un système E

1. On démarre avec un système E
2. On applique les règles de transformation tant qu'on peut, on obtient un problème P
3. Si le système P est en forme résolue
   - alors renvoyer $\vec{P}$.
   - sinon échec

Exemple

Soit $\mathcal{P} = \{f(x, h(b), c) \doteq f(g(y), y, c)\}$.

$$\cfrac{\cfrac{\cfrac{\cfrac{f(x, h(b), c) \doteq f(g(y), y, c)}{x \doteq g(y), h(b) \doteq y, c \doteq c}\,\mathsf{d}}{x \doteq g(y), h(b) \doteq y}\,\mathsf{e}}{x \doteq g(y), y \doteq h(b)}\,\mathsf{o}}{x \doteq g(h(b)), y \doteq h(b)}\,\mathsf{r}$$

L'unificateur principal de P est $\sigma = \{x/g(h(b)), y/h(b)\}$.

Ainsi, $\sigma f(x, h(b), c) = f(g(h(b)), h(b), c) = \sigma f(g(y), y, c)$.

Vers la correction et la complétude de l'algorithme

Lemme :

1. L'algorithme termine.
2. Si σ est un unificateur d'une forme résolue P, alors $\sigma = \sigma\vec{P}$.
3. Si une règle transforme un problème P dans un problème S, alors les solutions de P et S sont les mêmes.
4. Si E est en forme résolue, alors $\vec{E}$ est solution du problème E.

Correction et complétude de l'algorithme

Theorem : (Correction) Si l'algorithme trouve une substitution $\vec{S}$ pour le problème P, alors P est unifiable et $\vec{S}$ est un unificateur principal de P.

Autrement dit,

Si P n'est pas unifiable, l'algorithme échoue.

Theorem : (Complétude) Si le système P est unifiable, alors l'algorithme calcule l'unificateur principal de P.

Autrement dit,

Si l'algorithme échoue, alors le système P n'est pas unifiable.

Typage monomorphe à la Curry

Typage monomorphe

Expressions à la Curry

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

Quelques exemples

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

s'écrit maintenant

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

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

s'écrit maintenant

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

Règles du typage monomorphe à la Curry

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

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

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

Propriétés?

L'**unicité** n'est plus vraie :

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

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

Polymorphisme

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

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

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

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 $STC(fst) = \alpha \times \beta \to \alpha$.

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

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

Correction et complétude de l'algorithme de typage

Theorem : (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.

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

Principalité de l'algorithme de typage

Theorem : (Principalité) 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 $SE(M)$.