Polymorphisme

Motivations

- Décrire le comportement d'une opération de façon générale, i.e. de façon indépendante de la nature (ou type) de ses paramètres.
- Employer une même opération avec plusieurs types différents. Ceci rend le programme plus claire et plus succint.
- Faciliter la réutilisation.
- Faire porter le même nom à des opérations qui se "ressemblent" d'un point de vue sémantique.

Les différentes formes du polymorphisme

- Polymorphisme ad-hoc ou surcharge
- Polymorphisme d'inclusion ou sous-typage
- Polymorphisme paramétrique ou générique

Le polymorphisme ad-hoc

Dans les langages orientés objet $\Rightarrow$ définir des méthodes de même nom dans deux classes indépendantes.
Dans les langages procéduraux $\Rightarrow$ définir deux fonctions de même nom mais ayant des paramètres différents (type ou nombre).

Exemple : Opérateur $+$ pour réaliser l'addition entre deux nombres complexes, entiers ou réels.

Difficulté : Pour chaque utilisation de l'opérateur le compilateur doit décider quel code exécuter.

Le polymorphisme d'inclusion

Definition

Un type A' est un sous-type (ou est inclus dans) un type A si n'importe quelle opération définie sur A est aussi définie sur A'.

Intuition : Un objet de type A' peut remplacer sans danger un objet de type A.

Sous-typage entre objets :

Soient $A =< m_1 : A_1; ...; m_n : A_n >$ et
$A' =< m_1 : A'_1; ...; m_n : A'_n; m_{n+1} : C_{n+1}; ...; m_k : C_k >$. On dit que A' est un sous-type de A, noté $A' \leq A$, ssi $A'_i < A_i$ pour $i \in \{1, ..., n\}$.

Exemple : Le type colorpoint est un sous-type du type point.

Sous-typage dans les appels de fonctions :
Si $\Gamma \vdash f : A \to B$, $\Gamma \vdash M : A'$ et $A' \leq A$ alors $fM : B$.

Exemple : Le polymorphisme d'inclusion permet de construire des listes hétérogènes où chaque élément est sous-type du type des éléments de la liste. Ainsi, on peut construire une liste de points (de couleur ou pas) et appliquer une fonction sur cette liste.

Le polymorphisme paramétrique

Motivation :

Plusieurs fonctions identité :

$Id_{int} : int \to int$, $Id_{bool} : bool \to bool$,

$Id_{int \to int} : (int \to int) \to (int \to int)$, ...

Plusieurs fonctions pour rajouter un élément à une liste :

$Aj_{int} : int \times list(int) \to list(int)$,

$Aj_{bool} : bool \times list(bool) \to list(bool)$, ...

Idée :
$Id : \forall\alpha.\alpha \to \alpha$
et donc
$Id_{int} = Id[int]$
$Id_{bool} = Id[bool]$
$Id_{int \to int} = Id[int \to int]$

Remarque : Ce qui caractérise le polymorphisme paramétrique est la relation d'instantiation qui existe entre le type général de la fonction polymorphe α, et le type particulier de chaque spécialisation $int, bool, int \to int$.

Le polymorphisme à la Girard-Reynolds

Types : $A ::= \mathcal{T} \mid \alpha \mid A \times A \mid A \to A \mid \forall\alpha.A$

Notation : $\forall\alpha\forall\beta.A = \forall(\alpha,\beta).A.$

Expressions :

$$\begin{array}{rl} M ::= & x \mid cte \mid \langle M, M\rangle \mid M\ M \mid \\ & \lambda x : A.M \mid \textbf{let}\ x : A = M\ \textbf{in}\ M \mid \\ & M[A] \mid \Lambda\alpha.M \end{array}$$

Règles de réduction

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

Exemple

$Id = \Lambda\alpha.\lambda x : \alpha.x$

$$\begin{array}{lcl} (Id[int]) & \Rightarrow & \lambda x : int.x \\ (Id[int])\ 3 & \Rightarrow & (\lambda x : int.x)\ 3 \Rightarrow 3 \end{array}$$

Definition

Les variables libres d'un type sont définies par :

$$
\begin{array}{lcl}
\mathit{VarLib}(\mathcal{T}) & = & \emptyset \\
\mathit{VarLib}(\alpha) & = & \{\alpha\} \\
\mathit{VarLib}(\forall\alpha.A) & = & \mathit{VarLib}(A) \setminus \{\alpha\} \\
\mathit{VarLib}(A \times B) & = & \mathit{VarLib}(A) \cup \mathit{VarLib}(B) \\
\mathit{VarLib}(A \to B) & = & \mathit{VarLib}(A) \cup \mathit{VarLib}(B)
\end{array}
$$

Un type A est clos ssi $\mathit{VarLib}(A) = \emptyset$.

Exemple : $\mathit{VarLib}(\beta \to (\forall\alpha.\alpha \times \gamma)) = \{\beta, \gamma\}$ et $\forall\alpha\forall\beta.(\alpha \to \beta)$ est clos.

Règles du typage polymorphe à la Church

L'ensemble de variables libres d'un environnement est donné par $\mathit{VarLib}(\Gamma) = \bigcup_{x \in \Gamma} \mathit{VarLib}(\Gamma(x))$.
Pour obtenir les règles du typage polymorphe à la Church on rajoute aux règles monomorphes :

$$\frac{\Gamma \vdash M : A \quad \alpha \notin \mathit{VarLib}(\Gamma)}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.A} \qquad \frac{\Gamma \vdash M : \forall\alpha.A}{\Gamma \vdash M[B] : A\{\alpha/B\}}$$

Exemple :

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \vdash x : \alpha}{\vdash \lambda x : \alpha . x : \alpha \to \alpha}}{\vdash \Lambda\alpha.\lambda x : \alpha . x : \forall\alpha.(\alpha \to \alpha)}}{\vdash (\Lambda\alpha.\lambda x : \alpha . x)[\mathit{int}] : \mathit{int} \to \mathit{int}} \qquad 3 : \mathit{int}}{\vdash ((\Lambda\alpha.\lambda x : \alpha . x)[\mathit{int}])\ 3 : \mathit{int}}$$

Algorithme de typage

Étant donné un terme M, le problème de l'inférence de types consiste à savoir si $\exists \Gamma$ et $\exists A$ t.q. $\Gamma \vdash M : A$.

Théorème : (Schubert) Le problème de l'inférence de types dans le système de Girard est indécidable.

Idée : Réduire la grammaire de types $\rightarrow$ polymorphisme à la ML.

Le polymorphisme à la ML

Type matrice : $M ::= \mathcal{T} \mid \alpha \mid M \times M \mid M \to M$
Type : $S ::= \forall\alpha_1 \ldots \forall\alpha_n.M$
Un type matrice est un cas particulier de type.
Expressions : Comme les expressions à la Curry

Typage de constantes

$$
\begin{array}{ll}
TC(+) : & \mathit{int} \times \mathit{int} \to \mathit{int} \\
TC(\mathit{fst}) : & \forall(\alpha, \beta).(\alpha \times \beta) \to \alpha \\
TC(\mathit{snd}) : & \forall(\alpha, \beta).(\alpha \times \beta) \to \beta \\
TC(\mathit{ifthenelse}) : & \forall\alpha.(\mathit{bool} \times \alpha \times \alpha) \to \alpha \\
TC(\mathit{fix}) : & \forall\alpha.(\alpha \to \alpha) \to \alpha
\end{array}
$$

Instances d'un type

Definition

Le type A est une instance du type $\forall(\alpha_1, \ldots, \alpha_n).B$, noté $A \leq \forall(\alpha_1, \ldots, \alpha_n).B$ ssi ils existent $C_1, \ldots, C_n$ t.q. $A = B\{\alpha_1, \ldots, \alpha_n / C_1, \ldots, C_n\}$.
En particulier, pour $n = 0$, on a $A \leq B$ ssi $A \equiv B$.

Exemple :

$$int \to int \leq \forall\alpha.\alpha \to \alpha$$
$$bool \to int \leq \forall(\alpha, \beta).\alpha \to \beta$$
$$bool \to int \not\leq \forall\alpha.\alpha \to \alpha$$

Généralisations d'un type

Definition

L'opérateur *Gen* est donné par $Gen(A, \Gamma) = \forall(\alpha_1, \ldots, \alpha_n).A$, où $\{\alpha_1, \ldots, \alpha_n\} = VarLib(A) \setminus VarLib(\Gamma)$.

Exemple : Soit $A = \alpha \to \beta$ et $\Gamma = x : \beta, y : \forall\alpha.\alpha$. Alors $Gen(A, \Gamma) = \forall\alpha.\alpha \to \beta$.

Règles du typage polyomorphe à la Curry

$$\frac{A \leq \Gamma(x)}{\Gamma \vdash x : A} \qquad \frac{A \leq TC(cte)}{\Gamma \vdash cte : A}$$

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

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

Exemple de dérivation

Soit $A = \forall\alpha.\alpha \to \alpha$

$$\dfrac{\dfrac{y : \alpha \vdash y : \alpha}{\vdash \lambda y.y : \alpha \to \alpha} \quad \dfrac{\dfrac{int \to int \leq A}{f : A \vdash f : int \to int} \quad f : A \vdash 1 : int}{f : \forall\alpha.\alpha \to \alpha \vdash f\ 1 : int}}{\vdash \textbf{let}\ f = \lambda y.y\ \textbf{in}\ f\ 1 : int}$$

Propriétés

- **(Stabilité)** : Si $\Gamma, x : \forall(\alpha_1, \ldots, \alpha_n).B \vdash M : A$ et $\Gamma \vdash N : B$, alors $\Gamma \vdash M\{x/N\} : A$.
- **(Préservation)** : Si $\Gamma \vdash M : A$ et $M \Rightarrow M'$, alors $\Gamma \vdash M' : A$.

Inférence de types polymorphes

Soit $M \equiv \textbf{let}\ x = \lambda y.y\ \textbf{in}\ x$.
On considère l'ensemble d'équations :

$$\{\alpha_M \doteq \alpha_x, \alpha_x \doteq \mathit{Gen}(\alpha_B, \emptyset), \alpha_B \doteq \alpha_y \to \alpha_y\}$$

où α_B est le type de $B \equiv \lambda y.y$.
Si l'on traite la deuxième équation on obtient $\alpha_x \equiv \forall \alpha_B.\alpha_B$, **incorrect** car α_x doit être un type fonctionnel.

Vers un algorithme de typage : notations

Soit $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ un environnement et soit σ une substitution de types. Alors $\sigma(\Gamma) = x_1 : \sigma(A_1), \ldots, x_n : \sigma(A_n)$.

On note *inst*$(\forall(\alpha_1, \ldots, \alpha_n).A)$ le type $A\{\alpha_1, \ldots, \alpha_n / \beta_1, \ldots, \beta_n\}$, où $\beta_1, \ldots, \beta_n$ sont de variables fraiches.

Sécification de l'algorithme

Entrée : Un environnement Γ et un terme M t.q. $FV(M) \subseteq \Gamma$.
Sortie : Un type A et une substitution σ t.q. $\sigma(\Gamma) \vdash M : A$.

Algorithme de Damas-Milner-Tofte

$W((\Delta, x : A), x) = (inst(A), id)$
$W(\Delta, cte) = (inst(A), id)$ où A est le type de la constante cte
$W(\Delta, \lambda x.N) = (\rho(\alpha) \to B, \rho)$, où $W((\Delta, x : \alpha), N) = (B, \rho)$ et α est une variable fraiche
$W(\Delta, N\ L) = (\mu(\alpha), \mu \circ \rho_C \circ \rho_B)$, où $W(\Delta, N) = (B, \rho_B)$,
$W(\rho_B(\Delta), L) = (C, \rho_C)$, α est une variable fraiche et
$\mu = MGU(\rho_C(B) \doteq C \to \alpha)$
$W(\Delta, \langle N, L \rangle) = (\rho_C(B) \times C, \rho_C \circ \rho_B)$, $W(\Delta, N) = (B, \rho_B)$,
$W(\rho_B(\Delta), L) = (C, \rho_C)$
$W(\Delta, \textbf{let } x = N \textbf{ in } L) = (C, \rho_C \circ \rho_B)$, où $W(\Delta, N) = (B, \rho_B)$,
$W((\rho_B(\Delta), x : Gen(B, \rho_B(\Delta))), L) = (C, \rho_C)$.

Premier exemple

$W(\emptyset, \lambda x.x + x) = (int \to int, \{\alpha/int, \beta/int\})$
$W(x : \alpha, x + x) = (int, \{\alpha/int, \beta/int\})$
$W(x : \alpha, +) = (int \times int \to int, id)$
$W(id(x : \alpha), \langle x, x \rangle) = (\alpha \times \alpha, id)$
$W(x : \alpha, x) = (\alpha, id)$
$MGU(int \times int \to int \doteq \alpha \times \alpha \to \beta) = \{\alpha/int, \beta/int\}$

Deuxième exemple

$W(\emptyset, \textbf{let } f = \lambda x.x \textbf{ in } f\ 2) = (int, \{\beta/int, \gamma/int\})$
$W(\emptyset, \lambda x.x) = (\alpha \to \alpha, id)$
$W(x : \alpha, x) = (\alpha, id)$
$W(f : \forall\alpha.\alpha \to \alpha, f\ 2) = (int, \{\beta/int, \gamma/int\})$
$W(f : \forall\alpha.\alpha \to \alpha, f) = (\beta \to \beta, id)$
$W(f : \forall\alpha.\alpha \to \alpha, 2) = (int, id)$
$MGU(\beta \to \beta \doteq int \to \gamma) = \{\beta/int, \gamma/int\}$

Propriétés de l'algorithme

Théorème : (Correction) Si $W(\Delta, M) = (A, \sigma)$, alors $\sigma(\Delta) \vdash M : A$.

Théorème : (Complétude) Si $\tau(\Delta) \vdash M : B$, alors $W(\Delta, M) = (A, \sigma)$, où B est une instance de A et τ est une instance de σ

Corollaire : $\emptyset \vdash M : A$ ssi $W(\emptyset, M) = (B, \sigma)$ et A est une instance de B.