# Typage

**Examen du 12 décembre 2011 – durée : 2h**

Documents autorisés

## Réduction et typage

On considère ici le langage MiniML suivant :

| | |
|---|---|
| Expressions | $e ::= x \mid c \mid op \mid \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \mid \texttt{fun}\ x \to e \mid e_1\ e_2 \mid \texttt{let}\ x\ =\ e_1\ \texttt{in}\ e_2$ |
| Constantes | $c ::= \texttt{true} \mid \texttt{false} \mid 0 \mid 1 \mid \ldots$ |
| Opérateurs | $op ::= + \mid - \mid * \mid \texttt{not} \mid \texttt{and} \mid \texttt{fix} \mid \texttt{iszero}$ |
| Types | $\tau ::= \texttt{int} \mid \texttt{bool} \mid \alpha \mid \tau_1 \to \tau_2$ |
| Valeurs | $v ::= c \mid op \mid \texttt{fun}\ x \to e$ |

On pourra utiliser l'abréviation ($\texttt{fun}\ x\ y \to \ldots$) au lieu de ($\texttt{fun}\ x \to \texttt{fun}\ y \to \ldots$) pour les fonctions à plusieurs arguments. On pourra supposer que l'opérateur `fix` est toujours appliqué à des expressions de la forme $\texttt{fun}\ f\ x \to \ldots$, et l'abréviation $\texttt{let rec}\ f\ x\ =\ e_1\ \texttt{in}\ e_2$ désignera $\texttt{let}\ f\ =\ \texttt{fix}(\texttt{fun}\ f\ x \to e_1)\ \texttt{in}\ e_2$.

Nous utiliserons ici la sémantique à petit-pas avec substitutions vue en cours. Par exemple, la règle de réduction concernant `fix` est la suivante :

$$\texttt{fix}(\texttt{fun}\ f \to e) \quad \to_\epsilon \quad e\{f \leftarrow \texttt{fix}(\texttt{fun}\ f \to e)\}$$

On rappelle également le typage de l'opérateur `fix` :

$$\vdash\ \texttt{fix}\ :\ \forall\alpha.(\alpha \to \alpha) \to \alpha$$

### Exercice 1

Pour les expressions suivantes, donner la suite des réductions dans la sémantique à petit-pas avec substitutions et déterminer si l'on aboutit à une valeur, un blocage ou à une suite infinie de réductions.

1. $(\texttt{fun}\ x\ y \to x + y)\ 1\ 2$
2. $(\texttt{fun}\ x\ y \to x + y)\ 1$
3. $(\texttt{fun}\ x\ y \to x + y)\ 1\ 2\ 3$
4. $\texttt{if false then 1 else 2}$
5. $\texttt{if false then 1 2 else 3}$
6. $\texttt{let}\ f\ =\ \texttt{fun}\ x \to x\ \texttt{in if}\ f\ \texttt{true then}\ f\ 1\ \texttt{else}\ 2$
7. $\texttt{let rec}\ f\ x\ =\ x * f(x-1)\ \texttt{in}\ f\ 3$
8. $\texttt{let rec}\ f\ x\ =\ \texttt{if iszero}(x)\ \texttt{then}\ 1\ \texttt{else}\ x * f(x-1)\ \texttt{in}\ f\ 3$

### Exercice 2

On utilise le typage polymorphe du MiniML vu en cours. Pour les expressions de la question précédente, construire une dérivation de typage dans l'environnement initialement vide, ou justifier que l'expression est mal typée. Peut-on avoir des expressions mal typées mais se réduisant malgré tout vers une valeur ? Peut-on avoir des expressions bien typées ne se réduisant pas vers une valeur ?

## L'algorithme $\mathcal{M}$

L'algorithme $\mathcal{M}$ est une variante de l'algorithme d'inférence $\mathcal{W}$ de Damas-Milner qui permet de déterminer plus finement que $\mathcal{W}$ l'emplacement éventuel d'erreurs de typage. À la base, $\mathcal{W}$ est un algorithme d'inférence, qui répond un type (et une substitution) :

$$\mathcal{W} : env * expr \to typ * substitution$$

Par contre, $\mathcal{M}$ est présenté comme un algorithme de vérification, qui prend un type en argument :

$$\mathcal{M} : env * expr * typ \to substitution$$

En fait la différence entre les deux est faible, on peut toujours commencer par donner à $\mathcal{M}$ un type inconnu, c'est-à-dire une variable de type fraîche.

L'algorithme $\mathcal{M}$ utilise le même algorithme d'unification `mgu` que $\mathcal{W}$, ainsi que les mêmes fonctions `TypCst` et `TypOp` donnant les types des constantes et opérateurs. On utilise également la fonction $\texttt{Gen}_\Gamma$ qui généralise toutes les variables de types non présentes dans $\Gamma$, et la fonction `Inst` qui au contraire remplace les variables généralisées par de nouvelles variables fraîches.

Voici les cas principaux de $\mathcal{M}$ :

$$\begin{aligned}
\mathcal{M}(\Gamma,\ c,\ \tau) &= \texttt{mgu}(\tau, \texttt{TypCst}(c)) \\
\mathcal{M}(\Gamma,\ op,\ \tau) &= \texttt{mgu}(\tau, \texttt{Inst}(\texttt{TypOp}(op))) \\
\mathcal{M}(\Gamma,\ x,\ \tau) &= \texttt{mgu}(\tau, \texttt{Inst}(\Gamma(x))) \\
\mathcal{M}(\Gamma,\ \texttt{fun}\ x \to e,\ \tau) &= \textit{soit } \varphi_1 = \texttt{mgu}(\tau, \beta_1 \to \beta_2) \textit{ avec } \beta_1\ \beta_2 \textit{ variables fraîches} \\
&\quad \textit{soit } \varphi_2 = \mathcal{M}(\varphi_1(\Gamma) + x : \varphi_1(\beta_1),\ e,\ \varphi_1(\beta_2)) \\
&\quad \textit{retourner } \varphi_2 \circ \varphi_1 \\
\mathcal{M}(\Gamma,\ e_1\ e_2,\ \tau) &= \textit{soit } \varphi_1 = \mathcal{M}(\Gamma,\ e_1,\ \beta \to \tau) \textit{ avec } \beta \textit{ variable fraîche} \\
&\quad \textit{soit } \varphi_2 = \mathcal{M}(\varphi_1(\Gamma),\ e_2,\ \varphi_1(\beta)) \\
&\quad \textit{retourner } \varphi_2 \circ \varphi_1 \\
\mathcal{M}(\Gamma,\ \texttt{let}\ x\ =\ e_1\ \texttt{in}\ e_2,\ \tau) &= \textit{soit } \varphi_1 = M(\Gamma, e_1, \beta) \textit{ avec } \beta \textit{ variable fraîche} \\
&\quad \textit{soit } \varphi_2 = M(\varphi_1(\Gamma) + x : \texttt{Gen}_{\varphi_1(\Gamma)}(\varphi_1(\beta)),\ e_2,\ \varphi_1(\tau)) \\
&\quad \textit{retourner } \varphi_2 \circ \varphi_1
\end{aligned}$$

### Exercice 3

Complétez la définition de l'algorithme $\mathcal{M}$ pour le cas (`if ... then ... else ...`).

## Exercice 4

Montrez que l'on peut ajouter le cas suivant pour accélérer le traitement du `fix` sans modifier le résultat final :

$$\mathcal{M}(\Gamma,\ \texttt{fix}(\texttt{fun}\ f \to e),\ \tau) = \mathcal{M}(\Gamma + f : \tau,\ e,\ \tau)$$

## Exercice 5

Utilisez l'algorithme $\mathcal{M}$ sur les différentes expressions de l'exercice 1. À chaque fois, partez d'un environnement vide et d'une variable de type fraîche $\alpha$. Si l'algorithme $\mathcal{M}$ a terminé sans erreur, appliquez la substitution $\varphi$ obtenue sur $\alpha$ pour trouver le type final de l'expression considérée.

## Exercice 6

Démontrer la correction de l'algorithme $\mathcal{M}$. Plus précisément, montrer que si $\mathcal{M}(\Gamma, e, \tau)$ répond sans erreur une substitution $\varphi$, alors on a bien $\varphi(\Gamma) \vdash e : \varphi(\tau)$. On pourra admettre tous les résultats intermédiaires nécessaires (par exemple concernant `mgu`).