

Examen Compilation

Université Paris Diderot, Master Ingénierie Informatique (M1).

Première session 2011-2012. Durée 2h. L'utilisation de notes de cours est autorisée, l'utilisation de tout autre document ou dispositif électronique est interdite. Le barème est donné à titre indicatif.

Exercice 1 Dans le cours nous avons discuté une extension du langage *Imp* avec fonctions, disons *ImpF* ; cet exercice continue la discussion en raffinant certains points.

- La catégorie syntaxique des **commandes** est étendue comme suit :

$$S ::= \text{skip} \mid id := e \mid S; S \mid \text{if } b \text{ then } S \text{ else } S \mid \\ \text{while } b \text{ do } S \mid id := f(e, \dots, e) \mid \text{return}(e)$$

- Une mémoire locale est une fonction partielle $ls \in [id \mapsto \mathbb{Z}]$. La mémoire m est maintenant représentée par un couple (ls, s) et les fonctions de lecture et de mise-à-jour sont adaptées en conséquence.
- La catégorie syntaxique des **continuations** est étendue comme suit :

$$K ::= \text{halt} \mid \text{returnto}(f, id, K, ls) \mid S \cdot K$$

- La sémantique à grands pas des expressions et des conditions booléennes et à petits pas des commandes repose sur des jugements de la forme :

$$(e, m) \Downarrow v \quad (b, m) \Downarrow v \quad (f, S, K, m) \rightarrow (f', S', K', m')$$

- Un programme est une suite de déclarations de fonctions dont la dernière est le 'main()'. Une déclaration de fonction a la forme :

$$f(x_1, \dots, x_n) = \text{var } x_{n+1}, \dots, x_m; S$$

où l'on peut supposer que les variables x_1, \dots, x_m , sont toutes différentes et que les variables locales x_{n+1}, \dots, x_m sont initialisées avec la valeur 0. Le passage des paramètres est par valeur.

Question 1.1 (1 point) On utilise les notations $m(x)$ pour lire le contenu de la variable x dans la mémoire m et $m[v/x]$ pour mettre à jour le contenu de la variable x dans la mémoire m . Définissez ces notations.

Question 1.2 (4 points) Définissez la sémantique à petits pas des commandes.

La machine VMF Dans le cours nous avons aussi étudié une fonction de compilation \mathcal{C} du langage *Imp* vers une machine virtuelle *VM*. Nous décrivons dans la suite une extension de cette machine, disons *VMF* destinée à permettre la compilation du langage *ImpF*.

- La mémoire locale est maintenant une fonction (on pourrait aussi dire un tableau) $ls : \{0, \dots, n-1\} \rightarrow \mathbb{Z}$.
- Une configuration dans la machine *VMF* est un vecteur de la forme :

$$(f, i, \sigma, m, B)$$

où f est un nom de fonction, i un compteur ordinal (relatif à la fonction f), σ est une pile de valeurs comme dans VM, m est une mémoire (avec la nouvelle représentation de la mémoire locale) et B est une liste de triplets de la forme (g, j, ls') où g est un nom de fonction, j un compteur ordinal (relatif à la fonction g) et ls' est une mémoire locale. Le code de la machine C est un tableau (à deux dimensions) indexé sur les noms de fonctions et les compteurs ordinaux. Ainsi $C[f, i]$ dénote l'instruction i de la fonction f . La machine VMF dispose de **nouvelles instructions** avec la sémantique suivante.

- Si $C[f, i] = \text{lvar}(k)$, $m = (ls, s)$ et $ls(k) = v$ alors :

$$(f, i, \sigma, m, B) \rightarrow (f, i + 1, v \cdot \sigma, m, B)$$

- Si $C[f, i] = \text{lstovar}(k)$, $m = (ls, s)$, k est dans le domaine de ls et $m' = (ls[v/k], s)$ alors :

$$(f, i, v \cdot \sigma, m, B) \rightarrow (f, i + 1, \sigma, m', B)$$

- Si $C[f, i] = \text{call } g \ n$, $m = (ls, s)$, $m' = ([0/0, \dots, 0/(n-1)], s)$ alors

$$(f, i, \sigma, m, B) \rightarrow (g, 0, \sigma, m', (f, i + 1, ls) \cdot B)$$

- Si $C[g, i] = \text{return}$, $m = (ls, s)$ alors

$$(g, i, \sigma, m, (f, j, ls') \cdot B) \rightarrow (f, j, \sigma, (ls', s), B)$$

Compilation de ImpF à VMF On s'intéresse maintenant à étendre la fonction de compilation \mathcal{C} de Imp à ImpF .

Question 1.3 (9 points) Définissez la fonction de compilation \mathcal{C} sur les expressions, les conditions booléennes, les commandes et les définitions de fonction. Suggestion : il peut être utile de prévoir comme paramètre de \mathcal{C} la liste des paramètres et des variables locales.

Question 1.4 (3 points) En suivant vos définitions, calculez la compilation du programme suivant :

```
sum(x) =
var y;
while (0 < x) do (y := x + y; x := x + (-1));
return y;

main() =
y := 1;
x := call sum(y);
return(x);
```

Question 1.5 (3 points) Appliquez la sémantique de la VMF au programme compilé et vérifiez que la machine calcule bien le résultat attendu.

Solution

1.1 (1 point)

$$(ls, s)(x) = \begin{cases} ls(x) & \text{si } x \in \text{dom}(ls) \\ s(x) & \text{autr m nt} \end{cases}$$

$$(ls, s)[v/x] = \begin{cases} (ls[v/x], s) & \text{si } x \in \text{dom}(ls) \\ (ls, s[v/x]) & \text{autr m nt} \end{cases}$$

1.2 (4 points)

$$\begin{aligned} (f, \text{skip}, S \cdot K, m) &\rightarrow (f, S, K, m) \\ (f, x := e, K, m) &\rightarrow (f, \text{skip}, K, m[v/x]) \quad \text{si } (e, m) \Downarrow v \\ (f, S; S', K, m) &\rightarrow (f, S, S' \cdot K, m) \\ (f, \text{if } b \text{ then } S \text{ else } S', K, m) &\rightarrow \begin{cases} (f, S, K, m) & \text{si } (b, m) \Downarrow \text{true} \\ (f, S', K, m) & \text{si } (b, m) \Downarrow \text{false} \end{cases} \\ (f, \text{while } b \text{ do } S, K, s) &\rightarrow \begin{cases} (f, S, (\text{while } b \text{ do } S) \cdot K, m) & \text{si } (b, m) \Downarrow \text{true} \\ (f, \text{skip}, K, m) & \text{si } (b, m) \Downarrow \text{false} \end{cases} \\ (f, x := \text{call } g(e_1, \dots, e_n), K, m) &\rightarrow (g, S, \text{returnto}(f, x, K, ls), m') \quad \text{si } (e_i, m) \Downarrow v_i, i = 1, \dots, n, \\ &\quad g(x_1, \dots, x_n) = \text{var } x_{n+1}, \dots, x_m; S, m = (ls, s), m' = (ls', s), \\ &\quad ls' = [v_1/x_1, \dots, v_n/x_n, 0/x_{n+1}, \dots, 0/x_m] \\ (f, \text{return}(e), \text{returnto}(g, x, K, ls'), m) &\rightarrow (g, x := v, K, m') \quad \text{si } (e, m) \Downarrow v, m = (ls, s), m' = (ls', s) \\ (f, \text{return}(e), S \cdot K, m) &\rightarrow (f, \text{return}(e), K, m) \end{aligned}$$

1.3 (9 points) Soit X la liste des paramètres et variables locales. On dénote par $i(x, X)$ la position de x dans la liste à compter de 0.

$$\begin{aligned}
\mathcal{C}(x, X) &= \begin{cases} \text{lvar}(i(x, X)) & \text{si } x \in X \\ \text{var}(x) & \text{autrement} \end{cases} \\
\mathcal{C}(n, X) &= \text{cnst}(n) \\
\mathcal{C}(e + e', X) &= \mathcal{C}(e, X) \cdot \mathcal{C}(e', X) \cdot \text{add} \\
\mathcal{C}(e < e', k, X) &= \mathcal{C}(e', X) \cdot \mathcal{C}(e, X) \cdot \text{bge}(k) \\
\mathcal{C}(x := e, X) &= \begin{cases} \mathcal{C}(e, X) \cdot \text{lsetvar}(x) & \text{si } x \in X \\ \mathcal{C}(e, X) \cdot \text{setvar}(x) & \text{autrement} \end{cases} \\
\mathcal{C}(S; S', X) &= \mathcal{C}(S, X) \cdot \mathcal{C}(S', X) \\
\mathcal{C}(\text{if } b \text{ then } S \text{ else } S', X) &= \mathcal{C}(b, k, X) \cdot \mathcal{C}(S, X) \cdot (\text{branch}(k')) \cdot \mathcal{C}(S', X) \\
&\quad \text{où : } k = \text{sz}(S) + 1, \quad k' = \text{sz}(S') \\
\mathcal{C}(\text{while } b \text{ do } S, X) &= \mathcal{C}(b, k, X) \cdot \mathcal{C}(S, X) \cdot \text{branch}(k') \\
&\quad \text{où : } k = \text{sz}(S) + 1, \quad k' = -(\text{sz}(b) + \text{sz}(S) + 1) \\
\mathcal{C}(x := f(e_1, \dots, e_n), X) &= \begin{cases} \mathcal{C}(e_n, X) \cdots \mathcal{C}(e_1, X) \cdot (\text{call } f \ m) \cdot (\text{lstovar}(i(x, X))) & \text{si } x \in X \\ \mathcal{C}(e_n, X) \cdots \mathcal{C}(e_1, X) \cdot (\text{call } f \ m) \cdot (\text{stovar}(x)) & \text{autrement} \end{cases} \\
&\quad \text{où : } f(x_1, \dots, x_n) = \text{var } x_{n+1}, \dots, x_m; S \\
\mathcal{C}(\text{return}(e), X) &= \mathcal{C}(e, X) \cdot (\text{return}) \\
\mathcal{C}(f(x_1, \dots, x_n) = \text{var } x_{n+1}, \dots, x_m; S) &= (\text{lstovar}(0)) \cdots (\text{lstovar}(n-1)) \cdot \mathcal{C}(S, X) \\
&\quad \text{où : } X = x_1, \dots, x_m \\
\mathcal{C}(\text{main}() = \text{var } x_{n+1}, \dots, x_m; S) &= (\mathcal{C}_{\text{main}}(S, X)) \\
&\quad \text{où : } X = x_1, \dots, x_m
\end{aligned}$$

La fonction $\mathcal{C}_{\text{main}}$ est défini comm \mathcal{C} sauf sur la command **return** où il est défini d la façon suivant :

$$\mathcal{C}_{\text{main}}(\text{return}(e), X) = \mathcal{C}(e, X) \cdot (\text{halt})$$

1.4 (3 points) L cod pour **sum** av c $X = [x; y]$:

```

0:  lstovar(0)
1:  lvar(0)
2:  cnst(0)
3:  bge(9)
4:  lvar(0)
5:  lvar(1)
6:  add
7:  lstovar(1)
8:  lvar(1)
9:  cnst(-1)
10: add

```

```

11: lstovar(0)
12: branch(-12)
13: lvar(1)
14: return

```

L cod pour m in :

```

0:  cst(1)
1:  stovar(y)
2:  var(y)
3:  call sum 2
4:  stovar(x)
5:  var(x)
6:  halt

```

1.5 (3 points) L calcul n dépend pas de la configuration initial de la mémoire global qu'on indique par s . Aussi on indique par ϵ la pile de valeurs et la pile de blocs d'activations vides. Enfin, on indique par $[]$ la mémoire locale vide.

$$\begin{aligned}
& (main, 0, \epsilon, ([], s), \epsilon) \\
& \rightarrow (main, 1, 1, ([], s), \epsilon) \\
& \rightarrow (main, 2, \epsilon, ([], s[1/y]), \epsilon) \\
& \rightarrow (main, 3, 1, ([], s[1/y]), \epsilon) \\
& \rightarrow (sum, 0, 1, ([0/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 1, \epsilon, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 2, 1, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 3, 0 \cdot 1, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 4, \epsilon, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 5, 1, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 6, 0 \cdot 1, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 7, 1, ([1/0, 0/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 8, \epsilon, ([1/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 9, 1, ([1/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 10, (-1) \cdot 1, ([1/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 11, 0, ([1/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 12, \epsilon, ([0/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 1, \epsilon, ([0/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 2, 0, ([0/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 3, 0 \cdot 0, ([0/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 13, \epsilon, ([0/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (sum, 14, 1, ([0/0, 1/0], s[1/y]), (main, 4, [])) \\
& \rightarrow (main, 4, 1, ([], s[1/y]), \epsilon) \\
& \rightarrow (main, 5, \epsilon, ([], s[1/x, 1/y]), \epsilon) \\
& \rightarrow (main, 6, 1, ([], s[1/x, 1/y]), \epsilon) \not\rightarrow
\end{aligned}$$