

Partiel – 19 Mars 2008
Analyse de programme et Sémantique

Documents autorisés

Durée 2h

Exercice 1 On considère dans cet exercice un langage E d'expressions arithmétiques dont la syntaxe est définie par le système d'inférence suivant :

$$(R_1) \frac{}{n} \quad (n \in \mathbb{Z}) \quad (R_2) \frac{}{x} \quad (x \in V) \quad (R_3) \frac{e_1 \quad e_2}{e_1 + e_2} \quad (R_4) \frac{}{x ++} \quad (x \in V) \quad (R_5) \frac{}{++x} \quad (x \in V)$$

Une expression $e \in E$ est donc soit un entier, soit une variable (appartenant à un certain ensemble V), soit la somme de deux expressions, soit une expression de la forme $++x$ ou $x++$ où x est une variable.

Intuitivement, l'évaluation de l'expression $x++$ produit la valeur de x fournie par l'environnement courant, et incrémente de 1 la valeur de x dans cet environnement. L'évaluation de l'expression $++x$ procède différemment : l'environnement courant est d'abord modifié en incrémentant la valeur de x de 1, et c'est cette nouvelle valeur qui sert de résultat de l'évaluation de l'expression. Ainsi, du fait de la présence des expressions $++x$ et $x++$ dans le langage, l'évaluation d'une expression produit un valeur et modifie l'environnement courant. On introduit donc des jugements de la forme :

$$\langle e, \sigma_1 \rangle \rightsquigarrow (v, \sigma_2)$$

exprimant que, dans l'environnement σ_1 , l'expression e s'évalue en la valeur v et transforme l'environnement σ_1 en l'environnement σ_2 .

1. Définir le système d'inférence permettant de décrire l'évaluation des expressions (i.e. permettant de caractériser les jugements de la forme $\langle e, \sigma_1 \rangle \rightsquigarrow (v, \sigma_2)$ qui sont corrects).
2. On considère l'environnement σ tel que $\sigma(x) = 3$. Décrire, à l'aide des règles définies à la question 1., l'évaluation de l'expression $x + (x++)$.

On considère à présent le langage L comprenant trois instructions dont la syntaxe est définie comme suit :

$$(P_1) \frac{}{x := e} \quad (x \in V, e \in E) \quad (P_2) \frac{}{x+ := e} \quad (x \in V, e \in E) \quad (P_3) \frac{c_1 \quad c_2}{c_1; c_2}$$

L'instruction $x := e$ correspond à l'affectation "classique". L'instruction $c_1; c_2$ correspond à la séquence. Informellement, l'exécution de l'instruction $x+ := e$ dans un environnement courant consiste à évaluer l'expression e dans l'environnement courant, et modifier l'environnement obtenu lors de l'évaluation de e en affectant à la variable x la valeur obtenue lors de l'évaluation de e additionnée à la valeur associée à x dans l'environnement courant initial. Par exemple, si l'on considère un environnement σ tel que $\sigma(x) = 2$, l'exécution de l'instruction $x+ := (++x)$ conduira à un environnement pour lequel $\sigma(x) = 5$.

3. Définir le système d'inférence permettant de décrire l'exécution des instructions. Lorsqu'une règle nécessitera l'évaluation d'une expression $e \in E$, seul un jugement de la forme $\langle e, \sigma_1 \rangle \rightsquigarrow (v, \sigma_2)$ pourra être utilisé pour caractériser cette évaluation (i.e. utiliser la fonction \mathcal{A} vue en cours dans le contexte du langage considéré dans cet exercice n'a aucun sens).
4. Les programmes suivants sont-ils équivalents? (si oui, le prouver, sinon donner un contre-exemple).
 - (a) $++x$ et $(x++) + 1$
 - (b) $x := x++$ et $x+ := 1$
 - (c) $x+ := e$ et $x := x + e$
 - (d) $x+ := e$ et $x := e + x$

Exercice 2 Les systèmes de règles utiles pour cet exercice sont rappelés à la fin du sujet. On ajoute au langage vu en cours une nouvelle construction dont la syntaxe est la suivante :

`do c while e`

où c est un programme (i.e. un élément de E_C) et e est une expression booléenne (i.e. un élément de E_B) et dont la sémantique informelle est “*exécuter c, puis évaluer e et si cette expression est vraie, alors exécuter à nouveau la boucle, sinon terminer l'exécution*”.

1. Définir les règles décrivant la sémantique de cette nouvelle construction (sans utiliser la construction `while` vue en cours).
2. Montrer que les deux programmes :

`do c while e`
 et `c; if e then (do c while e) else skip`

sont équivalents.

3. Montrer que les deux programmes :

`do c while e`
 et `do c while (not not e)`

sont équivalents.

Rappels.

Expressions arithmétiques.

$$\begin{array}{ll} (\mathbb{A}_1) \frac{}{n} \quad (n \in \mathbb{Z}) & (\mathbb{A}_2) \frac{}{x} \quad (x \in V) \\ (\mathbb{A}_3) \frac{a_1 \ a_2}{a_1 + a_2} & (\mathbb{A}_4) \frac{a_1 \ a_2}{a_1 - a_2} \quad (\mathbb{A}_5) \frac{a_1 \ a_2}{a_1 \times a_2} \quad (\mathbb{A}_6) \frac{a_1 \ a_2}{a_1 / a_2} \end{array}$$

Evaluation des expressions arithmétiques.

$$\begin{array}{ll} (A_1) \frac{}{\langle n, \sigma \rangle \rightsquigarrow n} \quad (n \in \mathbb{Z}) & (A_2) \frac{}{\langle x, \sigma \rangle \rightsquigarrow \sigma(x)} \quad (x \in V) \\ (A_3) \frac{\langle a_1, \sigma \rangle \rightsquigarrow \text{Err}}{\langle a_1 \ \text{op}_2 \ a_2, \sigma \rangle \rightsquigarrow \text{Err}} \quad (\text{op}_2 \in \{+, -, \times, /\}) & \\ (A_4) \frac{\langle a_1, \sigma \rangle \rightsquigarrow n_1 \quad \langle a_2, \sigma \rangle \rightsquigarrow \text{Err}}{\langle a_1 \ \text{op}_2 \ a_2, \sigma \rangle \rightsquigarrow \text{Err}} & (n_1, n_2 \in \mathbb{Z}) \\ (A_5) \frac{\langle a_1, \sigma \rangle \rightsquigarrow n_1 \quad \langle a_2, \sigma \rangle \rightsquigarrow n_2}{\langle a_1 + a_2, \sigma \rangle \rightsquigarrow n_1 + n_2} & (A_6) \frac{\langle a_1, \sigma \rangle \rightsquigarrow n_1 \quad \langle a_2, \sigma \rangle \rightsquigarrow n_2}{\langle a_1 \times a_2, \sigma \rangle \rightsquigarrow n_1 \times n_2} \\ (A_7) \frac{\langle a_1, \sigma \rangle \rightsquigarrow n_1 \quad \langle a_2, \sigma \rangle \rightsquigarrow n_2}{\langle a_1 - a_2, \sigma \rangle \rightsquigarrow n_1 - n_2} & (A_8) \frac{\langle a_1, \sigma \rangle \rightsquigarrow n_1 \quad \langle a_2, \sigma \rangle \rightsquigarrow 0}{\langle a_1 / a_2, \sigma \rangle \rightsquigarrow \text{Err}} \\ (A_9) \frac{\langle a_1, \sigma \rangle \rightsquigarrow n_1 \quad \langle a_2, \sigma \rangle \rightsquigarrow n_2}{\langle a_1 / a_2, \sigma \rangle \rightsquigarrow n_1 / n_2} & (n_2 \neq 0) \end{array}$$

Programmes.

$$x \in V, a \in E_A, b \in E_B$$

$$(C_1) \frac{}{\text{skip}} \quad (C_2) \frac{}{x := a} \quad (C_3) \frac{c_1 \ c_2}{c_1; c_2} \quad (C_4) \frac{c_1 \ c_2}{\text{if } b \text{ then } c_1 \text{ else } c_2} \quad (C_5) \frac{c}{\text{while } b \text{ do } c}$$

Exécution des programmes.

$$\begin{array}{ll} (C_1) \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma} & (C_2) \frac{\langle a, \sigma \rangle \rightsquigarrow n}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \leftarrow n]} \quad n \in \mathbb{Z} \\ (C_3) \frac{\langle c_1, \sigma \rangle \rightarrow \sigma_1 \quad \langle c_2, \sigma_1 \rangle \rightarrow \sigma_2}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma_2} & \\ (C_4) \frac{\langle b, \sigma \rangle \rightsquigarrow \text{true} \quad \langle c_1, \sigma \rangle \rightarrow \sigma_1}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma_1} & (C_5) \frac{\langle b, \sigma \rangle \rightsquigarrow \text{false} \quad \langle c_2, \sigma \rangle \rightarrow \sigma_2}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma_2} \\ (C_6) \frac{\langle b, \sigma \rangle \rightsquigarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma} & \\ (C_7) \frac{\langle b, \sigma \rangle \rightsquigarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma_1 \quad \langle \text{while } b \text{ do } c, \sigma_1 \rangle \rightarrow \sigma_2}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma_2} & \end{array}$$
