TD2 : Sémantique opérationnelle

Description du langage

$$\begin{array}{rcl} \text{Expressions } e & ::= & x \mid i \mid \texttt{(e + e')} \mid \texttt{(e < e')} \mid \text{not e} \\ \text{Instructions } c & ::= & \text{skip} \mid x = e \mid \text{while e do c} \mid \text{if } e \text{ then } c_1 \text{ else } c_2 \mid c_1 ; c_2 \\ \text{Environnements } \sigma & ::= & \emptyset \mid \sigma[x \mapsto v] \end{array}$$

Évaluation des expressions

$$\frac{}{(\texttt{n}, \sigma) \to n}\ n \in \mathbb{N} \qquad \frac{}{(\texttt{x}, \sigma) \to \sigma(\texttt{x})} \qquad \frac{(\texttt{e}, \sigma) \to v_1 \qquad (\texttt{e'}, \sigma) \to v_2}{(\texttt{e + e'}, \sigma) \to v_1 + v_2}$$

$$\frac{(\texttt{e}, \sigma) \to v_1 \qquad (\texttt{e}', \sigma) \to v_2}{(\texttt{e < e'}, \sigma) \to \text{true si } v_1 < v_2, \text{ sinon false}} \qquad \frac{(e, \sigma) \to \text{true}}{(\text{not}(e), \sigma) \to \text{false}} \qquad \frac{(e, \sigma) \to \text{false}}{(\text{not}(e), \sigma) \to \text{true}}$$

Instructions

— $(\sigma, \texttt{ if e then c else c'}) \Rightarrow (\sigma, \texttt{ c})$ si $e \to \text{true}$ (ift)

— $(\sigma, \texttt{ if e then c else c'}) \Rightarrow (\sigma, \texttt{ c'})$ si $e \to \text{false}$ (iff)

— $(\text{while } e \text{ do } c, \sigma) \Rightarrow (\text{if } e \text{ then } \{c; \text{while } e \text{ do } c\} \text{ else skip}, \sigma)$ (while)

— $(\sigma, \text{skip ; c}) \Rightarrow (\sigma, c)$ (skip)

— $(x = e, \sigma) \Rightarrow (\text{skip}, \sigma[x \mapsto v])$ si $e \to v$ (aff)

— $(c_1; c_2, \sigma) \Rightarrow (c_1'; c_2, \sigma')$ si $(c_1, \sigma) \Rightarrow (c_1', \sigma')$ (seq)

Exercice 1 Compréhension — Cours

1. Quelles sont les expressions qui s'évaluent à une valeur dans un environnement vide ? A quel type d'erreur se heurterait-on dans les autres cas ?
2. Montrez que si $(e, \sigma) \to v$, alors $v \in \mathbb{N} \cup \{\text{true,false}\}$.
3. Pourquoi est-il intéressant d'avoir vérifié le typage des programmes avant de considérer leurs réductions possibles ?

Exercice 2 Expressions — Évaluation

1. Évaluez les expressions suivantes en utilisant les règles :
   (a) $1 + 1$
   (b) $\text{not}((1 + 1) < (1 + 2))$

Exercice 3 **Programmes** On considère qu'un programme p est évalué quand $\texttt{p} \to (\sigma, \texttt{p'})$ et que p' est réduit à `skip`. Comme nous sommes dans un cadre impératif, dans ce cas, évidemment, seul l'environnement résultant de l'exécution nous intéresse. Évaluez les programmes suivants :

1. `x=1;y=2; if(x<y) then skip else skip`
2. `arg=5;x=1;while(arg>0) do {x=x*arg;arg=arg-1}`
3. `x=1;while(not(x<1)) do x=x+1`
4. `while(true) do skip`

Exercice 4 **Terminaison** Certains programmes (comme le dernier de l'exercice précédent) ne peuvent pas s'évaluer, on considère donc qu'ils ont une sémantique (ou un comportement) *indéfinie*. Proposez une méthode pour déterminer les programmes ayant une sémantique définie :

1. En analysant les programmes systématiquement pour trier ceux qui sont définis et ceux qui sont indéfinis.
2. En modifiant le langage pour n'avoir que des programmes à sémantique définie.

Exercice 5 **Extension** Considérez les extensions suivantes du langage, et expliquez comment enrichir la sémantique opérationnelle pour les prendre en compte.

— Boucle for
— `x++`[1], `++x`, `x+=e`
— Opérations de listes
— Opérations de tableaux
— Fonctions
— `print` et `read`
— Exceptions

Exercice 6 **Sémantique à grands pas** Les instructions sont accompagnées d'une sémantique dite *à petits pas*. Modifiez les règles pour définir une sémantique à grands pas, c'est-à-dire où l'on n'a plus de relation $(c, \sigma) \to (c', \sigma')$ décrivant des étapes d'exécution, mais des relations $(c, \sigma) \to \sigma'$ aboutissant à chaque fois à l'état final de la mémoire (pas d'étapes de programme intermédiaires). Par exemple, en utilisant de telles règles, on doit pouvoir dériver :

$$\texttt{x=0;y=1;while(x<10)\{y=2*(y+x); x=x+1\}} \to [x \mapsto 10, y \mapsto 3050]$$

1. Éventuellement, qui peut être utilisé comme instruction *et* comme expression.