Sémantique des langages de programmation
TD n° 2

Semestre 5, 2022-23

1 IMP

1. Déterminer si les programmes impératifs suivants sont équivalents, en justifiant votre réponse :
   - pour toute expression booléenne b et tout programme c,
     `while` b `do` c et `if` b `then (`c`; while` b `do` c`) else skip`
   - `while x <= 0 do (y := 5; x := x + 1)` et `y:= 5;while x <= 0 do x := x + 1`
   - `x:= x + y; y := x - y; x = x - y` et `y := x + y; x := y - x; y := y - x`
   - `if 0 * x = 0 then y := 1 else y := -1` et `y := 1`
   - pour toute expression arithmétique a,
     `if 0 *` a `= 0 then y := 1 else y := -1` et `y := 1`
2. Montrer que le programme suivant termine :
   `while x <= y do (x := x + 2; y := y + 1)`

2 Pointeurs

On souhaite étendre le langage IMP avec la gestion de pointeurs. Pour cela, on étend la syntaxe des expressions et des instructions avec les constructions suivantes :

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \mid a_1/a_2 \mid \boldsymbol{\&x} \mid \boldsymbol{*a}$$
$$c ::= \mathsf{skip} \mid x := a \mid c_1; c_2 \mid \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \mid \mathsf{while}\ b\ \mathsf{do}\ c \mid \boldsymbol{a_1 \leftarrow a_2}$$

où x représente un nom de variable, a, a_1, a_2 des expressions arithmétiques, b une expression booléenne (inchangée par rapport au cours) c, c_1, c_2 des instructions.

L'intuition est que $\&x$ représente l'adresse de la variable x, $*a$ le contenu en mémoire à l'adresse calculée par a, et $a_1 \leftarrow a_2$ met la valeur calculée par a_2 à l'adresse calculée par a_1. (En C, cela s'écrirait `*`$\mathtt{a}_1$ `=` $\mathtt{a}_2$.)

Par exemple on peut considérer le programme suivant :

$$x := \&a; y := \&b; tmp := *x; x \leftarrow *y; y \leftarrow tmp \tag{1}$$

qui intuitivement échange les valeurs de a et b.

On va utiliser un modèle mémoire très simple pour représenter la sémantique de ce langage. On va considérer que n'importe quel entier correspond à une adresse valide, et

que chaque adresse-mémoire contient un entier. Pour cela, on ne considère plus seulement une valuation qui associe à une variable directement une valeur, mais une première valuation α qui associe à une variable son adresse et une deuxième valuation ν qui associe à une adresse la valeur présente à cette adresse. Les configurations seront donc de la forme $\langle c, \alpha, \nu \rangle$.

1. Étendre la sémantique opérationnelle à grands pas des expressions arithmétiques et de IMP pour prendre en compte ces changements. Les jugements seront donc de la forme respectivement $\langle a, \alpha, \nu \rangle \rightsquigarrow n$ et $\langle c, \alpha, \nu \rangle \to \langle \alpha', \nu' \rangle$. On indiquera en particulier comment les règles (A_2) et (C_2) sont modifiées.
2. Étendre la sémantique opérationnelle à petits pas des expressions arithmétiques et de IMP. Les jugements seront donc de la forme $\langle e, \alpha, \nu \rangle \hookrightarrow \langle e', \alpha', \nu' \rangle$. On indiquera en particulier comment les règles (AS_1) et (S_1) sont modifiées.
3. À partir des valuations $\alpha = \{x \mapsto 0; y \mapsto 1; a \mapsto 2; b \mapsto 3; tmp \mapsto 4\}$ et $\nu(x) = x$ pour tout x, donner les sémantiques à petits pas (en donnant les arbres d'inférence justifiant les étapes) et à grands pas du programme (1).
4. Même question en partant des valuations $\alpha = \{x \mapsto 0; y \mapsto 1; a \mapsto 1; b \mapsto 0; tmp \mapsto 4\}$ et $\nu(x) = x$ pour tout x.
5. Montrer que si $\langle c, \alpha, \nu \rangle \to \langle \alpha', \nu' \rangle$ alors $\alpha = \alpha'$. (Autrement dit, l'adresse des variables ne peut pas être modifié par un programme.)
6. Montrer l'équivalence entre les sémantiques opérationnelles à petits et à grands pas. On pourra ne considérer que les cas qui changent.
7. Les programmes ou expressions suivantes sont-elles équivalentes ? Si oui le démontrer, sinon donner un contre exemple.
   a) $\&x \leftarrow a$ et $x := a$
   b) $*(\&x)$ et x
   c) $x \leftarrow a; y := *x$ et $y := a$
   d) $x \leftarrow *x + 1; y \leftarrow *y \times 2$ et $y \leftarrow *y \times 2; x \leftarrow *x + 1$
   e) $tmp := *x; x \leftarrow *y; y \leftarrow tmp$ et
   $x \leftarrow *x + *y; y \leftarrow *x - *y; x \leftarrow *x - *y$
8. On part de la valuation $\alpha_0 = \{x \mapsto 0, y \mapsto 3\}$ et
   $\nu_0 = \{0 \mapsto 1, 1 \mapsto 2, 2 \mapsto 0, 3 \mapsto 4, 4 \mapsto 5, 5 \mapsto 6, 6 \mapsto 7\}$. Donner la sémantique à petits pas et à grands pas de
   while not $(*x = 0)$ do $(y \leftarrow *x; y := y + 1; x := x + 1)$.
   On donnera les arbres de dérivation.
9. Quelle est la sémantique du même programme si on part de $\alpha'_0 = \{x \mapsto 0, y \mapsto 2\}$ et du même ν_0 ?