

Programmation 1

TD n°6

Aliaume Lopez

30 octobre 2019

$$\begin{array}{c}
 \frac{}{\rho \vdash x := e \Rightarrow \rho[x \mapsto \llbracket e \rrbracket \rho]} \text{ (:=)} \qquad \frac{}{\rho \vdash \text{skip} \Rightarrow \rho} \text{ (Skip)} \\
 \\
 \frac{\rho \vdash c_1 \Rightarrow \rho' \quad \rho' \vdash c_2 \Rightarrow \rho''}{\rho \vdash c_1; c_2 \Rightarrow \rho''} \text{ (Seq)} \\
 \\
 \frac{\rho \vdash c_1 \Rightarrow \rho'}{\rho \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 \Rightarrow \rho' \text{ si } \llbracket e \rrbracket \rho \neq 0} \text{ (if}_1\text{)} \qquad \frac{\rho \vdash c_2 \Rightarrow \rho''}{\rho \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 \Rightarrow \rho' \text{ si } \llbracket e \rrbracket \rho = 0} \text{ (if}_2\text{)} \\
 \\
 \frac{\rho \vdash c \Rightarrow \rho' \quad \rho' \vdash \text{while } e \text{ do } c \Rightarrow \rho''}{\rho \vdash \text{while } e \text{ do } c \Rightarrow \rho'' \text{ si } \llbracket e \rrbracket \rho \neq 0} \text{ (while)} \qquad \frac{}{\rho \vdash \text{while } e \text{ do } c \Rightarrow \rho \text{ si } \llbracket e \rrbracket \rho = 0} \text{ (while}_{\text{fin}}\text{)}
 \end{array}$$

FIGURE 1 – La sémantique opérationnelle à grands pas de IMP.

Exercice 1 :

Prouvez que pour tout environnement ρ , il n'existe pas d'environnement ρ' tel que $\rho \vdash \text{while } 1 \text{ do skip} \Rightarrow \rho'$.

Exercice 2 :

Étant donnés deux programmes c_1, c_2 , on dit que c_1 et c_2 sont équivalents, noté $c_1 \sim c_2$ ssi pour tout environnements ρ, ρ' , $(\rho \vdash c_1 \Rightarrow \rho') \Leftrightarrow (\rho \vdash c_2 \Rightarrow \rho')$

1. Montrer que $\text{while } e \text{ do } c \sim \text{if } e \text{ then } (c; \text{while } e \text{ do } c) \text{ else skip}$
2. Quels sont les programmes équivalents à $\text{while } 1 \text{ do skip}$?
3. On considère un contexte C qui est simplement un programme avec une variable \square correspondant à une commande. La variable peut être utilisée à de multiples reprises. Montrer que $c_1 \sim c_2$ implique $C[c_1] \sim C[c_2]$.

Exercice 3 :

L'un des objectifs des cours suivants sera d'introduire les outils mathématiques nécessaires pour pouvoir définir une sémantique dénotationnelle de IMP : $\llbracket c \rrbracket \rho = \rho'$, ce qui signifie que l'exécution du programme c dans l'environnement ρ mène à l'environnement ρ' . Cette écriture «fonctionnelle» suggère que l'environnement ρ' est entièrement déterminé par c et ρ (bien sûr, puisque nos programmes sont «déterministes»).

1. Montrer cette propriété pour la sémantique opérationnelle : pour tout c, ρ, ρ_1, ρ_2 , si $\rho \vdash c \Rightarrow \rho_1$ et $\rho \vdash c \Rightarrow \rho_2$ alors $\rho_1 = \rho_2$.
2. On considère le langage non déterministe donné par la syntaxe suivante :

$$c ::= \text{skip} \mid x := e \mid c; c \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \mid c \vee c$$

où l'instruction $c_1 \vee c_2$ signifie «exécute c_1 ou exécute c_2 , de manière non déterministe». Proposez une sémantique opérationnelle de ce langage.

Exercice 4 :

En cours, vous avez aperçu la distinction entre sémantique dénotationnelle et sémantique opérationnelle des programmes IMP. Cependant, vous n'avez utilisé qu'une sémantique dénotationnelle des expressions arithmétiques (rappelée ci dessous). Dans cet exercice, on s'étudie le cas de deux extensions des opérations arithmétiques :

1. On étend nos expressions arithmétiques par la syntaxe suivante :

$$e ::= x \mid \dot{n} \mid e \dot{+} e \mid \dot{-} e \mid \dot{f}(e)$$

Pour interpréter le symbole \dot{f} , on suppose disposer d'une fonction **partielle** f des entiers dans les entiers.

- (a) Donnez une sémantique opérationnelle à grands pas (inspirez vous de celle pour IMP, vue en cours et rappelée en figure 1) pour ces expressions.
 - (b) Étendez la sémantique dénotationnelle vue en cours pour ces expressions.
 - (c) Montrez l'équivalence des deux sémantiques.
 - (d) Quelle différence essentielle rend la sémantique dénotationnelle pour les expressions arithmétiques beaucoup plus simple que celle pour les programmes ?
2. Expressions arithmétiques avec effets de bords : on étend nos expressions arithmétiques par la syntaxe suivante :

$$e ::= x \mid \dot{n} \mid e \dot{+} e \mid \dot{-} e \mid c \text{ resultis } e$$

Intuitivement, pour évaluer l'expression $c \text{ resultis } e$, on évalue d'abord la commande c , puis on évalue e dans l'environnement obtenu.

- (a) Donnez des sémantiques opérationnelle et dénotationnelle pour ces expressions (on supposera disposer d'une sémantique pour les programmes c).
- (b) Comment s'évalue le terme $((x := x \dot{+} 1) \text{ resultis } x) \dot{+} ((y := x \dot{+} x) \text{ resultis } x)$ dans l'environnement $\rho = [x \mapsto 3]$.
- (c) Vous comprenez désormais les horreurs que permet d'écrire le $C : T[i++] = i++$.

Exercice 5 :

Dans le cours, vous avez vu une version du langage IMP sans expressions booléennes. Dans votre compilateur, vous aviez définie ces expressions par :

$$b ::= \text{True} \mid \text{False} \mid e \dot{=} e \mid e \dot{<} e \mid \dot{\neg} b \mid b \dot{\vee} b \mid b \dot{\wedge} b$$

1. Donnez une sémantique opérationnelle pour les expressions booléennes
2. Modifiez vos règles pour qu'elles implémentent le "ou" paresseux : lors de l'évaluation de $b_1 \dot{\vee} b_2$, b_2 n'est pas évalué si b_1 s'évalue à **True**. Même chose pour le "et".
3. Modifiez vos règles pour qu'elles implémentent l'évaluation parallèle du "ou". Même chose pour le "et".
4. Dans les trois systèmes de déduction obtenu, que peut-on dire du nombre de dérivation d'un triplet $(b, \rho) \rightarrow _$?