Raphaël Charrondière
raphael.charrondiere@inria.fr

Correction TD 9: IMP et sa sémantique

Où l'on touche aux bases fondamentales

Exercice 1 Familiarisation avec IMP

Les programmes IMP ne renvoyant pas de résultat, on sauvegardera le résultat demandé dans la variable n.

Écrire un programme qui :

1. Calcule le max de x et y **P1.1**

`Correction :` **if** $x > y$ **then** $n := x$ **else** $n := y$

2. Somme les entiers naturels de 1 à k **P1.2**

`Correction :` $n := 0$**; while** $k > 0$ **do (**$n := n + k$**;** $k := k - 1$**)**

3. Calcule la factorielle de k **P1.3**

`Correction :` $n := 1$**; while** $k > 0$ **do (**$n := n \times k$**;** $k := k - 1$**)**

Exercice 2 Première prise de conscience

Soit les programmes :

if b **then** c_1 **else** c_2 **P2.1**
if b **then** c_1 **else skip; if** not b **then skip else** c_2 **P2.2**
while b_1 or b_2 **do** c **P2.3**
while b_1 **do** c**; while** b_2 **do** c **P2.4**

1. Les programmes P2.1 et P2.2 sont-ils équivalents ?

`Correction :` Non, les programmes P2.1 et P2.2 ne sont pas équivalents. Si b est vrai, **P2.1** exécute c_2, mais **P2.2** n'exécute aucune commande. Et même si on transformait **P2.2** en **if** b **then** c_1 **else skip; if** not b **then** c_2 **else skip**, alors on pourrait exécuter c_1 et c_2, par exemple : **if** $a > 4$ **then** $a := 0$**;** $b := 4$ **else skip; if** not b **then** $d := 24$ **else skip** avec 8 pour valeur initiale de a.

2. Même question avec P2.3 et P2.4

`Correction :` Non, les programmes P2.3 et P2.4 ne sont pas équivalents. Les conditions b_1 et b_2 peuvent alterner, par exemple $b_1 : a = 3$, $b_2 : a = 6$ et c : **if** $a > 5$ **then** $a := 3$ **else** $a := 6$, si a vaut initialement 3 ou 6, P2.3 ne termine pas, mais P2.4 termine.

Exercice 3 Expressions arithmétiques

Dans cet exercice, on ne traitera pas la soustraction de deux nombres, la multiplication et la division.

1. Rappeler la grammaire des expressions arithmétiques.

`Correction :` *Cf cours.*

2. Proposer des règles d'inférences pour les expressions arithmétiques .

`Correction :` Cas de base (nombre ou variable) : $\dfrac{}{\sigma, k \looparrowright k}$ $\dfrac{}{\sigma, x \looparrowright k}\ \sigma(x) = k$

Cas de la négation : $\dfrac{\sigma, a \looparrowright i}{\sigma, -a \looparrowright k}\ k = -i$

Cas de l'addition : $\dfrac{\sigma, a_1 \looparrowright i \qquad \sigma, a_2 \looparrowright j}{\sigma, a_1 + a_2 \looparrowright k}\ k = i + j$ (on utilise des règles similaires pour les autres opérateurs.)

Exercice 4 On plante des arbres ?

Raphaël Charrondière
raphael.charrondiere@inria.fr

Soit $\sigma_1\,(a \mapsto 1, b \mapsto 2, c \mapsto 3, d \mapsto 4)$, $\sigma_2(x \mapsto 3, y \mapsto 5)$ et $\sigma_3(x \mapsto 4, y \mapsto 1)$

1 Dérivez l'arbre de preuve pour $\sigma_1, (a + c) \times (d - b) \looparrowright 8$

`Correction :`

$$\dfrac{\dfrac{\dfrac{}{\sigma_1, a \looparrowright 1}\;\sigma_1(a) = 1 \qquad \dfrac{}{\sigma_1, c \looparrowright 3}\;\sigma_1(c) = 3}{\sigma_1, (a + c) \looparrowright 4}\;1 + 3 = 4 \qquad \dfrac{\dfrac{}{\sigma_1, d \looparrowright 4}\;\sigma_1(d) = 4 \qquad \dfrac{}{\sigma_1, b \looparrowright 2}\;\sigma_1(b) = 2}{\sigma_1, (d - b) \looparrowright 2}\;4 - 2 = 2}{\sigma_1, (a + c) \times (d - b) \looparrowright 8}\;4 \times 2 = 7$$

2 Dérivez l'arbre de preuve pour $\sigma_1, a + (b + (c + d)) \looparrowright 10$

`Correction :`

$$\dfrac{\dfrac{}{\sigma_1, a \looparrowright 1}\;\sigma_1(a) = 1 \qquad \dfrac{\dfrac{}{\sigma_1, b \looparrowright 2}\;\sigma_1(b) = 2 \qquad \dfrac{\dfrac{}{\sigma_1, c \looparrowright 3}\;\sigma_1(c) = 3 \qquad \dfrac{}{\sigma_1, d \looparrowright 4}\;\sigma_1(d) = 4}{\sigma_1, c + d \looparrowright 7}\;3 + 4 = 7}{\sigma_1, b + (c + d) \looparrowright 9}\;2 + 7 = 9}{\sigma_1, a + (b + (c + d)) \looparrowright 10}\;1 + 9 = 10$$

3 Reprenez le programme P1.1 et dérivez les arbres de preuve de $\sigma_2, P1.1 \Downarrow ?_2$ $\sigma_3, P1.1 \Downarrow ?_3$ en précisant les états mémoire en sortie.

`Correction :` On pose les états mémoires $\sigma_4 = (x \mapsto 3, y \mapsto 5, n \mapsto 5)$ et $\sigma_5(x \mapsto 4, y \mapsto 1, n \mapsto 4)$. On a les arbres :

$$\dfrac{\sigma_2, x > y \looparrowright false \qquad \dfrac{\sigma_2, y \looparrowright k}{\sigma_2, n := y \Downarrow \sigma_4}\;\sigma_4 = \sigma_2[k/n]}{\sigma_2, \textbf{if } x > y \textbf{ then } n := x \textbf{ else } n := y \Downarrow \sigma_4}$$

$$\dfrac{\sigma_3, x > y \looparrowright true \qquad \dfrac{\sigma_3, x \looparrowright k}{\sigma_3, n := x \Downarrow \sigma_5}\;\sigma_5 = \sigma_3[k/n]}{\sigma_3, \textbf{if } x > y \textbf{ then } n := x \textbf{ else } n := y \Downarrow \sigma_5}$$

Exercice 5 Tu termines ?

1 Comment traduire le fait qu'un programme (ne) termine (pas) de manière formelle ?

`Correction :` Pour un programme c, en partant d'un état mémoire σ, le programme termine ssi il existe σ' tel que $\sigma, c \Downarrow \sigma'$, (le programme ne termine pas si pour pour σ' on n'a pas $\sigma, c \Downarrow \sigma'$, (en langage maths : $\forall \sigma',\ \neg(\sigma, c \Downarrow \sigma')$)). On dit qu'un programme c termine ssi $\forall \sigma_1,\ \exists \sigma_2,\ \sigma_1, c \Downarrow \sigma_2 =$ Pour tout état mémoire initial σ_1, il existe un état mémoire σ_2 tel que l'exécution de c à partir de σ_1 donne σ_2.

Exercice 6 Parenthèses

Démontrer que les commandes $S_1; (S_2; S_3)$ et $(S_1; S_2); S_3$ sont sémantiquement équivalentes pour tout S_1, S_2, S_3

`Correction :` Si pour les états mémoires σ_1, σ_4 on a $\sigma_1, S_1; (S_2; S_3) \Downarrow \sigma_4$, on peut écrire l'arbre de dérivation (partiel) correspondant de manière unique :

$$\dfrac{\sigma_1, S_1 \Downarrow \sigma_2 \qquad \dfrac{\sigma_2, S_2 \Downarrow \sigma_3 \qquad \sigma_2, S_2 \Downarrow \sigma_4}{\sigma_2, S_2; S_3 \Downarrow \sigma_4}}{\sigma_1, S_1; (S_2; S_3) \Downarrow \sigma 4}$$

On a donc $\sigma_1, S_1 \Downarrow \sigma_2$, $\sigma_2, S_2 \Downarrow \sigma_3$ et $\sigma_2, S_2 \Downarrow \sigma_4$. On peut ainsi écrire l'arbre de dérivation (partiel) suivant :

$$\dfrac{\dfrac{\sigma_1, S_1 \Downarrow \sigma_2 \qquad \sigma_2, S_2 \Downarrow \sigma_3}{\sigma_1, S_1; S_2 \Downarrow \sigma_4} \qquad \sigma_3, S_3 \Downarrow \sigma_4}{\sigma_1, (S_1; S_2); S_3 \Downarrow \sigma 4}$$

Raphaël Charrondière
raphael.charrondiere@inria.fr

Cela démontre $\sigma_1, (S_1; S_2); S_3 \Downarrow \sigma_4$.
On démontre l'autre implication de manière similaire.

Exercice 7 Déterminisme des expressions arithmétiques

(1) Écrire le schéma général d'une preuve par induction sur les expressions arithmétiques.

`Correction :` Soit $P(\sigma, a, k)$ une propriété. Pour prouver par induction $\forall \sigma, a \in \text{expr},\ \forall k \in \mathbb{N},\ \sigma, a \nrightarrow k \implies P(\sigma, a, k)$, il faut montrer :

1. $\forall \sigma \in \mathbb{M},\ \forall k \in \mathbb{N},\ P(\sigma, k, k)$
2. $\forall \sigma \in \mathbb{M},\ \forall x \in \text{ident},\ \forall k \in \mathbb{N},\ \sigma(x) = k \implies P(\sigma, x, k)$
3. $\forall \sigma \in \mathbb{M},\ \forall a \in \text{expr},\ \forall k \in \mathbb{N},\ (\sigma, a \nrightarrow k) \wedge P(\sigma, a, k) \implies P(\sigma, -a, -k)$
4. $\forall \sigma \in \mathbb{M},\ \forall a_1, a_2 \in \text{expr},\ \forall k_1, k_2 \in \mathbb{N},$
   $(\sigma, a_1 \nrightarrow k_1) \wedge (\sigma, a_2 \nrightarrow k_2) \wedge P(\sigma, a_1, k_1) \wedge P(\sigma, a_2, k_2) \implies P(\sigma, a_1 + a_2, k_1 + k_2)$
5. idem pour les opérateurs $-$, $\times$ et $\div$

(2) Appliquer ce schéma sur le déterminisme des expressions arithmétiques.

`Correction :` On définit la propriété correspondante : $P(\sigma, a, k) \equiv \forall k' \in \mathbb{N},\ (\sigma, a \nrightarrow k') \implies k = k'$. On prouve les points de la question précédente.

1. Trivial
2. L'état mémoire est une fonction, donc $\sigma(x) = k' \wedge \sigma(x) = k \implies k = k'$
3. Soit $\sigma \in \mathbb{M}, a \in \text{expr},\ k \in \mathbb{N}$ avec $(\sigma, a \nrightarrow k)$ et $\forall k' \in \mathbb{N},\ (\sigma, a \nrightarrow k') \implies k = k'$ (H). S'il existe $k_1 \in \mathbb{N}$ tel que $\sigma, -a \nrightarrow k_1$ alors $\sigma, a \nrightarrow -k_1$, donc par (H) nous avons $-k_1 = k$, c'est-à-dire $k_1 = -k$.
4. Soit $\sigma \in \mathbb{M}, a_1, a_2 \in \text{expr},\ k_1, k_2 \in \mathbb{N}$ avec $(\sigma, a_1 \nrightarrow k_1)$, $(\sigma, a_2 \nrightarrow k_2)$, $\forall k' \in \mathbb{N},\ (\sigma, a_1 \nrightarrow k') \implies k_1 = k'$ (H1) et $\forall k' \in \mathbb{N},\ (\sigma, a_2 \nrightarrow k') \implies k_2 = k'$ (H2). S'il existe $k_s \in \mathbb{N}$ tel que $\sigma, a_1 + a_2 \nrightarrow k_s$ alors il existe k_s^1 et k_s^2 tels que $(\sigma, a_1 \nrightarrow k_s^1)$ et $(\sigma, a_2 \nrightarrow k_s^2)$. Par (H1) et (H2) $k_s^1 = k_1$ et $k_s^2 = k_2$. Donc $k_s = k_s^1 + k_s^2 = k_1 + k_2 = k$.
5. idem pour les opérateurs $-$, $\times$ et $\div$

(3) Nous avons vu qu'un programme IMP peut ne pas terminer, qu'en est-il de l'évaluation des expressions arithmétiques ?

`Correction :` L'évaluation des expressions arithmétiques fini toujours.

(4) Pouvez-vous le prouver ? (Essayez de trouver une idée)

`Correction :` Un raisonnement par induction n'est pas satisfaisant dans le sens où l'on n'essaie pas de prouver si une expression s'évalue alors l'évaluation finie ($\forall \sigma, a \in \text{expr},\ \forall k \in \mathbb{N},\ \sigma, a \nrightarrow k \implies P(\sigma, a, k)$), mais on cherche à montrer que pour toute expression arithmétique, si elle s'évalue, elle s'évalue en un temps fini, et si elle ne s'évalue pas, on peut détecter cela. On peut s'amuser à montrer qu'une expression arithmétique s'évalue toujours, et trivialement finir la preuve par induction. Proposons le raisonnement informel suivant : Il existe toujours un k tel que l'on peut construire de l'arbre de dérivation pour $\sigma, a \nrightarrow k$. En ignorant les "k" dans un premier temps on peut construire l'arbre de dérivation en un temps fini, car la taille des expressions arithmétiques des prémisses est toujours strictement inférieur à celle de la conclusion. Enfin, en partant des axiomes, il est possible de retrouver les "k".
