Sémantiques du langage de programmation IMP

On va étudier deux sémantiques différentes (la sémantique opérationnelle et la sémantique dénotationnelle) sur le langage IMP.

Pour introduire le langage IMP, on va avoir besoin de nombres, de variables, d'expressions arithmétiques, d'expressions booléennes munis de leur sémantique. Ainsi on pourra ensuite définir la syntaxe des instructions puis donner différentes sémantiques sur cette syntaxe.

On définit les nombres Num :
Syntaxe :

$$n ::= 0 \mid 1 \mid n0 \mid n1 \mid -n.$$

Sémantique :

$$\mathcal{N} : \mathsf{Num} \to \mathbb{Z}$$

définie par induction structurelle

$$\begin{aligned} \mathcal{N}[\![0]\!] &= 0; & \mathcal{N}[\![n0]\!] &= 2.\mathcal{N}[\![n]\!]; \\ \mathcal{N}[\![1]\!] &= 1; & \mathcal{N}[\![n1]\!] &= 2.\mathcal{N}[\![n]\!] + 1; \\ & & \mathcal{N}[\![-n]\!] &= -\mathcal{N}[\![n]\!]. \end{aligned}$$

On définit les variables Var par un ensemble de symboles $\{x, y, z, \dots\}$. On peut définir alors une mémoire

$$\mathsf{State} : \mathsf{Var} \to \mathbb{Z}.$$

On notera la mémoire $s = [x \mapsto 2, y \mapsto 7, \dots]$. On suppose qu'elle est totale.

On définit les expressions arithmétiques Aexp :
Syntaxe :

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2.$$

Sémantique :

$$\mathcal{A} : \mathsf{Aexp} \to (\mathsf{State} \to \mathbb{Z})$$

définie par induction structurelle

$$\begin{aligned} \mathcal{A}[\![n]\!]_s &= \mathcal{N}[\![n]\!]; & \mathcal{A}[\![a_1 + a_2]\!]_s &= \mathcal{A}[\![a_1]\!]_s + \mathcal{A}[\![a_2]\!]_s; \\ \mathcal{A}[\![x]\!]_s &= s(x); & \mathcal{A}[\![a_1 * a_2]\!]_s &= \mathcal{A}[\![a_1]\!]_s.\mathcal{A}[\![a_2]\!]_s; \\ & & \mathcal{A}[\![a_1 - a_2]\!]_s &= \mathcal{A}[\![a_1]\!]_s - \mathcal{A}[\![a_2]\!]_s; \end{aligned}$$

On définit les expressions booléennes Bexp :
Syntaxe :

$$b ::= \mathsf{true} \mid \mathsf{false} \mid a_1 = a_2 \mid a_1 \leqslant a_2 \mid \neg b \mid b_1 \wedge b_2.$$

Sémantique :

$$\mathcal{B} : \mathsf{Bexp} \to (\mathsf{State} \to \{tt, ff\})$$

définie par induction structurelle

$$\mathcal{B}[\![\mathsf{true}]\!]_s = tt; \qquad \mathcal{B}[\![\neg b]\!]_s = \begin{cases} tt & \text{si } \mathcal{B}[\![b]\!]_s = ff \\ ff & \text{si } \mathcal{B}[\![b]\!]_s = tt \end{cases};$$

$$\mathcal{B}[\![\mathsf{false}]\!]_s = ff; \qquad \mathcal{B}[\![b_1 \wedge b_2]\!]_s = \begin{cases} tt & \text{si } \mathcal{B}[\![b_1]\!]_s = \mathcal{B}[\![b_2]\!]_s = tt \\ ff & \text{sinon} \end{cases};$$

$$\mathcal{B}[\![a_1 = a_2]\!]_s = \begin{cases} tt & \text{si } \mathcal{A}[\![a_1]\!]_s = \mathcal{A}[\![a_2]\!]_s \\ ff & \text{si } \mathcal{A}[\![a_1]\!]_s \neq \mathcal{A}[\![a_2]\!]_s \end{cases};$$

$$\mathcal{B}[\![a_1 \leqslant a_2]\!]_s = \begin{cases} tt & \text{si } \mathcal{A}[\![a_1]\!]_s \leqslant \mathcal{A}[\![a_2]\!]_s \\ ff & \text{si } \mathcal{A}[\![a_1]\!]_s > \mathcal{A}[\![a_2]\!]_s \end{cases}.$$

Avant d'introduire la syntaxe des instructions du langage IMP, on va définir la substitution (opération syntaxique) :

On note la substitution $a_1[x \to a_2]$ définie par

$$\begin{aligned} n[x \to a] &= n; \\ x[x \to a] &= a; \\ y[x \to a] &= y \ (\text{si } y \neq x); \\ (a_1 + a_2)[x \to a] &= a_1[x \to a] + a_2[x \to a]; \\ (a_1 - a_2)[x \to a] &= a_1[x \to a] - a_2[x \to a]; \\ (a_1 * a_2)[x \to a] &= a_1[x \to a] * a_2[x \to a]; \\ \mathcal{A}[\![a[y \to a_0]]\!]_s &= \mathcal{A}[\![a]\!]_{(s[y \mapsto \mathcal{A}[\![a_0]\!]_s])}. \end{aligned}$$

On peut maintenant définir la syntaxe des instructions statement du langage IMP :

$$S ::= x := a \mid \mathsf{skip} \mid S_1; S_2 \mid \ \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \mid \mathsf{while}\ b\ \mathsf{do}\ S$$

où

- $x := a$ représente l'affectation de la valeur a à la variable x,
- skip représente l'instruction vide (qui ne fait rien),
- $S_1; S_2$ représente la séquence d'instructions (on va faire l'instruction S_1 puis l'instruction S_2),
- if b then S_1 else S_2 représente la conditionnelle,
- while b do S représente la boucle while.

On va maintenant définir des sémantiques différentes sur la syntaxe du langage IMP.

Sémantique opérationnelle à grands pas (ou sémantique naturelle) : On va définir les règles de dérivation de la sémantique à grands pas :

$$[\mathsf{skip}_{NS}] \qquad \langle \mathsf{skip}, s \rangle \to s$$

$$[\mathsf{ass}_{NS}] \qquad \langle x := a, s \rangle \to s\,[x \mapsto \mathcal{A}[\![a]\!]_s]$$

$$[\text{comp}_{NS}] \qquad \frac{\langle S_1, s\rangle \to s' \qquad \langle S_2, s'\rangle \to s''}{\langle S_1; S_2, s\rangle \to s''}$$

$$[\text{if}^{tt}_{NS}] \qquad \frac{\langle S_1, s\rangle \to s'}{\langle \text{ if } b \text{ then } S_1 \text{ else } S_2, s\rangle \to s'} \text{ si } \mathcal{B}[\![b]\!]_s = tt$$

$$[\text{if}^{ff}_{NS}] \qquad \frac{\langle S_2, s\rangle \to s'}{\langle \text{ if } b \text{ then } S_1 \text{ else } S_2, s\rangle \to s'} \text{ si } \mathcal{B}[\![b]\!]_s = ff$$

$$[\text{while}^{tt}_{NS}] \qquad \frac{\langle S, s\rangle \to s' \ \langle \text{while } b \text{ do } S, s'\rangle \to s''}{\langle \text{while } b \text{ do } S, s\rangle \to s''} \text{ si } \mathcal{B}[\![b]\!]_s = tt$$

$$[\text{while}^{ff}_{NS}] \qquad \langle \text{while } b \text{ do } S, s\rangle \to s \text{ si } \mathcal{B}[\![b]\!]_s = ff.$$

La sémantique à grands pas est donc définie par

$$\mathcal{S}_{ns}[\![S]\!]_s = \begin{cases} s' & \text{si on a } \langle S, s\rangle \to s' \\ undef & \text{sinon} \end{cases}.$$

Sémantique opérationnelle à petits pas :

On va définir les règles de dérivation de la sémantique à petits pas :

$$[\text{skip}_{SOS}] \qquad \langle \text{skip}, s\rangle \Rightarrow s$$

$$[\text{ass}_{SOS}] \qquad \langle x := a, s\rangle \Rightarrow s\,[x \mapsto \mathcal{A}[\![a]\!]_s]$$

$$[\text{comp}^{1}_{SOS}] \qquad \frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle}$$

$$[\text{comp}^{2}_{SOS}] \qquad \frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

$$[\text{if}^{tt}_{SOS}] \qquad \langle \text{ if } b \text{ then } S_1 \text{ else } S_2, s\rangle \Rightarrow \langle S_1, s\rangle \text{ si } \mathcal{B}[\![b]\!]_s = tt$$

$$[\text{if}^{ff}_{SOS}] \qquad \langle \text{ if } b \text{ then } S_1 \text{ else } S_2, s\rangle \Rightarrow \langle S_2, s\rangle \text{ si } \mathcal{B}[\![b]\!]_s = ff$$

$$[\text{while}_{SOS}] \quad \langle \text{while } b \text{ do } S, s\rangle \Rightarrow \langle \text{ if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, s\rangle.$$

La sémantique à petits pas est donc définie par

$$\mathcal{S}_{sos}[\![S]\!]_s = \begin{cases} s' & \text{si on a } \langle S, s\rangle \Rightarrow^* s' \\ undef & \text{sinon} \end{cases}.$$

Théorème 1. Pour toute instruction S du langage IMP, on a

$$\mathcal{S}_{ns}[\![S]\!] = \mathcal{S}_{sos}[\![S]\!].$$

La preuve de ce résultat fait l'objet du développement : Équivalence entre sémantiques opérationnelles.

Sémantique dénotationnelle :

On pose la fonction

$$\mathcal{S}_{ds} : \mathsf{statement} \to (\mathsf{State} \to \mathsf{State})$$

définie par

$$\begin{aligned} \mathcal{S}_{ds}[\![x := a\ ]\!]_s &= s[x \mapsto \mathcal{A}[\![a]\!]_s]; \\ \mathcal{S}_{ds}[\![\mathsf{skip}\ ]\!]_s &= \mathrm{Id}; \\ \mathcal{S}_{ds}[\![S_1; S_2\ ]\!] &= \mathcal{S}_{ds}[\![S_2\ ]\!] \circ \mathcal{S}_{ds}[\![S_1\ ]\!]; \\ \mathcal{S}_{ds}[\![\ \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ ]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!]); \\ \mathcal{S}_{ds}[\![\mathsf{while}\ b\ \mathsf{do}\ S\ ]\!] &= \mathrm{Fix}F. \end{aligned}$$

avec

$$cond : (\mathsf{State} \to \{tt, ff\}) \times (\mathsf{State} \to \mathsf{State}) \times (\mathsf{State} \to \mathsf{State}) \to (\mathsf{State} \to \mathsf{State})$$

définie par

$$cond(f, \sigma_1, \sigma_2)s = \begin{cases} \sigma_1 s & \text{si } fs = tt \\ \sigma_2 s & \text{si } fs = ff \end{cases}$$

et

$$F : \begin{array}{ccc} (\mathsf{State} \to \mathsf{State}) & \to & (\mathsf{State} \to \mathsf{State}) \\ g & \mapsto & cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], \mathrm{Id}) \end{array}.$$

L'opérateur Fix est un opérateur de point fixe.

$$\mathrm{Fix} : ((\mathsf{State} \to \mathsf{State}) \to (\mathsf{State} \to \mathsf{State})) \to (\mathsf{State} \to \mathsf{State}).$$

On développe cette théorie dans le livre [NN07].

Théorème 2. Pour toute instruction S du langage IMP, on a

$$\mathcal{S}_{sos}[\![S]\!] = \mathcal{S}_{ds}[\![S]\!].$$

Ainsi les trois sémantiques que l'on a vu sont équivalentes.

Logique de Hoare :

On va étudier la correction partielle de programmes sur le langage IMP grâce à la logique de Hoare.

Définition 7. On appelle triplet de Hoare le triplet suivant

$$\{P\}S\{Q\}$$

où P et Q sont des prédicats et S une instruction du langage IMP.

Il faut comprendre que P est un prédicat vrai avant le programme et Q est un prédicat vrai après l'exécution de l'instruction S.

On définit les règles de la logique de Hoare :

$$[\mathsf{skip}_H] \qquad \{P\}\ \mathsf{skip}\ \{P\}$$

$$[\mathsf{ass}_H] \qquad \{P[x \mapsto \mathcal{A}[\![a]\!]]\}\ x := a\ \{P\}$$

$$[\mathsf{comp}_H] \qquad \frac{\{P\}\ S_1\ \{Q\} \qquad \{Q\}\ S_2\ \{R\}}{\{P\}\ S_1; S_2\ \{R\}}$$

$$[\mathsf{if}_H] \qquad \frac{\{\mathcal{B}[\![b]\!] \wedge P\}\ S_1\ \{Q\} \qquad \{\neg\mathcal{B}[\![b]\!] \wedge P\}\ S_2\ \{Q\}}{\{P\}\ \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \{Q\}}$$

$$[\mathsf{while}_H] \qquad \frac{\{\mathcal{B}[\![b]\!] \wedge P\}\ S\ \{P\}}{\{P\}\ \ \mathsf{while}\ b\ \mathsf{do}\ S\ \{\neg\mathcal{B}[\![b]\!] \wedge P\}}$$

$$[\mathsf{cons}_H] \qquad \frac{\{P'\}\ S\ \{Q'\}}{\{P\}\ S\ \{Q\}}\ \text{si } P \Rightarrow P' \text{ et } Q' \Rightarrow Q.$$

Définition 8. On note $\vdash_p \{P\}S\{Q\}$ si et seulement s'il existe un arbre correct aboutissant à $\{P\}S\{Q\}$ avec les règles ci-dessus.

C'est un aspect syntaxique.

Définition 9. On note $\vDash_p \{P\}S\{Q\}$ si et seulement si pour tout état s tel que $Ps = tt$, si $\langle S, s\rangle \to s'$, alors $Qs' = tt$.

C'est un aspect sémantique.

Théorème 3. Pour tout P, Q, S,

$$\vdash_p \{P\}S\{Q\} \iff \vDash_p \{P\}S\{Q\}.$$

Le sens direct correspond à la correction de la logique de Hoare. On le montre par induction sur la dérivation.

Le sens indirect correspond à la complétude de la logique de Hoare. La preuve fait l'objet du développement : Complétude de la logique de Hoare.