930 — Sémantique des langages de programmation. Exemples.

Motivations : * Raisonner sur le comportement attendu des programmes d'un langage de programmation via des modèles mathématiques (une sémantique).
* Plusieurs modèles mathématiques peuvent décrire un même langage de programmation.

I. Le langage IMP [1, p. 7]

A. Syntaxe du langage IMP

Définition 1 : On définit la syntaxe du langage IMP avec les ensembles :
* l'ensemble des entiers n noté Num ;
* l'ensemble des variables x noté Var ;
* l'ensemble des expressions arithmétiques a noté Aexp ;
* l'ensemble des expressions Booléennes b noté Bexp ;
* l'ensemble des instructions S noté Stm.

et la grammaire associée :

$$a ::= n \mid x \mid a_1 \oplus a_2 \mid a_1 \otimes a_2 \mid a_1 \ominus a_2$$
$$b ::= \text{True} \mid \text{False} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2$$
$$S ::= x := a \mid \text{skip} \mid S_1 ; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S.$$

Exemple 2 : Factorielle (n) : $i := 1 ; rep := 1 ; \text{while } i \leq n \text{ do } (rep := rep \otimes i ; i := i \oplus 1)$

Fibonacci (n) : $x := 1 ; y := 1 ; i := 2 ; \text{while } i \leq n \text{ do } (y := x \oplus y ; x := y \ominus x ; i := i \oplus 1)$

Arbres de syntaxe en annexe (Figure 1).

B. Sémantique dénotationnelle des expressions.

Idée : Les sémantiques dénotationnelles modélisent le comportement par une fonction mathématique.

Définition 3 : On définit la sémantique des expressions du langage IMP par :

* $\mathcal{N} : \text{Num} \to \mathbb{Z}$, $\mathcal{N}[\![n]\!] = n$
* $\text{State} : \text{Var} \to \mathbb{Z}$
* $\mathcal{A} : \text{Aexp} \to (\text{State} \to \mathbb{Z})$
$$\mathcal{A}[\![n]\!]_s = \mathcal{N}[\![n]\!]$$
$$\mathcal{A}[\![x]\!]_s = s(x)$$
$$\mathcal{A}[\![a_1 \oplus a_2]\!]_s = \mathcal{A}[\![a_1]\!]_s + \mathcal{A}[\![a_2]\!]_s$$
$$\mathcal{A}[\![a_1 \ominus a_2]\!]_s = \mathcal{A}[\![a_1]\!]_s - \mathcal{A}[\![a_2]\!]_s$$
$$\mathcal{A}[\![a_1 \otimes a_2]\!]_s = \mathcal{A}[\![a_1]\!]_s \times \mathcal{A}[\![a_2]\!]_s$$
avec $s \in \text{State}$.
* $\mathcal{B} : \text{Bexp} \to (\text{State} \to \{tt, ff\})$
$$\mathcal{B}[\![\text{True}]\!]_s = tt$$
$$\mathcal{B}[\![\text{False}]\!]_s = ff$$
$$\mathcal{B}[\![a_1 = a_2]\!]_s = \begin{cases} tt & \text{si } \mathcal{A}[\![a_1]\!]_s = \mathcal{A}[\![a_2]\!]_s \\ ff & \text{sinon} \end{cases}$$
$$\mathcal{B}[\![a_1 \leq a_2]\!]_s = \begin{cases} tt & \text{si } \mathcal{A}[\![a_1]\!]_s \leq \mathcal{A}[\![a_2]\!]_s \\ ff & \text{sinon} \end{cases}$$
$$\mathcal{B}[\![\neg b]\!]_s = \begin{cases} tt & \text{si } \mathcal{B}[\![b]\!]_s = ff \\ ff & \text{sinon} \end{cases}$$
$$\mathcal{B}[\![b_1 \wedge b_2]\!]_s = \begin{cases} tt & \text{si } \mathcal{B}[\![b_1]\!]_s = \mathcal{B}[\![b_2]\!]_s = tt \\ ff & \text{sinon} \end{cases}$$

Remarque 4 : On suppose que la fonction State est totale. On la représentera par une liste finie.

Exemple 5 : On pose $s = [x \mapsto 3]$. On calcule :
$$\mathcal{A}[\![x \oplus 1]\!]_s = \mathcal{A}[\![x]\!]_s + \mathcal{A}[\![1]\!]_s = s(x) + \mathcal{N}[\![1]\!] = 3 + 1 = 4$$
$$\mathcal{B}[\![x = 1]\!]_s = \begin{cases} tt & \text{si } \mathcal{A}[\![x]\!]_s = \mathcal{A}[\![1]\!]_s \\ ff & \text{sinon} \end{cases} = \begin{cases} tt & \text{si } s(x) = \mathcal{N}(1) \\ ff & \text{sinon} \end{cases} = \begin{cases} tt & \text{si } 3 = 1 \\ ff & \text{sinon} \end{cases} = ff$$

Remarque 6 : Les fonctions $\mathcal{A}$ et $\mathcal{B}$ sont totales, mais elles ne le seraient pas en présence d'un opérateur binaire de division.

II. Sémantiques opérationnelles.

Idée : Les sémantiques opérationnelles modélisent le comportement par un système de transitions dont les règles $\langle S, s\rangle \to s'$ modélisent le fait « l'exécution de S sur l'état s termine dans l'état s' ».

A. Sémantique à grands pas [1, p. 20]

Définition 7 : On définit inductivement la sémantique à grands pas des instructions du langage IMP à l'aide de règles de la forme $\langle S, s\rangle \to s'$ pour $s, s' \in \text{State}$.

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

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

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

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

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

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

Définition 8 : On appelle arbre de dérivation la succession d'application des règles et des axiomes.

Exemple 9 : L'arbre de dérivation de la factorielle (Figure 2 en annexe).

Proposition 10 : Un langage est déterministe sous la sémantique à grands pas si et seulement si pour tout S, s, s', s'', ($\langle S, s\rangle \to s'$ et $\langle S, s\rangle \to s''$) implique $s' = s''$.

Application 11 : Le langage IMP est déterministe sous la sémantique à grands pas.

Contre-exemple 12 : Le langage IMP muni d'une nouvelle instruction choose $S_1 \square S_2$ dont la sémantique est donnée par les règles : $\dfrac{\langle S_1, s\rangle \to s' \quad \langle S_2, s\rangle \to s''}{\langle S_1 \square S_2, s\rangle \to s'}$ $\dfrac{\langle S_1, s\rangle \to s' \quad \langle S_2, s\rangle \to s''}{\langle S_1 \square S_2, s\rangle \to s''}$ est un langage non-déterministe sous la sémantique à grands pas.

Définition 13 : La sémantique à grands pas peut s'écrire sous la forme d'une fonction telle que $\mathcal{S}_{NS} : \text{Stm} \to (\text{State} \hookrightarrow \text{State})$ et $\mathcal{S}_{NS}[\![S]\!]_s = \begin{cases} s' & \text{si } \langle S, s\rangle \to s' \\ \text{undef} & \text{sinon} \end{cases}$

Exemple 14 : $\mathcal{S}_{NS}[\![\text{while true do skip}]\!]_s = \text{undef}$

Définition 15 : Deux instructions S_1 et S_2 sont sémantiquement équivalentes si pour tout $s, s' \in \text{State}$, $\langle S_1, s\rangle \to s'$ si et seulement si $\langle S_2, s\rangle \to s'$.

Exemple 16 : Les instructions (while b do S) et (if b then (S; while b do S) else skip) sont sémantiquement équivalentes.

Contre-exemple 17 : Les instructions $x := 3 ; x := x \oplus 1$ et $x := x \oplus 1 ; x := 3$ ne sont pas équivalentes sémantiquement.

Limite : La sémantique à grands pas ne permet pas d'exprimer la terminaison.

B - Sémantique à petits pas [1, p. 33]

Idée : La sémantique à petits pas est plus fine que la précédente : elle se concentre sur les étapes de l'exécution : les affectations et les tests.

Définition 18 : On définit inductivement la sémantique à petits pas des instructions du langage IMP à l'aide de règles du type $\langle S, s\rangle \Rightarrow \gamma$ où $s \in State$ et γ est de la forme $\langle S', s'\rangle$ ou s' avec $s' \in State$ et $S' \in Stm$.

$[ass_{sos}]\ \langle x := a, s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]_s]$ $\qquad [skip_{sos}]\ \langle skip, s\rangle \Rightarrow s$

$$[comp^1_{sos}]\ \frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1 ; S_2, s\rangle \Rightarrow \langle S_1' ; S_2, s'\rangle} \qquad [comp^2_{sos}]\ \frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1 ; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

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

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

$[while_{sos}]\ \langle \text{while } b \text{ do } S, s\rangle \Rightarrow \langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, s\rangle$

Définition 19 : Une séquence de dérivations pour $S \in Stm$ et $s \in State$ est soit une séquence finie $\gamma_0 \Rightarrow \dots \Rightarrow \gamma_k$ avec $\gamma_0 = \langle S, s\rangle$ et γ_k une configuration finale ; soit une séquence infinie $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \dots$ où $\gamma_0 = \langle S, s\rangle$.

Exemple 20 : Une séquence de dérivations de la factorielle (Figure 3)

Proposition 21 : Un langage est déterministe sous la sémantique à petits pas si et seulement si pour tout $S \in Stm$, $s \in State$, γ et γ', $\langle S, s\rangle \Rightarrow \gamma$ et $\langle S, s\rangle \Rightarrow \gamma'$ implique que $\gamma = \gamma'$.

Application 22 : Le langage IMP est déterministe sous la sémantique à petits pas.

Définition 23 : La fonction sémantique à petits pas est telle que :
$\mathcal{S}_{sos} : Stm \to (State \hookrightarrow State)$ et $\mathcal{S}_{sos}[\![S]\!]_s = \begin{cases} s' & \text{si } \langle S, s\rangle \Rightarrow^* s' \\ \text{undef} & \text{sinon} \end{cases}$

Proposition 24 : Si $\langle S_1 ; S_2, s\rangle \Rightarrow^k s''$, alors il existe $s' \in State$ et $k_1, k_2 \in \mathbb{N}$ tels que $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ et $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$ avec $k_1 + k_2 = k$.

Proposition 25 : Si $\langle S_1, s\rangle \Rightarrow^k s'$ alors $\langle S_1 ; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$.

Définition 26 : Deux instructions S_1 et S_2 sont sémantiquement équivalentes si et seulement si :
- $\exists s'' \in State$, $\langle S_1, s\rangle \Rightarrow^* s''$ et $\langle S_2, s\rangle \Rightarrow^* s''$

ou
- $\forall k \in \mathbb{N}, \exists \gamma_1, \gamma_2$, $\langle S_1, s\rangle \Rightarrow^k \gamma_1$ et $\langle S_2, s\rangle \Rightarrow^k \gamma_2$

Exemple 27 : Les instructions (while b do S) et (if b then (S; while b do S) else skip) sont sémantiquement équivalentes.

C - Equivalence de ces deux sémantiques [1, p. 41]

Théorème 28 (Equivalence des sémantiques)
Pour toute instruction S du langage IMP, on a $\mathcal{S}_{NS}[\![S]\!] = \mathcal{S}_{sos}[\![S]\!]$

DEV1

III - Sémantique dénotationnelle [1, p. 91]

Idée : On définit la sémantique dénotationnelle pour les instructions du langage IMP (modélisation par des fonctions). On utilise alors la théorie du point fixe afin de définir la sémantique du while.

Remarque 29 : On commence par définir la sémantique pour l'ensemble des instructions. Puis on montre que cette sémantique est bien définie en détaillant la théorie du point fixe.

Définition 30 : On définit la sémantique dénotationnelle des instructions du langage IMP par la fonction sémantique $\mathcal{S}_{DS} : Stm \to (State \hookrightarrow State)$ définie par : $\mathcal{S}_{DS}[\![x := a]\!]_s = s[x \mapsto \mathcal{A}[\![a]\!]_s]$ $\quad \mathcal{S}_{DS}[\![skip]\!] = id$

$\mathcal{S}_{DS}[\![S_1 ; S_2]\!] = \mathcal{S}_{DS}[\![S_2]\!] \circ \mathcal{S}_{DS}[\![S_1]\!]$ $\quad \mathcal{S}_{DS}[\![\text{if } b \text{ then } S_1 \text{ else } S_2]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{DS}[\![S_1]\!], \mathcal{S}_{DS}[\![S_2]\!])$

$\mathcal{S}_{DS}[\![\text{while } b \text{ do } S]\!] = Fix\ F$

avec $F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{DS}[\![S]\!], id)$ $\quad cond(p, g_1, g_2)s = \begin{cases} g_1 s & \text{si } ps = tt \\ g_2 s & \text{sinon} \end{cases}$

$Fix : ((State \hookrightarrow State) \to (State \hookrightarrow State)) \to (State \hookrightarrow State)$
qui se lit « le plus petit point fixe ».

Exercice 31 : while $\neg(x=0)$ do skip possède plusieurs points fixes.
$\mathcal{S}_{DS}[\![\text{while } \neg(x=0) \text{ do skip}]\!] = (F'g)s = \begin{cases} gs & \text{si } \neg(sx = 0) \\ s & \text{sinon} \end{cases}$. On pose $g_1 s = s$ et $g_2 s = \begin{cases} \text{undef} & \text{sinon} \\ s & \text{si } s(x) = 0 \end{cases}$ qui sont des points fixes pour cette instruction.

Définition 32 : Deux instructions S_1 et S_2 sont sémantiquement équivalentes si et seulement si $\mathcal{S}_{DS}[\![S_1]\!] = \mathcal{S}_{DS}[\![S_2]\!]$.

Exemple 33 : (S; skip) et (S) sont sémantiquement équivalentes.

A. Théorie du point fixe

Définition 34 : On introduit un ordre partiel $\sqsubseteq$ sur $(State \hookrightarrow State)$: $g_1 \sqsubseteq g_2$ si et seulement si $g_1 s = s' \Rightarrow g_2 s = s'$.

Définition 35 : Etant donné $(D, \sqsubseteq)$, $C \subseteq D$ est une chaîne si $\forall d_1, d_2 \in C$ $d_1 \sqsubseteq d_2$ ou $d_2 \sqsubseteq d_1$.

Exemple 36 : On définit $\forall n \in \mathbb{N}$, $g_n s = \begin{cases} \text{undef} & \text{si } x > n \\ s[x \mapsto 1] & \text{si } 0 \le x \le n \\ s & \text{sinon} \end{cases}$
$\{(g_i)\}$ est une chaîne pour $(State \hookrightarrow State, \sqsubseteq)$.

Définition 37 : $(D, \sqsubseteq)$ est un CCPO (chain complete partial order) si et seulement si pour toute chaîne Y, Y a une borne inférieure $\sqcup Y$.

Proposition 38 : $(State \hookrightarrow State, \sqsubseteq)$ est un CCPO de borne inférieure $\bot$ où $\bot$ est la fonction partielle $\bot s = \text{undef}$ $\forall s$.

Définition 39 : Soient $(D, \sqsubseteq)$, $(D', \sqsubseteq')$ deux CCPO et $f : D \to D'$ une fonction :
- f est monotone si et seulement si $\forall d_1, d_2 \in D$, $d_1 \sqsubseteq d_2 \Rightarrow f(d_1) \sqsubseteq' f(d_2)$
- f est continue si et seulement si elle est monotone et pour toute chaîne de D (non vide) $\sqcup'\{f(d), d \in Y\} = f(\sqcup Y)$.

Proposition 40 : La fonction $F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{DS}[\![S]\!], id)$ est une fonction continue.

Théorème 41: Soit f une fonction continue de $D \to D$. $Fix(f) = \sqcup\{f^n(\bot) \mid n \geq 0\}$ est un élément de D et se lit "le plus petit point fixe".
Conséquence: La sémantique dénotationnelle de l'instruction while est bien définie.

C - Équivalence des sémantiques

Théorème 42: Pour toute instruction S du langage IMP, on a $\mathcal{S}_{SOS}[\![S]\!] = \mathcal{S}_{DS}[\![S]\!]$
Corollaire 43: Pour toute instruction S du langage IMP, on a $\mathcal{S}_{DS}[\![S]\!] = \mathcal{S}_{NS}[\![S]\!]$

IV - Sémantique axiomatique (Hoare partielle) [1, p.212]

Idée: Cette sémantique facilite la preuve de programme et son automatisation. On se base sur les systèmes de déduction en logique que nous sommes capable de bien manipuler.
Correction partielle: on donne des garanties que si le programme termine.

A - Triplet de Hoare.

Définition 44: On définit le langage des prédicats :
$P ::= b \mid P \wedge P \mid P \vee P \mid \neg P \mid P[x \mapsto \mathcal{A}[\![a]\!]] \mid P \Rightarrow P$
avec $b \in Bexp$, $a \in Aexp$ et $x \in Var$.
Définition 45: On définit la sémantique des prédicats : pour tout $\sigma \in State$
* $b\,\sigma$ si et seulement si $\mathcal{B}[\![b]\!]_\sigma = tt$
* $(P_1 \wedge P_2)\sigma$ si et seulement si $P_1\sigma$ et $P_2\sigma$
* $(P_1 \vee P_2)\sigma$ si et seulement si $P_1\sigma$ ou $P_2\sigma$
* $(\neg P)_\sigma$ si et seulement si $\neg(P\sigma)$
* $(P[x \mapsto \mathcal{A}[\![a]\!]])_\sigma$ si et seulement si $P(\sigma[x \mapsto \mathcal{A}[\![a]\!]_\sigma])$
* $(P_1 \Rightarrow P_2)_\sigma$ si et seulement si $P_1\sigma$ implique $P_2\sigma$.

Définition 46: Un triplet de Hoare est la donnée d'une précondition P, d'une instruction S et d'une postcondition Q avec P et Q des prédicats. On le note $\{P\}\,S\,\{Q\}$.
Exemple 47: Un triplet de Hoare pour la factorielle
$\{n \geq 0\}$ $y := 1$; while $\neg(x = 0)$ do $(y := y \otimes x$; $x := x \ominus 1)$ $\{y = n!\}$
Définition 48 (Validité pour la sémantique à grands pas)
$\models_p \{P\}\,S\,\{Q\}$ si et seulement si $\forall\sigma$ $P\sigma = tt$, si $\langle S, \sigma\rangle \to \sigma'$ alors $Q\sigma' = tt$
Exemple 49: $\models_p \{n \geq 0\}$ $x := n$; $x := 1$ $\{x = 1\}$
Définition 50: Deux instructions S_1 et S_2 sont sémantiquement équivalentes si et seulement si pour tout prédicat P, Q, $\models_p \{P\}\,S_1\,\{Q\}$ est équivalent à $\models_p \{P\}\,S_2\{Q\}$.
Exemple 51: $(S ; skip)$ est sémantiquement équivalent à S.

B - Logique de Hoare partielle

Définition 52: On définit les règles d'inférences de la logique de Hoare partielle pour les instructions du langage IMP par :

$[ass_p]$ $\{P[x \mapsto \mathcal{A}[\![a]\!]]\}\ x := a\ \{P\}$ $\qquad$ $[skip_p]$ $\{P\}$ skip $\{P\}$

$$[comp_p]\ \frac{\{P\}\,S_1\,\{Q\} \quad \{Q\}\,S_2\,\{R\}}{\{P\}\,S_1 ; S_2\,\{R\}} \qquad [cons_p]\ \frac{\{P'\}\,S\,\{Q'\}}{\{P\}\,S\,\{Q\}} \text{ si } P \Rightarrow P' \text{ et } Q' \Rightarrow Q$$

$$[if_p]\ \frac{\{\mathcal{B}[\![b]\!] \wedge P\}\,S_1\,\{Q\} \quad \{\neg\mathcal{B}[\![b]\!] \wedge P\}\,S_2\,\{Q\}}{\{P\}\ \text{if } b \text{ then } S_1 \text{ else } S_2\ \{Q\}}$$

$$[while_p]\ \frac{\{\mathcal{B}[\![b]\!] \wedge P\}\,S\,\{P\}}{\{P\}\ \text{while } b \text{ do } S\ \{\neg\mathcal{B}[\![b]\!] \wedge P\}}$$

Exemple 53: On montre que $\{True\}$ while True do skip $\{False\}$ à l'aide de la logique de Hoare (Figure 4)
Définition 54: On note $\vdash_p \{P\}\,S\,\{Q\}$ s'il existe une preuve donnée par un arbre d'inférence tel que $\{P\}S\{Q\}$ soit à la racine et que toutes ses feuilles soient des axiomes.
Exemple 55: $\vdash_p \{True\}$ while True do skip $\{False\}$
Proposition 56: Pour toute instruction S et prédicat P, on a $\vdash_p \{P\}\,S\,\{True\}$
Définition 57: Deux instructions S_1 et S_2 sont prouvées équivalentes si et seulement si pour tout prédicat P et Q, $\vdash_p \{P\}\,S_1\{Q\}$ est équivalent à $\vdash_p \{P\}\,S_2\,\{Q\}$.
Exemple 58: $(S ; skip)$ est prouvé équivalente à S.

C - Correction et complétude.

Théorème 59 (Correction): La logique de Hoare est correcte.
Pour tous prédicats P, Q et instruction S, $\vdash_p \{P\}\,S\,\{Q\}$ implique $\models_p \{P\}\,S\,\{Q\}$
Définition 60: La précondition la plus faible est un prédicat défini tel que $wlp(S, Q)\sigma = tt$ si et seulement si $\forall\sigma' \in State$, si $\langle S, \sigma\rangle \to \sigma'$ alors $Q\sigma' = tt$
Lemme 61: $\models_p \{wlp(S, Q)\}\,S\,\{Q\}$
Lemme 62: $\models_p \{P\}\,S\,\{Q\}$ implique $P \Rightarrow wlp(S, Q)$

Théorème 63 (Complétude). La logique de Hoare est complète.
Pour tous prédicats P, Q et instruction S, $\models_p \{P\}\,S\,\{Q\}$ implique $\vdash_p \{P\}S\{Q\}$. DEV2

Ouverture:
* Les sémantiques offrent de nombreuses applications :
  + Certification de compilateur : compiler notre langage IMP dans un langage plus simple
  + Vérifier la sécurité de programme : présence ou non d'interruption de programme, leur communication, confidentialité des données, ...
* Il existe d'autres sémantiques dont on ne parle pas ici, comme la sémantique opérationnelle à petit pas de la continuation qui permet de simplifier la règle de la séquence

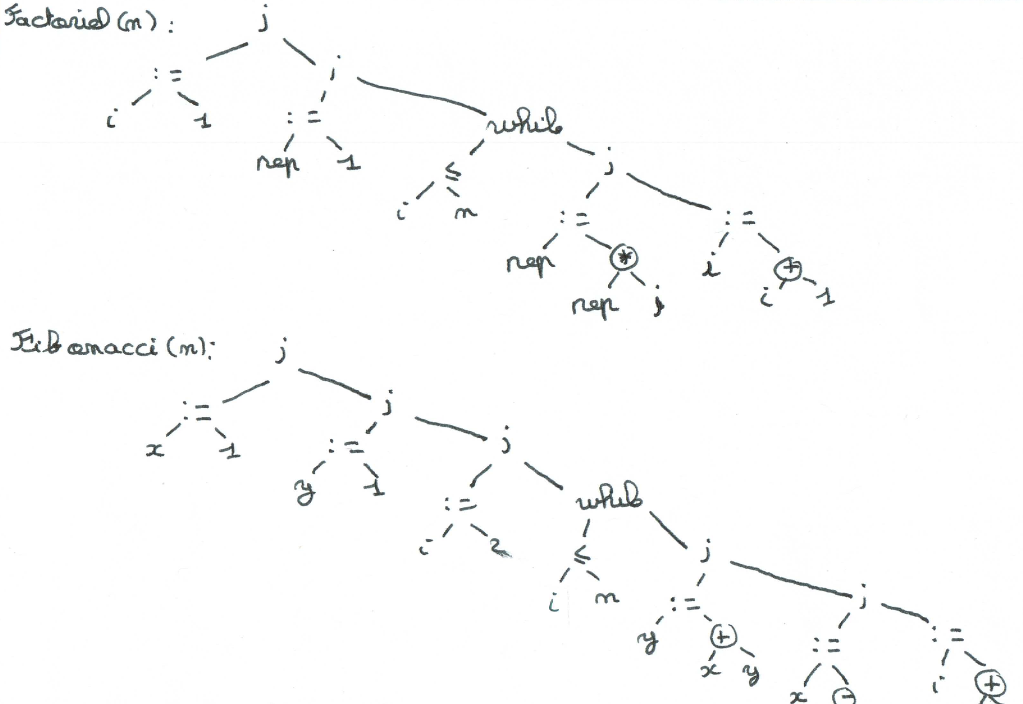

Figure 1: Arbres de syntaxe.

$$\dfrac{[\text{ass}_{NS}]\ \langle y:=1,[x\mapsto 2]\rangle \to \overbrace{[x\mapsto 2, y\mapsto 1]}^{s} \qquad [\text{while}^{tt}_{NS}]\ \dfrac{\vdots}{\langle \text{while } \neg(x=1) \text{ do } (\dots), s\rangle \to [x\mapsto 1, y\mapsto 2]}}{\langle y:=1;\ \text{while } \neg(x=1) \text{ do } (y:=y\otimes x;\ x:=x\ominus 1),[x\mapsto 2]\rangle \to [x\mapsto 1, y\mapsto 2]}\ [\text{comp}_{NS}]$$

Figure 2: Arbre de dérivation (sémantique à grands pas) pour factorielle 2.

$$[\text{comp}^2_{SOS}]\ \dfrac{\langle \underbrace{y:=1}_{S_1}, \underbrace{[x\mapsto 2]}_{s}\rangle \Rightarrow \underbrace{[x\mapsto 2; y\mapsto 1]}_{s'}}{\langle \underbrace{y:=1}_{S_1};\ \underbrace{\text{while } \neg(x=1) \text{ do } (y:=y\otimes x;\ x:=x\ominus 1)}_{S_2}, \underbrace{[x\mapsto 2]}_{s}\rangle} \Rightarrow$$

$$[\text{while}_{SOS}]\ \dfrac{}{\langle \underbrace{\text{while } \neg(x=1) \text{ do } (\underbrace{y:=y\otimes x;\ x:=x\ominus 1}_{S})}_{S_2}, \underbrace{[x\mapsto 2; y\mapsto 1]}_{s'}\rangle \Rightarrow}$$

$$\langle \text{if } \neg(x=1) \text{ then } (S;\ \text{while } b \text{ do } S) \text{ else skip}\ ;\ [x\mapsto 2; y\mapsto 1] \Rightarrow$$

...

Figure 3: Séquence de dérivation pour factorielle 2 (sémantique à petit pas).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{\text{True}\}\ \text{skip}\ \{\text{True}\}}\ [\text{skip}_p]}{\{\text{True} \wedge \text{True}\}\ \text{skip}\ \{\text{True}\}}\ [\text{cons}_p]}{\{\text{True}\}\ \text{while True do skip}\ \{\text{False} \wedge \text{True}\}}\ [\text{while}_p]}{\{\text{True}\}\ \text{while True do skip}\ \{\text{False}\}}\ [\text{cons}_p]$$

Figure 4: Preuve par la logique de Hoare de $\{\text{True}\}$ while True do skip $\{\text{False}\}$.

Références :

[1] H.R. Nielson et F. Nielson, Semantics with applications.