Sémantique des Langages de Programmation

Sémantique opérationnelle

Stefano Guerrini
stefano.guerrini@univ-paris13.fr

LIPN - Institut Galilée, Université Paris Nord 13
Sup Galillée Informatique, 1ère année

2009–2010

Sémantique opérationnelle

- Une commande in While modifie l'état, en modifiant le contenu de la mémoire.
  - Exemple : $\mathtt{x} := \mathtt{x} + 1$
    La valeur dans l'emplacement de mémoire de $\mathtt{x}$ est augmentée de 1.
- La sémantique opérationnelle décrie comme les états évoluent au cours de l'exécution d'un programme.
- La sémantique opérationnelle donne une description de l'exécution d'un programme et pas simplement des valeurs finales des exécutions.
- On considérera deux types de sémantique opérationnelle :
  - la sémantique naturelle ou à grandes étapes
  - la sémantique opérationnelle structurée ou à petites étapes

Systèmes de transition

- Pour décrire la sémantique opérationnelle des commandes on utilisera des système de transition.
- Un système de transition est composé de
  - une ensembles de configurations qui décrient les états du système
  - une relation de transition entre les configurations qui décrit comme se déroulent les changements de configuration/état
- A noter que la relation de transition donne au même temps
  - les transitions possibles à partir d'une configuration, c'est-à-dire les configuration où on peut aller à partir d'une configuration donnée
  - les conditions nécessaire pour qu'une transition se déroule

Configurations et règles de transitions

- Les configurations qu'on considérera seront de deux types :
  $\langle S, s\rangle$ où S est un commande et s un état.
  Qui représente que le commande S est exécuté dans l'état s
  s un état.
  Qui représente un état finale.
- Les transitions seront des paires
  $C \to C'$ où C et C' sont des configurations
- Les règles du système de transition seront écrit sous la forme

$$\frac{C_1 \to C_1' \quad \dots \quad C_k \to C_k'}{C_0 \to C_0'} \quad P$$

  - les transitions $C_1 \to C_1' \quad \dots \quad C_k \to C_k'$ ce sont les prémisses de la règle
  - $C_0 \to C_0'$ c'est la conclusion
  - P c'est une condition nécessaire pour pouvoir appliquer la règle

  si $k = 0$, on écriera simplement $C_0 \to C_0'$ si P.

Sémantique naturelle (ou à grandes étapes)

- Dans la sémantique naturelle, la relation de transition met en relation l'état initial de la computation directement avec l'état final.
- Les transitions ce sont donc de la forme

$$\langle S, s\rangle \to s'$$

et leur interprétation intuitive est

l'exécution de S à partir de l'état initial s
termine et l'état final ce sera s'

- Dans la sémantique naturelle, les règles ce seront du type

$$\frac{\langle S_1, s_1\rangle \to s_1' \qquad \cdots \qquad \langle S_n, s_n\rangle \to s_n'}{\langle S, s\rangle \to s'} \; P$$

et correspondent au fait que

  - si la condition P est vrai
  - et si les exécutions des S_i a partir des états initiaux s_i terminent dans les états finaux s_i'
  - alors l'exécution de S à partir de l'état initial s termine dans l'état final s'

Sémantique des commandes

(ass) $$\langle x := a, s\rangle \to s[x \mapsto \mathcal{A}[\![a]\!]s]$$

(skip) $$\langle \texttt{skip}, s\rangle \to s$$

(comp) $$\frac{\langle S_1, s\rangle \to s' \qquad \langle S_2, s'\rangle \to s''}{\langle S_1 \,;\, S_2, s\rangle \to s''}$$

(if-tt) $$\frac{\langle S_1, s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \to s'} \quad \text{if } \mathcal{B}[\![s]\!]s = tt$$

(if-ff) $$\frac{\langle S_2, s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \to s'} \quad \text{if } \mathcal{B}[\![s]\!]s = ff$$

(while-tt) $$\frac{\langle S_1, s\rangle \to s' \qquad \langle \texttt{while}\ b\ \texttt{do}\ S, s'\rangle \to s''}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to s''} \quad \text{if } \mathcal{B}[\![s]\!]s = tt$$

(while-ff) $$\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to s \quad \text{if } \mathcal{B}[\![s]\!]s = ff$$

Mise à jour de l'état

- Dans le cas d'une état $s[x_1 \mapsto a_1] \cdots [x_k \mapsto a_k]$ obtenu par une chaîne de mises a jour de l'état s,
  - l'ordre des changements de valeur de deux variable distinguée peut toujours être inversé, à cause du fait que, si $y \neq z$, alors

$$s'[y \mapsto b][z \mapsto c] = s'[z \mapsto c][y \mapsto b]$$

  - au même temps, si on change plusieurs fois la valeur de la même variable x, on peut simplifier l'expression en retenant que la dernière mise à jour de x, car

$$s'[y \mapsto c][y \mapsto b] = s'[y \mapsto b]$$

- Donc, pour le cas de mises à jour multiples on utilisera une écriture simplifiée

$$s[x_1 \mapsto a_1] \cdots [x_k \mapsto a_k] = s[x_1 \mapsto a_1, \ldots, x_k \mapsto a_k]$$

  et, sans perte de généralité (voir les propriétés précédentes), on fera l'assomption que
  - les variables son distingués, c'est-à-dire, $x_i = x_j$ ssi $i = j$
  - l'ordre des mises à jour des variables n'est pas relevant.

Exemple : la fonction factoriel

Exemple

En While, on peut écrire un commande qui calcul le factoriel

$$Fact \;=\; \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x} = 1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y} * \mathtt{x};\ \mathtt{x} := \mathtt{x} - 1)$$

Si dans l'état initial s on a $s\,\mathtt{x} = 3$, alors

$$\langle Fact, s\rangle \to s[\mathtt{y} \mapsto 6][\mathtt{x} \mapsto 1]$$

$$B \;=\; \mathtt{y} := \mathtt{y} * \mathtt{x};\ \mathtt{x} := \mathtt{x} - 1$$
$$W \;=\; \mathtt{while}\ \neg(\mathtt{x} = 1)\ \mathtt{do}\ B$$
$$Fact \;=\; \mathtt{y} := 1; W$$

Exemple : calcul du factoriel (sém. naturelle)

$$\dfrac{\dfrac{}{\langle \mathtt{y} := 1, s\rangle \to s[\mathtt{y} \mapsto 1]}\,(\text{ass}) \qquad \vdots \quad \langle W, s[\mathtt{y} \mapsto 1]\rangle \to s'}{\langle \mathit{Fact}, s\rangle \to s'}\,(\text{comp})$$

$$\dfrac{\vdots \quad \langle B, s[\mathtt{y} \mapsto 1]\rangle \to s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2] \qquad \vdots \quad \langle W, s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]\rangle \to s'}{\langle W, s[\mathtt{y} \mapsto 1]\rangle \to s'}\,(\text{while-tt})$$

$$\dfrac{\dfrac{}{\langle \mathtt{y} := \mathtt{y} * \mathtt{x}, s[\mathtt{y} \mapsto 1]\rangle \to s[\mathtt{y} \mapsto 3]}\,(\text{ass}) \qquad \dfrac{}{\langle \mathtt{x} := \mathtt{x} - 1, s[\mathtt{y} \mapsto 3]\rangle \to s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]}\,(\text{ass})}{\langle B, s[\mathtt{y} \mapsto 1]\rangle \to s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]}\,\text{comp}$$

$$\dfrac{\vdots \quad \langle B, s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]\rangle \to s' \qquad \dfrac{}{\langle W, s'\rangle \to s'}\,(\text{while-ff})}{\langle W, s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]\rangle \to s'}\,\text{while-tt}$$

$$\dfrac{\dfrac{}{\langle \mathtt{y} := \mathtt{y} * \mathtt{x}, s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]\rangle \to s[\mathtt{y} \mapsto 6, \mathtt{x} \mapsto 2]} \qquad \dfrac{}{\langle \mathtt{x} := \mathtt{x} - 1, s[\mathtt{y} \mapsto 6, \mathtt{x} \mapsto 2]\rangle \to s'}}{\langle B, s[\mathtt{y} \mapsto 3, \mathtt{x} \mapsto 2]\rangle \to s'}\,\text{comp}$$

Terminaison

- L'exécution d'un commande S dans l'état s termine ssi il y a un état s' t.q. $\langle S, s\rangle \to s'$
  L'exécution d'un commande S dans l'état s boucle ssi n'existe aucun état s' t.q. $\langle S, s\rangle \to s'$
- Un commande S termine toujours si l'exécution de S termine dans tous les états initiaux s.
  Un commande S boucle toujours si l'exécution de S boucle dans tous les états initiaux s.

Équivalence sémantique

Définition

Deux commandes S_1 et S_2 sont sémantiquement équivalents ssi, pour tout état s

$$\langle S_1, s\rangle \to s' \qquad ssi \qquad \langle S_2, s\rangle \to s'$$

Exemple

$$\texttt{while}\ b\ \texttt{do}\ S$$

est sémantiquement équivalents à

$$\texttt{if}\ b\ \texttt{then}\ (S\ ;\ \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}$$

Fonctions partielles et totales

- On dit que une relation binaire $R \subseteq A \times B$ est fonctionnelle quand

$$(a, b) \in R \quad \text{et} \quad (a, b') \in R \qquad \Longrightarrow \qquad b = b'$$

- La relation $R \subseteq A \times B$ représente une fonction (totale) de A vers B si est fonctionnelle et

$$\forall a \in A : \quad \exists b \in B : \qquad (a, b) \in R$$

et donc on peut définir

$$f_R : A \to B \qquad f_R(a) = b \quad \text{t.q.} \quad (a, b) \in R$$

- Plus en générale, on peut associer une fonction partielle de A vers B à toutes les relations fonctionnelles $R \subseteq A \times B$

$$f_R(a) = \begin{cases} b & \text{si } (a, b) \in R \\ \bot & \text{sinon} \end{cases}$$

où la valeur particulière $\bot$ dénote que la fonction n'est pas définie.
Donc, quand $f_R(a) = \bot$, on dira que f_R n'est pas définie pour a.

- Si $f_R(a) \neq \bot$ pour tout $a \in A$ alors f_R est une fonction de A vers B au sens traditionnel. Dans ce cas, on dira aussi que f_R est une fonction totale.

Déterminisme

- Un système de transition est déterministe si

$$C \to C' \quad \text{et} \quad C \to C'' \qquad \Longrightarrow \qquad C' = C''$$

- Le système du langage While est déterministe.

Théorème

La sémantique naturelle du langage While est déterministe, c'est-à-dire

$$\langle S, s\rangle \to s' \qquad \textit{and} \qquad \langle S, s\rangle \to s'' \iff s' = s''$$

- Cette propriété correspond à l'unicité du résultat de tout computation.
- La sémantique opérationnelle du langage While a une propriété de déterminisme encore plus forte : toutes les computations de $S \in \textsc{State}$ sont déterministes. En fait :
  - pour tout $\langle S, s\rangle$ que termine, pas seulement l'état final s' est unique,
  - mais il y a une seule dérivation pour $\langle S, s\rangle \to s'$ et donc une seule computation pour S a partir de l'état s (et cette computation termine dans l'état s').

La fonction sémantique $\mathcal{S}_{ns}$

$$\mathcal{S}_{ns} : \textsc{Stm} \to (\textsc{State} \hookrightarrow \textsc{State})$$

$$\mathcal{S}_{\mathrm{ns}}[\![S]\!] \in \textsc{State} \hookrightarrow \textsc{State}$$

- c'est une fonction, grâce au déterminisme
- partielle, à cause de l'existence de programmes que ne termine pas

$$\mathcal{S}_{\mathrm{ns}}[\![S]\!]s = \begin{cases} s' & \text{si } \langle S, s\rangle \to s' \\ \bot & \textit{sinon} \end{cases}$$

Sémantique opérationnelle structurée (ou à petites étapes)

- Dans la sémantique opérationnelle structurée, ou SOS, la relation de transition décrit chaque étape de l'exécution d'un programme.
- Les transitions ce sont donc de la forme

$$\langle S, s\rangle \to \langle S', s'\rangle \qquad \text{ou} \qquad \langle S, s\rangle \to s'$$

et leur interprétation intuitive est

  - dans le cas $\langle S, s\rangle \to \langle S', s'\rangle$, l'exécution de S à partir de s donne comme configuration intermédiaire $\langle S', s'\rangle$ et pour terminer l'exécution il faut continuer l'exécutions à partir ce configuration là.
  - dans le cas $\langle S, s\rangle \to s'$, la computation est terminée dans l'état final est s'.

- Dans la SOS, les règles ce seront du type

$$\frac{\langle S_1, s_1\rangle \to \gamma_1}{\langle S, s\rangle \to \gamma} P \qquad \langle S, s\rangle \to \gamma \quad \text{si} \quad P \qquad \langle S, s\rangle \to \gamma$$

et correspondent au fait que

  - si la condition P est vrai (quand il y une condition de bord P)
  - et si l'exécutions de S_1 a partir de l'état initiale s_1 arrivent dans les configurations γ_1 (quand il y a une prémisse dans la règle)
  - alors l'exécution de S à partir de l'état initial s arrive dans la configuration γ

Sémantique des commandes

(ass) $\langle x := a, s\rangle \to s[x \mapsto \mathcal{A}[\![a]\!]s]$

(skip) $\langle \texttt{skip}, s\rangle \to s$

(comp-1) $$\frac{\langle S_1, s\rangle \to \langle S_1', s'\rangle}{\langle S_1 \,;\, S_2, s\rangle \to \langle S_1' \,;\, S_2, , s'\rangle}$$

(comp-2) $$\frac{\langle S_1, s\rangle \to s'}{\langle S_1 \,;\, S_2, s\rangle \to \langle S_2, , s'\rangle}$$

(if-tt) $\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \to \langle S_1, s\rangle$ if $\mathcal{B}[\![b]\!]s = tt$

(if-ff) $\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \to \langle S_2, s\rangle$ if $\mathcal{B}[\![b]\!]s = ff$

(while) $\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to \langle \texttt{if}\ b\ \texttt{then}\ (S \,;\, \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, s\rangle$

Séquences de dérivation

- Une séquence de dérivation c'est soit une séquence finie

$$\langle S_0, s_0\rangle \to \langle S_1, s_1\rangle \to \cdots \to \gamma_k$$

  t.q. $\gamma_k \not\to \gamma$ pour toutes les configurations γ.
  - Si $\gamma_k = s$, alors on dira que la computation de S_0 à partir de s_0 termine (dans l'état s)
  - sinon, si $\gamma_k = \langle S_k, s_k\rangle$, on dira que la computation est bloquée (dans la configuration γ_k)
- soit une séquence infinie

$$\langle S_0, s_0\rangle \to \langle S_1, s_1\rangle \to \cdots \to \langle S_k, s_k\rangle \to \cdots$$

  (qui donc n'arrive jamais dans une configuration $\gamma = s$)
  - Dans ce cas là, on dira que la computation de S_0 à partir de s_0 boucle.
- Notations
  - $\langle S, s\rangle \to^k \gamma_k$, ou $k \geq 0$ si $\langle S_0, s_0\rangle \to \langle S_1, s_1\rangle \to \cdots \to \gamma_k$
  - $\langle S, s\rangle \to^* \langle S', s'\rangle$, si $\langle S, s\rangle \to^k \langle S', s'\rangle$ pour quelque k

Exemple : calcul du factoriel (SOS) - 1/2

$$Fact = \mathtt{y} := 1; W \qquad W = \mathtt{while}\ \neg(\mathtt{x} = 1)\ \mathtt{do}\ B$$
$$B = \mathtt{y} := \mathtt{y} * \mathtt{x};\ \mathtt{x} := \mathtt{x} - 1$$

$$\text{(Comp-2)}\ \frac{\langle \mathtt{y} := 1, s[\mathtt{x} \mapsto 3]\rangle \to s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1]}{\langle Fact, s[\mathtt{x} \mapsto 3]\rangle \to \langle W, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1]\rangle}$$

$$\text{(Comp-1)}\ \frac{\text{(Comp-1)}\ \dfrac{\langle \mathtt{y} := \mathtt{y} * \mathtt{x}, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1]\rangle \to s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 3]}{\langle B, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1]\rangle \to \langle \mathtt{x} := \mathtt{x} - 1, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 3]\rangle}}{\langle B\ ;\ W, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1]\rangle \to \langle \mathtt{x} := \mathtt{x} - 1\ ;\ W, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 3]\rangle}$$

Exemple : calcul du factoriel (SOS) - 2/2

$$
\begin{aligned}
\langle \mathit{Fact}, s[\mathtt{x} \mapsto 3] \rangle &\to \langle W, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1] \rangle \\
&\to \langle \mathtt{if}\ \neg(\mathtt{x} = 0)\ \mathtt{then}\ (B\ ;\ W)\ \mathtt{else}\ \mathtt{skip}, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1] \rangle \\
&\to \langle B\ ;\ W, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 1] \rangle \\
&\to \langle \mathtt{x} := \mathtt{x} - 1\ ;\ W, s[\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 3] \rangle \\
&\to \langle W, s[\mathtt{x} \mapsto 2, \mathtt{y} \mapsto 3] \rangle \\
&\to \langle \mathtt{if}\ \neg(\mathtt{x} = 0)\ \mathtt{then}\ (B\ ;\ W)\ \mathtt{else}\ \mathtt{skip}, s[\mathtt{x} \mapsto 2, \mathtt{y} \mapsto 3] \rangle \\
&\to \langle B\ ;\ W, s[\mathtt{x} \mapsto 2, \mathtt{y} \mapsto 3] \rangle \\
&\to \langle \mathtt{x} := \mathtt{x} - 1\ ;\ W, s[\mathtt{x} \mapsto 2, \mathtt{y} \mapsto 6] \rangle \\
&\to \langle W, s[\mathtt{x} \mapsto 1, \mathtt{y} \mapsto 6] \rangle \\
&\to \langle \mathtt{if}\ \neg(\mathtt{x} = 0)\ \mathtt{then}\ (B\ ;\ W)\ \mathtt{else}\ \mathtt{skip}, s[\mathtt{x} \mapsto 1, \mathtt{y} \mapsto 6] \rangle \\
&\to \langle \mathtt{skip}, s[\mathtt{x} \mapsto 1, \mathtt{y} \mapsto 6] \rangle \\
&\to s[\mathtt{x} \mapsto 1, \mathtt{y} \mapsto 6]
\end{aligned}
$$

La fonction sémantique $\mathcal{S}_{sos}$

$$\mathcal{S}_{\text{sos}} : \text{STM} \to (\text{STATE} \hookrightarrow \text{STATE})$$

$$\mathcal{S}_{\text{sos}}[\![S]\!] \in \text{STATE} \hookrightarrow \text{STATE}$$

$$\mathcal{S}_{\text{sos}}[\![S]\!]s = \begin{cases} s' & \text{si } \langle S, s\rangle \to^* s' \\ \bot & \textit{sinon} \end{cases}$$

Théorème

$$\forall S \in \text{STM} : \qquad \mathcal{S}_{\text{ns}}[\![S]\!] = \mathcal{S}_{\text{sos}}[\![S]\!]$$