

Devoir de maison : le langage `while` : sémantique structurale

(à rendre le 17 novembre)

1 Introduction

On rappelle la syntaxe du langage `while`. L'idée de ce devoir est d'étudier une autre façon d'aborder la description de la sémantique d'un langage. Alors que la sémantique vue en cours, appelée *sémantique naturelle* (ou sémantique à *grands pas*) montre comment une commande transforme un état, nous nous proposons d'étudier la *sémantique structurale* ou *sémantique à petits pas* qui montre comment l'état du programme se transforme en un pas de calcul.

Dans la suite, nous utiliserons l'abréviation **SOS** pour "Sémantique opérationnelle structurale".

Nous distinguons la relation de transition de la sémantique naturelle, notée en cours \longrightarrow de celle de la **SOS** que l'on notera \Longrightarrow .

2 Le langage sans les blocs

La syntaxe décrit l'ensemble des commandes (**Stm**), des expressions arithmétiques (**Aexp**) et des expressions booléennes (**Bexp**).

S	$::=$	$x := a \mid \text{skip} \mid S; S \mid \text{if } b \text{ then } S \text{ else } S \mid \text{while } b \text{ do } S$	$S \in \text{Stm}$
a	$::=$	$n \mid x \mid a + a \mid a - a \mid a * a$	$a \in \text{Aexp}$
b	$::=$	$\text{true} \mid \text{false} \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b$	$b \in \text{Bexp}$

2.1 Sémantique des commandes

Rappel des fonctions et des domaines sémantiques La **SOS** définit un système de transition $(\Gamma, \gamma_0, T, \Longrightarrow)$ où :

- Γ est l'ensemble des configurations, une configuration est de la forme (S, σ) où $S \in \text{Stm}$ est une commande et $\sigma \in \text{Mem}$ est une mémoire,
- γ_0 la configuration initiale, généralement de la forme (S, σ_0) où S est la commande à évaluer dans la mémoire initiale σ_0 ,
- T l'ensemble des configurations terminales (de la forme σ) et
- $\Longrightarrow \subseteq \Gamma \times \Gamma$ la relation de transition (ou dérivation).

$$(x := a, \sigma) \Longrightarrow \sigma[x \mapsto \mathcal{A}[a]_\sigma]$$

$$(\text{skip}, \sigma) \Longrightarrow \sigma$$

$$\frac{(S_1, \sigma) \Longrightarrow (S'_1, \sigma')}{(S_1; S_2, \sigma) \Longrightarrow (S'_1; S_2, \sigma')}$$

$$\frac{(S_1, \sigma) \Longrightarrow \sigma'}{(S_1; S_2, \sigma) \Longrightarrow (S_2, \sigma')}$$

$$(\text{if } b \text{ then } S_1 \text{ else } S_2, \sigma) \Longrightarrow (S_2, \sigma)$$

$$\text{si } \mathcal{B}[b]_\sigma = \text{ff}$$

$$(\text{if } b \text{ then } S_1 \text{ else } S_2, \sigma) \Longrightarrow (S_1, \sigma)$$

$$\text{si } \mathcal{B}[b]_\sigma = \text{tt}$$

$$(\text{while } b \text{ do } S, \sigma) \Longrightarrow (\text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, \sigma)$$

2.2 Propriétés de la sémantique

Contrairement à la sémantique naturelle, les transitions dans la SOS sont de la forme $(S, \sigma) \Longrightarrow \gamma$ où γ est soit de la forme (S', σ') , ce qui représente une étape intermédiaire dans l'évaluation de S dans l'état σ , soit de la forme σ' , ce qui signifie que l'évaluation de S dans σ est terminée.

Une configuration de *blocage* est une configuration (S, σ) sans successeur.

Un séquence de dérivation à partir d'une configuration (S, σ) est soit

- une séquence finie $\gamma_0 \cdots \gamma_k$, telle que $\gamma_0 = (S, \sigma)$, $\gamma_i \Longrightarrow \gamma_{i+1}$, γ_k est soit une configuration terminale, soit une configuration de blocage,
- une séquence infinie $\gamma_0 \cdots \gamma_k \cdots$, telle que $\gamma_0 = (S, \sigma)$ et $\gamma_i \Longrightarrow \gamma_{i+1}$.

Nous écrivons $\gamma \xRightarrow{k} \gamma'$ si il y a k pas d'exécution à partir de γ pour atteindre γ' et $\gamma \xRightarrow{*} \gamma'$ pour indiquer qu'il y a un nombre fini de pas d'exécution entre γ et γ' (fermeture transitive).

Nous dirons que l'exécution de S dans σ se termine (resp. boucle) si il existe une séquence de dérivation finie (resp. infinie) à partir de (S, σ) .

Généralement, les preuves concernant les propriétés de la SOS utilisent un schéma d'induction sur la longueur des séquences de dérivation.

LEMME 2.1 Si $(S_1; S_2, \sigma) \xRightarrow{k} \sigma''$ alors il existe σ', k_1, k_2 tel que

- $(S_1, \sigma) \xRightarrow{k_1} \sigma'$,
- $(S_2, \sigma') \xRightarrow{k_2} \sigma''$,
- $k = k_1 + k_2$

LEMME 2.2 Si $(S_1, \sigma) \xRightarrow{k} \sigma'$ alors $(S_1; S_2, \sigma) \xRightarrow{k} (S_2, \sigma')$.

Question 1 Montrer le lemme 2.1.

Question 2 Montrer que la SOS est déterministe.

La fonction sémantique \mathcal{S}_{SOS} On définit la fonction \mathcal{S}_{SOS} de la manière suivante :

$$\mathcal{S}_{SOS}[[S]]_{\sigma} = \begin{cases} \sigma' & \text{si } (S, \sigma) \xrightarrow{*} \sigma' \\ \text{undef} & \text{sinon} \end{cases}$$

LEMME 2.3 Montrer que pour tout S , tout σ, σ'

si $(S, \sigma) \longrightarrow \sigma'$ alors $(S, \sigma) \xrightarrow{*} \sigma'$

LEMME 2.4 Montrer que pour tout S , tout σ, σ'

si $(S, \sigma) \xrightarrow{k} \sigma'$ alors $(S, \sigma) \longrightarrow \sigma'$

THÉORÈME 2.1 Pour chaque S , on a $\mathcal{S}_{ns}[[S]] = \mathcal{S}_{SOS}[[S]]$, où \mathcal{S}_{ns} est la fonction sémantique vue en cours pour la sémantique naturelle.

Question Montrer ce théorème.

2.3 Extensions

2.3.1 Expressions

Question Donner une SOS pour les expressions arithmétiques et logiques en supposant qu'une expression est évaluée de gauche à droite.

Question Donner une SOS pour les expressions arithmétiques en supposant qu'une expression peut être évaluée dans n'importe quel ordre.

2.4 Arrêt d'une instruction

On introduit l'instruction `abort` qui stoppe l'exécution du programme. Pour toute mémoire σ , la configuration (abort, σ) n'a pas de successeur.

Question Montrer que `abort` et `skip` sont différents du point de vue de la sémantique naturelle et de la SOS .

Question Montrer que `abort` et `while true do skip` sont indistinguables du point de vue de la sémantique naturelle mais distinguables du point de vue de la SOS .

2.4.1 Assertion

On introduit l'instruction `assert b before S`, dont la sémantique en français est donnée par : "si `b` est vérifiée, on exécute `S`, sinon le programme s'arrête".

Question Donner la SOS de cette nouvelle construction ; montrer que `assert true before S` est sémantiquement (SOS) équivalent à `S` mais que `assert false before S` n'est équivalent ni à `skip` ni à `while true do skip`.

2.4.2 Non-déterminisme

On introduit un opérateur d'évaluation non-déterministe `or` tel que l'évaluation de `S1 or S2` dans σ conduit soit à l'évaluation de `S1`, soit à l'évaluation de `S2` dans σ . La sémantique est décrite dans les deux styles de sémantique ci-dessous :

Sémantique naturelle	SOS
$\frac{(S_1, \sigma) \longrightarrow \sigma'}{(S_1 \text{ or } S_2, \sigma) \longrightarrow \sigma'}$	$(S_1 \text{ or } S_2, \sigma) \Longrightarrow (S_1, \sigma)$
$\frac{(S_2, \sigma) \longrightarrow \sigma'}{(S_1 \text{ or } S_2, \sigma) \longrightarrow \sigma'}$	
$\frac{(S_1, \sigma) \longrightarrow \sigma' \quad (S_2, \sigma) \longrightarrow \sigma'}{(S_1 \text{ or } S_2, \sigma) \longrightarrow \sigma'}$	$(S_1 \text{ or } S_2, \sigma) \Longrightarrow (S_2, \sigma)$

Question Montrer que, contrairement à la SOS, la sémantique naturelle supprime les boucles infinies si c'est possible.

2.4.3 Parallélisme

On introduit un opérateur de parallélisme dont la sémantique naturelle est donnée ci-dessous :

Sémantique naturelle
$\frac{(S_1, \sigma) \longrightarrow \sigma' \quad (S_2, \sigma') \longrightarrow \sigma''}{(S_1 \text{ par } S_2, \sigma) \longrightarrow \sigma''}$
$\frac{(S_2, \sigma) \longrightarrow \sigma' \quad (S_1, \sigma') \longrightarrow \sigma''}{(S_1 \text{ par } S_2, \sigma) \longrightarrow \sigma''}$

Question Donnez la SOS pour la construction `par`.

2.5 Partie facultative : Bloc

Question ouverte (facultative) Si on introduit la notion `bloc`, comme vu dans le cours :

$$S ::= \dots \mid \text{begin } D_V \ S \ \text{end}$$

proposer une façon de décrire la SOS.