

1 Exercice I : Sémantique opérationnelle : extension et modification du langage while

On rappelle la syntaxe du langage while

$$\begin{aligned} S &::= x := a \mid \text{skip} \mid S; S \mid \text{if } b \text{ then } S \text{ else } S \mid \text{while } b \text{ do } S \\ a &::= n \mid x \mid a + a \\ b &::= \text{true} \mid \text{false} \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b \end{aligned}$$

1.1 Définition des variables affectées et utilisées dans les commandes

On définit les fonctions **Def** et **Use** qui, à chaque construction syntaxique, associent respectivement l'ensemble des variables affectées (qui apparaissent en partie gauche d'affectation) et l'ensemble des variables utilisées (qui apparaissent en partie droite d'affectation ou dans les expressions).

On a ,

$$\begin{aligned} \text{Def}(a) = \text{Def}(b) = \emptyset \quad \text{Def}(x := a) = \{x\} \quad \text{Def}(\text{skip}) = \emptyset \quad \text{Def}(\text{while } b \text{ do } S) = \emptyset \\ \text{Def}(S_1; S_2) = \text{Def}(S_1) \cup \text{Def}(S_2) \quad \text{Def}(\text{if } b \text{ then } S_1 \text{ else } S_2) = \text{Def}(S_1) \cap \text{Def}(S_2) \end{aligned}$$

$$\begin{aligned} \text{Use}(n) = \emptyset \quad \text{Use}(x) = \{x\} \quad \text{Use}(a_1 + a_2) = \text{Use}(a_1) \cup \text{Use}(a_2) \\ \text{Use}(\text{true}) = \emptyset = \text{Use}(\text{false}) \\ \text{Use}(a_1 = a_2) = \text{Use}(a_1) \cup \text{Use}(a_2) \quad \text{Use}(b_1 \wedge b_2) = \text{Use}(b_1) \cup \text{Use}(b_2) \\ \text{Use}(x := a) = \text{Use}(a) \quad \text{Use}(\text{skip}) = \emptyset \quad \text{Use}(\text{while } b \text{ do } S) = \text{Use}(S) \cup \text{Use}(b) \\ \text{Use}(S_1; S_2) = \text{Use}(S_1) \cup \text{Use}(S_2) \quad \text{Use}(\text{if } b \text{ then } S_1 \text{ else } S_2) = \text{Use}(S_1) \cup \text{Use}(S_2) \cup \text{Use}(b) \end{aligned}$$

Question 1. Donner le résultat de l'application des fonctions **Def** et **Use** sur la commande suivante : `if a < b then c := d-y else y := e -x.`

1.2 Ajout d'un opérateur de parallélisme, noté ||

On étend la syntaxe des commandes de la manière suivante : $S ::= \dots \mid S \parallel S$. La sémantique de cet opérateur est définie comme suit :

$\frac{(S_1, \sigma) \longrightarrow \sigma_1 \quad (S_2, \sigma_1) \longrightarrow \sigma_2}{(S_1 \parallel S_2, \sigma) \longrightarrow \sigma_2} \quad \frac{(S_2, \sigma) \longrightarrow \sigma_1 \quad (S_1, \sigma_1) \longrightarrow \sigma_2}{(S_1 \parallel S_2, \sigma) \longrightarrow \sigma_2}$

Question 2. En appliquant les règles de sémantique, calculer le ou les états obtenus sachant que l'état initial σ_0 est tel que x et y valent 1.

$$x := x + 1 \parallel y := y + 1$$

Question 3. Même question sur la commande

$x := y + 1 \parallel y := x + 2.$

Question 4. Donner une condition suffisante pour que les 2 règles donnent le même résultat, c'est à dire que, lorsqu'on évalue $S_1 \parallel S_2$ dans un état σ , alors l'état σ_2 obtenu est indépendant de l'ordre d'évaluation de S_1 et de S_2 . (indication : on peut utiliser les fonctions **Def** et **Use**, définies précédemment).

Question 5. Proposer une seule règle de sémantique qui évalue $S_1 \parallel S_2$ dans l'état σ en évaluant S_1 et S_2 dans l'état σ et en vérifiant que la condition précédente est satisfaite.

Question 6. Appliquer cette règle sur la commande de la question 2.

2 Exercice II Une certaine forme de typage

On cherche à détecter par typage les variables non initialisées. Par exemple, dans la commande $i := 1 ; j := 2 ; k := i + j$ les variables sont correctement utilisées. Par contre dans la commande $i := 1 ; j := i + j$, j n'est pas correctement utilisée.

Définition Une variable est *utilisée* si elle apparaît dans une expression en partie droite d'une affectation. Une variable est *définie* si elle apparaît en partie gauche d'affectation.

On s'intéresse dans un premier temps au langage **while** sans la construction \parallel . Un environnement de type est simplement l'ensemble des variables initialisées.

2.1 Syntaxe du langage

Catégories Syntaxiques :

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

Bexp $b ::= \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2$

Stm $\left\{ \begin{array}{l} S ::= x := a \mid \text{skip} \mid S_1 ; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S \\ \quad \mid \text{begin } D_V S \text{ end} \end{array} \right.$

Dec_V $D_V ::= \epsilon \mid \text{var } x ; D_V$

Prg $P ::= \text{begin } D_V S \text{ end}$

On va définir la règle suivante : $\Gamma \vdash S : \Gamma'$ qui signifie que, dans l'environnement Γ toutes les variables utilisées dans S sont correctement initialisées et la commande S produit le nouvel environnement Γ' . On donne les règles pour l'affectation, la commande **skip** et pour la composition séquentielle.

$$\overline{\Gamma \vdash x := a : \Gamma \cup \{x\}} \text{Use}(a) \subseteq \Gamma \quad \overline{\Gamma \vdash \text{skip} : \Gamma} \quad \frac{\Gamma \vdash S_1 : \Gamma_1 \quad \Gamma_1 \vdash S_2 : \Gamma_2}{\Gamma \vdash S_1 ; S_2 : \Gamma_2}$$

Question 1. Donner les règles de typage pour les expressions arithmétiques et booléennes.

Question 2. Donner la règle de typage pour la conditionnelle.

Question 3. Donner la règle de typage pour la construction `while`.

Question 4. Appliquer aux exemples suivants, avec $\Gamma = \emptyset$:

- `x := 1; if x = 0 then y := x + 1 else y := x - 1,`
- `x := 1; if x = 0 then x := x + 1 else y := x - 1,`
- `x := 1; while x ≤ 10 do y := x + y ; x := x + 1.`

Question 5. Donner la règle de typage pour la composition parallèle `||`.

Question 5. Modifier la sémantique standard du langage pour la détection des variables non-initialisées.