Langages et Traducteurs

Sémantique Statique (extensions)

Extensions du langage while

On étend la syntaxe abstraite du langage en considérant maintenant :

- qu'une déclaration peut contenir des *déclarations de procédure*, soit sans paramètre (noté `proc p C`), soit avec un paramètre unique (noté `proc p (y t) C`).
- qu'une commande `C` peut être un *bloc* (une commande avec une déclaration locale, noté `B`), ou un *appel de procédure*, soit sans paramètre (noté `call p`), soit avec un paramètre unique (noté `call p e`);

La syntaxe abstraite complète est donc la suivante :

$$\begin{aligned}
 P & ::= B \\
 B & ::= D C \\
 D & ::= \text{var } x \ t \mid \text{proc } p \ C \mid \text{proc } p \ (y \ t) \ C \mid D ; D \\
 C & ::= x := E \mid C ; C \mid \text{si } E \ \text{alors } C \ \text{sinon } C \mid \text{tantque } E \ C \\
 & \quad \mid B \mid \text{call } p \mid \text{call } p \ e \\
 E & ::= n \mid b \mid x \mid E + E \mid E * E \mid E \ \text{et } E \mid \text{not } E
 \end{aligned}$$

Pour prendre en compte les procédures on étend l'ensemble des types :

$$\begin{aligned}
 \text{Type_de_Base} & = \{ \text{Entier}, \text{Booleen}, \text{Void} \}. \\
 \text{Type} & = \text{Type_de_Base} \cup \{ \{ \text{Proc} \} \times \text{Type_de_Base} \}
 \end{aligned}$$

Le type d'une procédure sans paramètre sera donc noté $(\text{Proc}, \text{Void})$, celui d'une procédure avec paramètre de type t sera noté (Proc, t) .

On introduit également une notion de configuration pour les blocs, et on modifie les configurations utilisées pour les déclarations :

$$\begin{aligned}
 \mathcal{C}_B & \subseteq (B \times \text{Env}) \cup \text{Type} \\
 \mathcal{C}_D & \subseteq (D \times \text{Env}) \cup \text{Type}
 \end{aligned}$$

Enfin, on introduit un opérateur de *composition* de fonctions qui permet de prendre en compte l'imbrication possible des blocs :

$$f1[f2](x) = \begin{cases} f1(x) & \text{si } x \notin \text{Dom}(f2) \\ f2(x) & \text{sinon} \end{cases}$$

Sémantique statique pour les blocs (relation $\xrightarrow{b} \subseteq \mathcal{C}_B \times \mathcal{C}_B$)

Un bloc `D ; C` est correct dans l'env. ρ ssi les déclaration `D` sont correctes dans ρ et produisent l'env. ρ_l et si les commandes `C` sont correctes dans l'env. composé $\rho[\rho_l]$:

$$\frac{\langle D, \rho \rangle \xrightarrow{d} \rho_l \quad \langle C, \rho[\rho_l] \rangle \xrightarrow{c} \text{Void}}{\langle D \ C, \rho \rangle \xrightarrow{b} \text{Void}}$$

Nouvelles règles pour les déclarations (relation $\xrightarrow{d} \subseteq \mathcal{C}_D \times \mathcal{C}_D$)

Une déclaration de variable est toujours correcte :

$$\langle \text{var } x \ t, \rho \rangle \xrightarrow{d} [x \mapsto t]$$

Une déclaration de procédure sans paramètre p est correcte dans un env. ρ ssi son corps C est correct dans cet environnement ρ étendu avec la définition de p (pour prendre en compte un possible appel récursif sur p) :

$$\frac{\langle C, \rho[p \mapsto (Proc, Void)] \rangle \xrightarrow{c} Void}{\langle \text{proc } p \ C, \rho \rangle \xrightarrow{d} [p \mapsto (Proc, Void)]}$$

Pour une procédure p avec un paramètre y il faut en plus étendre ρ avec la définition de y :

$$\frac{\langle C, \rho[p \mapsto (Proc, t), y \mapsto t] \rangle \xrightarrow{c} Void}{\langle \text{proc } p \ (y \ t) \ C, \rho \rangle \xrightarrow{d} [p \mapsto (Proc, t)]}$$

Lorsque deux déclarations $D1$ et $D2$ ont lieu en séquence, on “propage” l’environnement généré par $D1$ pour vérifier $D2$ (ainsi les variables ou procédures déclarées dans $D1$ sont définies lorsque l’on vérifie $D2$) :

$$\frac{\langle D1, \rho \rangle \xrightarrow{d} \rho1 \quad \langle D2, \rho[\rho1] \rangle \xrightarrow{d} \rho2 \quad Dom(\rho1) \cap Dom(\rho2) = \emptyset}{\langle D1 ; D2, \rho \rangle \xrightarrow{d} \rho1 \cup \rho2}$$

Nouvelles règles pour les commandes (relation $\xrightarrow{c} \subseteq \mathcal{C}_C \times \mathcal{C}_C$)

Une commande de type “bloc” (B) est correcte ssi le bloc correspondant est correct :

$$\frac{\langle B, \rho \rangle \xrightarrow{b} Void}{\langle B, \rho \rangle \xrightarrow{c} Void}$$

Un appel de procédure sans paramètre est correct dans l’env. ρ ssi cette procédure est définie dans ρ :

$$\frac{x \in Dom(\rho) \quad \rho(x) = (Proc, Void)}{\langle \text{call } x, \rho \rangle \xrightarrow{c} Void}$$

Un appel de procédure avec paramètre est correct dans l’env. ρ ssi cette procédure est définie dans ρ , si le paramètre effectif est correct, et si son type correspond à celui du paramètre formel :

$$\frac{x \in Dom(\rho) \quad \rho(x) = (Proc, t) \quad \langle e, \rho \rangle \xrightarrow{e} t}{\langle \text{call } x \ e, \rho \rangle \xrightarrow{c} Void}$$

Enfin, la correction d’un programme est maintenant définie par la règle suivante :

$$\frac{\langle B, \emptyset \rangle \xrightarrow{b} Void}{P \longrightarrow Void}$$