Éléments de correction du TD9 - Sémantique Statique (2)

Exercice 1

Programme 1

On commence par la règle du programme, qui fournit une branche pour la déclaration des variables/procédures, une branche pour l'utilisation de ces variables :

$$\dfrac{\dfrac{\overset{\text{Branche 1}}{< D, \emptyset > \xrightarrow{d} \rho_l} \qquad \overset{\text{Branche 2}}{< C, \emptyset[\rho_l] > \xrightarrow{c} \text{Void}}}{< B, \emptyset > \xrightarrow{b} \text{Void}}\ [\text{bloc}]}{B \rightarrow \text{Void}}\ [\text{prog}]$$

Traitons le cas des déclarations des variables et procédures :

$$\dfrac{\dfrac{\rho_1 : [x_1 \mapsto \text{Entier}]}{\text{var } x_1\text{Entier} \xrightarrow{d} \rho_1} \qquad \dfrac{\dfrac{\rho_3 : [x_2 \mapsto \text{Booléen}]}{\text{var } x_2\text{Booléen} \xrightarrow{d} \rho_3} \qquad < DeclPROC, \rho_3 > \xrightarrow{e} \rho_4}{(\text{var } x_2\text{Booléen}; D_3) \xrightarrow{d} \rho_2 \qquad Dom(\rho_1) \cap Dom(\rho_2) = \emptyset \quad \rho_2 = \rho_3 \cup \rho_4}\ [\text{seq}]}{(\text{var } x_1\text{Entier}; D_2) \xrightarrow{d} \rho_l \qquad \rho_l = \rho_1 \cup \rho_2}\ [\text{seq}]$$

On obtient facilement $\rho_l = \begin{cases} x_1 \mapsto \text{Entier} \\ x_2 \mapsto \text{Booléen} \\ p1 \mapsto (\text{Proc}, \text{Booléen}) \end{cases}$ tout en vérifiant que $x_2 := x_1$ est bien typé dans l'environnement $\rho_3[p_1 \mapsto (\text{Proc}, \text{Booléen}), x_1 \mapsto \text{Booléen}$. (sous-arbre à faire).

REMARQUE 1 La sémantique impose que x_1 soit typé Entier à l'extérieur de l'appel de la fonction p_1, mais Booléen à l'intérieur.

Maintenant, traitons la branche "commande" qui est réduite à un call :

$$\dfrac{\overset{\text{OK}}{< p_1 \in Dom(\rho_l) >} \qquad \overset{\text{OK}}{\rho_l(p_1) = (Proc, \text{Booléen})} \qquad \dfrac{x_2 \in Dom(\rho_l)}{< x_2, \rho_l > \xrightarrow{e} \text{Booléen}}\ \text{car } \rho_l(x_2) = \text{Booléen}}{< call\ p_1\ x_2, \rho_l > \xrightarrow{c} Void}$$

Les dernières sous-branches ne sont pas difficiles, et permettent de clore l'arbre, le programme est donc bien typé.

Programme 2

On essaie de typer, et on s'aperçoit que on ne peut pas boucler l'arbre, on a un souci au niveau de la déclaration de la procédure p_1, on ne peut pas dire que l'appel `call p2` est bien typé car on ne dispose pas du type de p_2 à ce moment-là :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{< call\ p_2, [p_1 \mapsto (\text{Proc}, \text{Void})] > \xrightarrow{c} \text{Void}} \quad \text{[decl proc] avec } \rho_l = [p_1 \mapsto (\text{Proc}, Void)]}{< proc\ p_1\ call(p_2), \emptyset > \xrightarrow{d} \rho_1} \quad \dots \quad \dots}{< D, \emptyset > \xrightarrow{d} \rho_l} \quad \text{[seq]} \qquad < C, \rho_l > \xrightarrow{c} \text{Void}}{< D; C, \emptyset > \xrightarrow{b} \text{Void}} \quad \text{[bloc]}}{prog \rightarrow \text{Void}} \quad \text{[prog]}$$

On voit que pour faire en sorte que le programme soit bien typé, il faut faire une première passe où l'on construit les types $(Proc, t)$ et ensuite une deuxième phase où l'on vérifie que les corps des procédures sont bien typées dans ce "gros" environnement. Pour réaliser cela, il nous faut de nouvelles règles :

Règle du bloc :

$$\frac{< D, \rho > \xrightarrow{d} \rho_l \quad < D, \rho_l > \xrightarrow{new} OK \quad < C, \rho[\rho_l] > \xrightarrow{c} \text{Void}}{< D; C, \rho > \xrightarrow{b} \text{Void}}$$

Il faut donc modifier les règles de déclaration, et créer les règles de vérification :

Déclaration : tout est un axiome :

$$< proc\ x\ S, \rho > \xrightarrow{d} [x \mapsto (Proc, Void)]$$

$$< proc\ x(y\ t)\ S, \rho > \xrightarrow{d} [x \mapsto (Proc, t)]$$

Et ensuite, on doit vérifier que tout va bien, par exemple pour la déclaration d'une procédure sans paramètre :

$$\frac{< S, \rho[x \mapsto (Proc, Void)] > \xrightarrow{e} \text{Void}}{< proc\ x\ S, \rho > \xrightarrow{new} OK}$$

Il reste à rajouter les autres règles et à typer le programme exemple pour vérifier que cela marche. Cet exercice est laissé au lecteur.

Exercice 2

On doit rajouter des règles de type "déclaration" pour autoriser l'initialisation, ie on doit pouvoir tester si cette initialisation a un sens à ce moment là :

$$\frac{< e, \rho > \xrightarrow{e} t}{< var\ x\ :\ t\ := e\ ,\ \rho > \xrightarrow{e} [x \mapsto t]}$$

On vérifie que la séquence des déclarations fait bien passer les indices de type.