Éléments de correction du TD10 - Sémantique Dynamique

# Exercice 1

La structure des preuves en sémantique dynamique est très similaire à celle de sémantique statique pour le même programme, sauf qu'il faut calculer la valeur de la mémoire à tous stades du programme.

## Programme 1

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

$$\dfrac{\overset{\text{Branche 1}}{d \xrightarrow{d} \eta} \qquad\qquad \overset{\text{Branche 2}}{< c, \eta, \emptyset > \xrightarrow{c} \sigma}}{d; c \to \sigma}\ \text{[prog]}$$

Traitons ensuite le cas de la branche 1 (déclarations) :

$$\dfrac{\dfrac{@1 = newadr()}{\text{var } x_1 \xrightarrow{d} \eta_1 = [x_1 \mapsto @1]} \qquad \dfrac{@2 = newadr()}{\text{var } x_2 \xrightarrow{d} \eta_2 = [x_2 \mapsto @2]}}{(\text{var } x_1; \text{var } x_2) \xrightarrow{d} \eta = \eta_1 \cup \eta_2}\ \text{[seq]}$$

Toutes les feuilles sont des axiomes et les applications des règles sont correctes ssi $\eta = [x_1 \mapsto @1, x_2 \mapsto @2]$. Traitons maintenant le cas des commandes avec $\eta$ comme environnement.

$$\dfrac{\text{[aff]}\ \dfrac{\overset{\text{axiome ssi } v_1 = 3}{< 3, \eta, \emptyset > \xrightarrow{e} v_1}}{< x_1 := 3, \eta, \emptyset > \xrightarrow{c} \sigma'_0} \qquad \dfrac{\dfrac{\overset{\text{axiome ssi } v_2 = 1}{< 1, \eta, \sigma'_0 > \xrightarrow{e} v_2}}{< x_2 := 1, \eta, \sigma'_0 > \xrightarrow{c} \sigma'_1} \quad \dfrac{\overset{\text{Branche 2b}}{< \text{tq } ec, \eta, \sigma'_1 > \xrightarrow{c} \sigma'_2} \overset{\text{Branche 2c}}{< x_2 := 2, \eta, \sigma'_2 > \xrightarrow{c} \sigma}}{< \text{tq } ec; ..., \eta, \sigma'_1 > \xrightarrow{c} \sigma}\ \text{[seq]}}{< x_2 := 1; ..., \eta, \sigma'_0 > \xrightarrow{c} \sigma}\ \text{[seq]}}{< x_1 := 3; x_2 := 1; \text{tq } ec; x_2 := 2, \eta, \emptyset > \xrightarrow{c} \sigma}\ \text{[seq]}$$

On écrit dans un coin au fur et à mesure les indications que l'on récupère avec les applications des règles :
- $\sigma'_0 = \emptyset[@1 \mapsto v_1]$ ;
- $v_1 = 3$, donc $\sigma'_0 = [@1 \mapsto 3]$ ;
- $\sigma'_1 = \sigma'_0[@2 \mapsto v_2] = [@1 \mapsto 3, @2 \mapsto 1]$ ;

Réalisons maintenant la branche 2b (la sémantique du test $x_1 > 0$ n'est pas dans le cours, mais on fait avec !) :

$$\dfrac{\dfrac{\text{OK car}\sigma'_1(\eta(x_1))>0}{<x_1>0,\eta,\sigma'_1> \xrightarrow{e} tt} \quad \dfrac{\dfrac{\text{OK car } \sigma'_1(\eta(x_2))=1}{<x_2>2,\eta,\sigma'_1> \xrightarrow{e} ff} \quad \dfrac{\dfrac{<x_1,\eta,\sigma'_1> \xrightarrow{e} 3 \qquad <2,\eta,\sigma'_1> \xrightarrow{e} 2}{<x_1-2,\eta,\sigma'_1> \xrightarrow{e} v_4}[\text{moins}]}{<x_1 := x_1-2,\eta,\sigma'_1> \xrightarrow{c} \sigma'_3}[\text{aff}]}{<\text{ si ... alors ... sinon ...},\eta,\sigma'_1> \xrightarrow{c} \sigma'_3} \quad \dfrac{\text{Branche2balpha}}{<\text{ tq } x_1>0 \text{ si ... alors ... sinon ...},\eta,\sigma'_3> \xrightarrow{c} \sigma'_2}}{<\text{ tq } x_1>0 \text{ si ... alors ... sinon ...},\eta,\sigma'_1> \xrightarrow{c} \sigma'_2}[\text{while}]$$

La branche 2balpha est laissée au lecteur. A ce stade, on a accumulé les indications intermédiaires :
- $v_4 = 3\ minus\ 2 = 1$ ;
- $\sigma'_3 = \sigma'_1[@1 \mapsto v_4 = 1]$ ;
- $\sigma'_2 = \sigma'_3[@1 \mapsto 1-2 = -1]$.

Il reste maintenant la branche 2c :

$$\frac{<2,\eta,\sigma'_2> \xrightarrow{e} 2}{<x_2 := 2,\eta,\sigma'_2> \xrightarrow{c} \sigma}[\text{ aff }],$$

avec l'information suivante : $\sigma = \sigma'_2[\eta(x_2) \mapsto 2]$. Et finalement on trouve : $\sigma \circ \eta : [x_2 \mapsto 2, x_1 \mapsto -1]$.

### Programme 2

Le déroulement des règles est similaire, sauf que l'on ne peut pas construire d'arbre de preuve fini.

## Exercice 2

1. Expressions : on veut pouvoir effectuer une évaluation paresseuse de $e_1 \wedge e_2$, d'où le `etpuis` :

$$\frac{<e_1,\eta,\sigma> \xrightarrow{e} ff}{<e_1\ etpuis\ e_2,\eta,\sigma> \xrightarrow{e} ff} \quad \frac{<e_1,\eta,\sigma> \xrightarrow{e} tt \quad <e_2,\eta,\sigma> \xrightarrow{e} b}{<e_1\ etpuis\ e_2,\eta,\sigma> \xrightarrow{e} b}.$$

2. Commandes : on fait une première fois la commande $c$, et ensuite on teste si on doit continuer (valeur de l'expression $e$ dans $\sigma'$) :

$$\frac{<c,\eta,\sigma> \xrightarrow{c} \sigma' \quad <e,\eta,\sigma'> \xrightarrow{e} tt}{<repeter\ c\ jusqua\ e,\eta,\sigma> \xrightarrow{c} \sigma'}$$

$$\frac{<c,\eta,\sigma> \xrightarrow{c} \sigma' \quad <e,\eta,\sigma'> \xrightarrow{e} ff \quad <repeter\ c\ jusqua\ e,\eta,\sigma'> \xrightarrow{c} \sigma"}{<repeter\ c\ jusqua\ e,\eta,\sigma> \xrightarrow{c} \sigma"}$$

3. Déclarations : Il faut calculer ce que vaut l'expression $e$ dans l'environnement courant, et ensuite mettre cette expression à la nouvelle adresse, pour cela on s'aperçoit que l'on doit maintenant se rappeler de la mémoire courante, et de l'environnement aussi, et on modifie à la fois la valeur de l'environnement et la valeur de la mémoire :

$$\frac{ad = newadr() \quad <e,\eta,\sigma> \xrightarrow{e} v}{(\text{var } x : Entier := e,\eta,\sigma) \xrightarrow{d} ([x \mapsto ad], \sigma[ad \mapsto v])}$$

Il faudrait donc changer la configuration des déclarations pour prendre en compte cette nouvelle sémantique, et aussi prendre en compte ceci lors de la règle des blocs/ règle du programme.

## Exercice 3

Flemme du correcteur, on trouve à la fin : $\sigma \circ \eta : [x_2 \mapsto 1, x_1 \mapsto 4]$. Attention à la sortie du bloc les variables sont restaurées.