LIFLC – Logique classique
TD7 – Sémantique de *Imp*

Licence informatique UCBL – Automne 2017–2018

Les (parties d') exercices noté(e)s avec † sont plus difficiles.

On rappelle les définitions de structure du langage *Imp* :

Définition 1. *Les expressions sont vues comme des termes construits sur les signatures suivantes :*

— *Expressions arithmétiques (*$\mathtt{aexp}$*) :*

$\mathcal{C}_{\mathtt{aexp}}$ **:** *entiers naturels représentés en base 10 (notés* $\overline{n}$*)*

$\mathcal{F}_{\mathtt{aexp}}$ **:** $\{\mathtt{APlus}/2, \mathtt{AMinus}/2, \mathtt{AMult}/2\}$

— *Expressions booléennes (*$\mathtt{bexp}$*) :*

$\mathcal{C}_{\mathtt{bexp}}$ **:** $\{\mathtt{BTrue}, \mathtt{BFalse}\}$

$\mathcal{F}_{\mathtt{bexp}}$ **:** $\{\mathtt{BEq}/2, \mathtt{BLe}/2, \mathtt{BNot}/1, \mathtt{BAnd}/2\}$

$\{\mathtt{BEq}/2, \mathtt{BLe}/2$ *: ce sont symboles particuliers : leurs arguments sont construits sur* $\mathtt{aexp}$

Définition 2. *Les programmes sont vus comme un ensemble* $\mathtt{prog}$ *inductif dont les constructeurs sont :*

— $\mathtt{CSkip}$

— $\mathtt{CAss}(x, e)$ *si* $x \in \mathcal{V}$ *et* $e \in \mathtt{aexp}$

— $\mathtt{CSeq}(p_1, p_2)$ *si* p_1 *et* p_2 *sont des programmes*

— $\mathtt{CIf}(b, p_1, p_2)$ *si* $b \in \mathtt{bexp}$ *et si* p_1 *et* p_2 *sont des programmes*

— $\mathtt{CWhile}(b, p)$ *si* $b \in \mathtt{bexp}$ *et si* p *est un programme.*

Définition 3. *Les expressions (*$\mathtt{aexp}$ *et* $\mathtt{bexp}$*) sont évaluées (dans les entiers naturels pour les expressions arithmétiques et dans {true,false} pour les expressions booléennes) en utilisant l'interprétation I suivante :*

— *Constantes entières :* $I(\overline{n}) = \mathit{parse}(\overline{n})$ *où parse est la fonction qui la fonction qui prend une suite de chiffres en base 10 et les convertis en nombre entier.*

— $I(\mathtt{APlus}) = n_1, n_2 \mapsto n_1 + n_2$

— $I(\mathtt{AMinus}) = n_1, n_2 \mapsto n_1 - n_2$

— $I(\mathtt{AMult}) = n_1, n_2 \mapsto n_1 \times n_2$

— *Constantes :* $I(\mathtt{BTrue}) = \mathit{true}$, $I(\mathtt{BFalse}) = \mathit{false}$

— $I(\mathtt{BEq}) : (n_1, n_2) \mapsto \begin{cases} \mathit{true} \text{ si } n_1 = n_2 \\ \mathit{false} \text{ sinon} \end{cases}$

— $I(\mathtt{BLe}) : (n_1, n_2) \mapsto \begin{cases} \mathit{true} \text{ si } n_1 \leq n_2 \\ \mathit{false} \text{ sinon} \end{cases}$

— $I(\mathtt{BNot}) : b \mapsto \begin{cases} \mathit{true} \text{ si } b = \mathit{false} \\ \mathit{false} \text{ sinon} \end{cases}$

— $I(\texttt{BLe}) : (b_1, b_2) \mapsto \begin{cases} \textit{true} \text{ si } b_1 = \textit{true} \text{ et } b_2 = \textit{true} \\ \textit{false} \text{ sinon} \end{cases}$

Définition 4. *La sémantique opérationnelle de Imp est donnée comme une transformation d'état, de la forme $P : \zeta \rightsquigarrow \zeta'$, qui exprime que P transforme ζ en ζ'. La construction de cette relation est faite à travers les règles données par la figure 1.*

$$\frac{}{\texttt{CSkip} : \zeta \rightsquigarrow \zeta} \ (\mathit{Imp}_{\mathit{Skip}})$$

$$\frac{}{\texttt{CAss}(x, a) : \zeta \rightsquigarrow \zeta[x := n]} \ (\mathit{Imp}_{\mathit{Ass}}) \quad \text{si } \texttt{aeval}(\zeta)(e) = n$$

$$\frac{P_1 : \zeta \rightsquigarrow \zeta' \quad P_2 : \zeta' \rightsquigarrow \zeta''}{\texttt{CSeq}(P_1, P_2) : \zeta \rightsquigarrow \zeta''} \ (\mathit{Imp}_{\mathit{Seq}})$$

$$\frac{P_1 : \zeta \rightsquigarrow \zeta'}{\texttt{CIf}(b, P_1, P_2) : \zeta \rightsquigarrow \zeta'} \ (\mathit{Imp}_{\mathit{IfTrue}}) \quad \text{si } \texttt{beval}(\zeta)(b) = \textit{true}$$

$$\frac{P_2 : \zeta \rightsquigarrow \zeta'}{\texttt{CIf}(b, P_1, P_2) : \zeta \rightsquigarrow \zeta'} \ (\mathit{Imp}_{\mathit{IfFalse}}) \quad \text{si } \texttt{beval}(\zeta)(b) = \textit{false}$$

$$\frac{}{\texttt{CWhile}(b, P) : \zeta \rightsquigarrow \zeta} \ (\mathit{Imp}_{\mathit{WhileFalse}}) \quad \text{si } \texttt{beval}(\zeta)(b) = \textit{false}$$

$$\frac{P : \zeta \rightsquigarrow \zeta' \quad \mathit{CWhile}(b, P) : \zeta' \rightsquigarrow \zeta''}{\texttt{CWhile}(b, P) : \zeta \rightsquigarrow \zeta''} \ (\mathit{Imp}_{\mathit{WhileTrue}})$$

FIGURE 1 – Règle de sémantique opérationnelle pour *Imp*

Exercice 1 : Optimisation des multiplications

1. Écrire une fonction d'optimisation des expressions arithmétiques utilisant le fait que 1 est élément neutre et 0 élément absorbant du produit dans les entiers.
2. Énoncer et prouver un théorème de correction de cette fonction d'optimisation.

Exercice 2 : Évaluation de programme en utilisant la sémantique formelle de *Imp*

Soit le programme *Imp* suivant :

```
1  Y ::= 1;;
2  WHILE Y * Z <= X DO
3     Y ::= Y + 1
4  END;;
5  Y ::= Y - 1
```

1. Donner l'arbre de syntaxe abstraite de ce programme. Nommer chacun de ses nœuds de façon à pouvoir facilement y faire référence par la suite. La racine de l'arbre sera nommée `P`.

2. Soit

$$\zeta : \begin{array}{lcl} X & \mapsto & 9 \\ Y & \mapsto & 9 \\ Z & \mapsto & 4 \end{array}$$

   Trouver ζ' telle que $P : \zeta \rightsquigarrow \zeta'$ et exhibant la dérivation appropriée.

3. Supposons que la ligne 2 du programme soit changée en :

$$\text{WHILE X} <= \text{Y} * \text{Z DO}$$

   Exhiber un état $\zeta^{\uparrow}$ initial pour lequel ce programme ne termine pas. Expliquer ce qui ce passe si on essaie de construire une dérivation de ce programme avec $\zeta^{\uparrow}$ comme état initial.

Exercice 3 : † Propagation des constantes

Soit $\overline{n}$ la représentation d'un entier en base 10 et soit $n = \mathit{parse}(\overline{n})$. Soit la substitution $\sigma = [{}^{\overline{n}}/_{X}]$. Soit P un programme *Imp*ne contenant aucune affectation sur la variable X.

1. Montrer que pour toute expression $e \in \texttt{aexp}$, si $\zeta(X) = n$ alors $\mathit{eval}(I, \zeta)(e) = \mathit{eval}(I, \zeta)(e\sigma)$.

2. Définir par induction $P\sigma$, le programme P dans lequel toutes les expressions se sont vues appliquées la substitution σ.
3. Montrer que $\texttt{X::=}\overline{n}\ \texttt{;;}\ P$ et $P\sigma$ sont équivalent, c'est-à-dire que $\texttt{CSeq}(\texttt{CAss}(X, \overline{n}), P) : \zeta \rightsquigarrow \zeta'$ si et seulement si $P\sigma : \zeta \rightsquigarrow \zeta'$.