LIFLC – Logique classique
Sémantique de *Imp*

Licence informatique UCBL – Automne 2018–2019

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 :* $I(\overline{n}) = parse(\overline{n})$ *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 :* $I(\mathtt{BTrue}) = true$
 $I(\mathtt{BFalse}) = false$

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

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

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

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

Comme I est fixée, on peut introduire des version spécifique de l'évaluation des expressions :

— $\textit{aeval}(\zeta)(e) = \textit{eval}(I, \zeta)(e)$ *si* $e \in \mathit{aexp}$

— $\textit{beval}(\zeta)(b) = \textit{eval}(I, \zeta)(b)$ *si* $b \in \mathit{bexp}$.

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.*

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

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

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

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

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

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

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

FIGURE 1 – Règle de sémantique opérationnelle pour *Imp*