LIFLC – Logique classique
TD7 – Sémantique de *Imp*

Licence informatique UCBL – Automne 2018–2019

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

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
$$\zeta : \begin{array}{rcl} X & \mapsto & 9 \\ Y & \mapsto & 9 \\ Z & \mapsto & 4 \end{array}$$
Trouver ζ' telle que $P : \zeta \rightsquigarrow \zeta'$ et exhiber la dérivation appropriée.
3. Supposons que la ligne 2 du programme soit changée en :
$$\texttt{WHILE X <= Y * Z DO}$$
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 $\sigma = [X := \overline{n}]$. Soit P un programme *Imp* ne contenant aucune affectation sur la variable X.

1. Montrer que pour toute expression $e \in \texttt{aexp}$ et toute valuation ζ :
$$\mathit{eval}(I, \zeta[X := n])(e) = \mathit{eval}(I, \zeta)(e\sigma)$$

.

2. Définir par induction $P\sigma$, le programme P dans lequel toutes les expressions se sont vues appliquées la substitution σ.
3. Montrer que $\texttt{X::=}\overline{n}\ \texttt{;;}\ P$ fait la même chose que $P\sigma$, sauf pour la valeur finale de X. Plus précisément, montrer que $P\sigma : \zeta \rightsquigarrow \zeta'$, alors $\texttt{CSeq}(\texttt{CAss}(X, \overline{n}), P) : \zeta \rightsquigarrow \zeta'[X := n]$.

Corrections

Solution de l'exercice 1

1. on défini *optM* comme la fonction d'optimisation de la multiplication.

   Définition 1. *La fonction récursive optM* : $\mathtt{aexp} \to \mathtt{aexp}$ *est définie par cas :*
   — $optM(n) = n$ *si* $n \in \mathcal{C}_{aexp}$
   — $optM(x) = x$ *si* $x \in \mathcal{V}$
   — $optM(\mathtt{AMult}(1, e_2) = optM(e_2)$
   — $optM(\mathtt{AMult}(e_1, 1) = optM(e_1)$
   — $optM(\mathtt{AMult}(0, e_2) = 0$
   — $optM(\mathtt{AMult}(e_1, 0) = 0$
   — $optM(\mathtt{AMult}(e_1, e_2) = \mathtt{AMult}(optM(e_1), optM(e_2))$ *si* $e_1, e_2 \notin \{0, 1\}$
   — $optM(\mathtt{APlus}(e_1, e_2)) = \mathtt{APlus}(optM(e_1), optM(e_2))$
   — $optM(\mathtt{AMinus}(e_1, e_2)) = \mathtt{AMinus}(optM(e_1), optM(e_2))$

   Remarque : on pourrait optimiser un peu plus : au lieu de vérifier qu'un argument est 0 (ou 1), on pourrait faire cette vérification sur sa version optimisée.
2. Lé théorème de correction indique que la version optimisée s'évalue comme la version de départ (i.e. donne les mêmes résultats) :

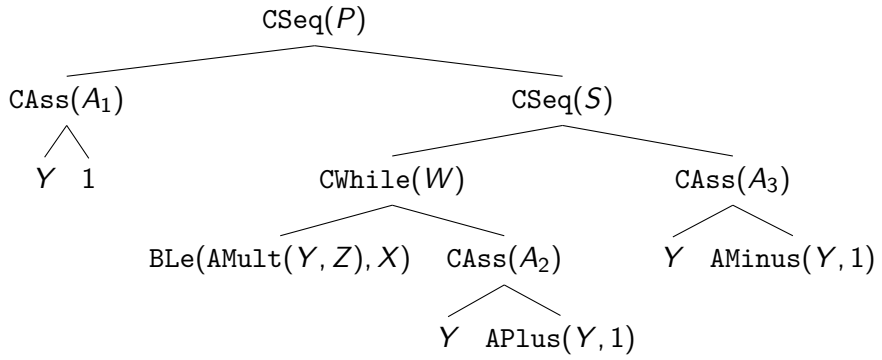
   Théorème 1. *Pour toute expression* $e \in \mathtt{aexp}$*, pour toute valuation* ζ*,*
   $eval(I, \zeta)(e) = eval(I, \zeta)(optM(e))$

   Démonstration. Par induction et par cas sur e :
   — si $e = n$ avec $n \in \mathcal{C}_{\mathtt{aexp}}$ ou si $e = x$ avec $x \in \mathcal{V}$, alors $optM(e) = e$,
   donc $eval(I, \zeta)(e) = eval(I, \zeta)(optM(e))$
   — si $e = \mathtt{AMult}(e_1, e_2)$, alors $eval(I, \zeta)(e)(e) = I(\mathtt{AMult})(eval(I, \zeta)(e_1), eval(I, \zeta)(e_2)) = eval(I, \zeta)(e_1) \times eval(I, \zeta)(e_2)$. On procède par cas sur e_1 et e_2 :
      — si $e_1 = 0$, alors $eval(I, \zeta)(e_1) = 0$, donc $eval(I, \zeta)(e) = 0 = eval(I, \zeta)(0) = eval(I, \zeta)(optM(e))$
      — si $e_2 = 0$: ce cas est similaire au précédent
      — si $e_1 = 1$, alors $eval(I, \zeta)(e_1) = 1$, donc $eval(I, \zeta)(e) = eval(I, \zeta)(e_2)$.
      Hypothèse d'induction sur e_2 : $eval(I, \zeta)(e_2) = eval(I, \zeta)(optM(e_2))$.
      Donc $eval(I, \zeta)(e) = eval(I, \zeta)(optM(e_2)) = eval(I, \zeta)(optM(e))$.
      — si $e_2 = 1$: ce cas est similaire au précédent
      — sinon :
      Hypothèse d'induction sur e_1 : $eval(I, \zeta)(e_1) = eval(I, \zeta)(optM(e_1))$.
      Hypothèse d'induction sur e_2 : $eval(I, \zeta)(e_2) = eval(I, \zeta)(optM(e_2))$.
      $eval(I, \zeta)(e)(e) = eval(I, \zeta)(e_1) \times eval(I, \zeta)(e_2)$
      $= eval(I, \zeta)(optM(e_1)) \times eval(I, \zeta)(optM(e_2)) = eval(I, \zeta)(optM(e))$.
   — les autres cas (`APlus` et `AMinus`) sont similaires au dernier sous-cas précédent.

   □

Solution de l'exercice 2

1. Ici, on a décidé de ne pas représenter les arbres de syntaxe des expressions.



2. Si on déroule à la main le programme, on obtient

$$\begin{array}{rrcl} \zeta' : & X & \mapsto & 9 \\ & Y & \mapsto & 2 \\ & Z & \mapsto & 4 \end{array}$$

i.e. $\zeta' = \zeta[Y := 2]$. Afin de réduire l'écriture de la dérivation, on pose $\zeta_i = \zeta[Y := i]$. Ainsi $\zeta = \zeta_9$ et $\zeta' = \zeta_2$. La dérivation qui le montre est la suivante :

$$\dfrac{\dfrac{}{A_1 : \zeta_9 \rightsquigarrow \zeta_1}\ (Imp_{Ass}) \quad \dfrac{\dfrac{D}{W : \zeta_1 \rightsquigarrow \zeta_3} \quad \dfrac{}{A_3 : \zeta_3 \rightsquigarrow \zeta_2}\ (Imp_{Ass})}{S : \zeta_1 \rightsquigarrow \zeta_2}\ (Imp_{Seq})}{P : \zeta_9 \rightsquigarrow \zeta_2}\ (Imp_{Seq})$$

avec D la dérivation suivante :

$$\dfrac{\dfrac{}{A_2 : \zeta_1 \rightsquigarrow \zeta_2}\ (Imp_{Ass}) \quad \dfrac{\dfrac{}{A_2 : \zeta_2 \rightsquigarrow \zeta_3}\ (Imp_{Ass}) \quad \dfrac{}{W : \zeta_3 \rightsquigarrow \zeta_3}\ (Imp_{WhileFalse})}{W : \zeta_2 \rightsquigarrow \zeta_3}\ (Imp_{WhileTrue})}{W : \zeta_1 \rightsquigarrow \zeta_3}\ (Imp_{WhileTrue})$$

3. Si on prend l'état $\zeta^{\uparrow} = \zeta[Z := 10]$, on constate que le programme ne termine pas. Lorsqu'on essaie de construire la dérivation, on va se mettre à empiler les applications de $(Imp_{WhileTrue})$ à l'infini : le programme boucle et on ne sait pas donner de dérivation finie à partir de $P : \zeta^{\uparrow} \rightsquigarrow \zeta''$ quelque soit le choix de ζ''.

Solution de l'exercice 3 Dans la correction de cet exercice, les (représentations entières d') entiers dans les expressions sont notés $\overline{n}$ et l'entier naturel correspondant est noté n.

1. *Démonstration.* Supposons $\zeta(X) = n$. Par induction et par cas sur e :
   — Si $e = \overline{n_1}$, $e\sigma = e$, donc $eval(I, \zeta[X := n])(\overline{n_1}) = n_1 = eval(I, \zeta)(e\sigma)$.
   — Si $e = X$, alors $e\sigma = \overline{n}$. $eval(I, \zeta[X := n])(X) = \zeta[X := n](X) = n = eval(I, \zeta)(\overline{n}) = eval(I, \zeta)(e\sigma)$.
   — Si $e = Y$, avec $Y \neq X$, alors $e\sigma = \overline{n}$. $eval(I, \zeta[X := n])(Y) = \zeta[X := n](Y) = \zeta(Y) = eval(I, \zeta)(Y) = eval(I, \zeta)(e\sigma)$.
   — Si $e = \texttt{APlus}(e_1, e_2)$:
   Hypothèse d'induction sur e_1 : $eval(I, \zeta[X := n])(e_1) = eval(I, \zeta)(e_1\sigma)$
   Hypothèse d'induction sur e_2 : $eval(I, \zeta[X := n])(e_2) = eval(I, \zeta)(e_2\sigma)$
   $eval(I, \zeta[X := n])(\texttt{APlus}(e_1, e_2))$
   $= I(\texttt{APlus})(eval(I, \zeta[X := n])(e_1), eval(I, \zeta[X := n])(e_2))$
   $= I(\texttt{APlus})(eval(I, \zeta)(e_1\sigma), eval(I, \zeta)(e_2\sigma)) = eval(I, \zeta)(\texttt{APlus}(e_1\sigma, e_2\sigma)) = eval(I, \zeta)(e)$

— Les cas $e = \texttt{AMult}(e_1, e_2)$ et $e = \texttt{AMinus}(e_1, e_2)$ sont similaire au cas précédent.

□

2. On procède récursivement et par cas sur la structure du programme. Attention, dans le cas où il y a une affectation sur une variable du domaine de σ, $P\sigma$ n'est pas définie (*i.e.* on ne peut pas appliquer σ). Cela se justifie par la remarque suivante :
   — Si $\sigma(X)$ est un terme composite, on pourrait remplacer le membre gauche de l'affectation par quelque chose qui n'est pas une variable, ce qui est interdit par la grammaire de *Imp*.
   — Si on ne fait pas ce remplacement, alors on aura remplacé la variable ailleurs (plus loin dans l'exécution du programme) et la valeur de la nouvelle expression a toutes les chances d'être différente de la valeur qui avait été affectée à la variable.

   Définition 2. *Soit P un programme Imp. L'application de la substitution σ sur P, notée $P\sigma$, est récursivement définie par :*
   — $\texttt{CSkip}\ \sigma = \texttt{CSkip}$
   — $\texttt{CAss}(Y, e)\sigma = \texttt{CAss}(Y, e\sigma)$ *si* $Y \notin dom(\sigma)$.
   — $\texttt{CSeq}(P_1, P_2)\sigma = \texttt{CSeq}(P_1\sigma, P_2\sigma)$.
   — $\texttt{CIf}(b, P_1, P_2)\sigma = \texttt{CIf}(b\sigma, P_1\sigma, P_2\sigma)$
   — $\texttt{CWhile}(b, P_1)\sigma = \texttt{CWhile}(b\sigma, P_1\sigma)$

3. La démonstration se base sur deux lemmes intermédiaires : un lemme similaire à la question 1, mais sur les expressions booléennes et un lemme sur les dérivations de sémantique de *Imp*. Ce dernier constitue le cœur de la démonstration de la question 3.

   Lemme 1. *Pour tout $b \in \mathtt{bexp}$ et toute valuation ζ, $eval(I, \zeta[X := n])(b) = eval(I, \zeta)(b\sigma)$.*

   Démonstration. Par induction et par cas sur b.
   — $b = \texttt{BEq}(e_1, e_2)$. D'après la question 1, $eval(I, \zeta[X := n])(e_1) = eval(I, \zeta)(e_1\sigma)$ et $eval(I, \zeta[X := n])(e_2) = eval(I, \zeta)(e_2\sigma)$.
   Donc $eval(I, \zeta[X := n])(\texttt{BEq}(e_1, e_2))$
   $= I(\texttt{BEq})(eval(I, \zeta[X := n])(e_1), eval(I, \zeta[X := n])(e_2))$
   $= I(\texttt{BEq})(eval(I, \zeta)(e_1\sigma), eval(I, \zeta)(e_2\sigma))$
   $= eval(I, \zeta)(\texttt{BEq}(e_1, e_2\sigma)$.
   — $b = \texttt{BLe}(e_1, e_2)$. Ce cas est similaire au précédent.
   — $b = \texttt{BTrue}$ ou $b = \texttt{BFalse}$. Il suffit de remarquer que $b = b\sigma$.
   — $b = \texttt{BNot}(b_1)$.
   Hypothèse d'induction sur b_1 : $eval(I, \zeta[X := n])(b_1) = eval(I, \zeta)(b_1\sigma)$.
   $eval(I, \zeta[X := n])(\texttt{BNot}(b_1))$
   $= I(\texttt{BNot})(eval(I, \zeta[X := n])(b_1))$
   $= I(\texttt{BNot})(eval(I, \zeta)(b_1\sigma))$
   $= eval(I, \zeta)(\texttt{BNot}(b_1)\sigma)$
   — $b = \texttt{BAnd}(b_1, b_2)$.
   Hypothèse d'induction sur b_1 : $eval(I, \zeta[X := n])(b_1) = eval(I, \zeta)(b_1\sigma)$.
   Hypothèse d'induction sur b_2 : $eval(I, \zeta[X := n])(b_2) = eval(I, \zeta)(b_2\sigma)$.
   $eval(I, \zeta[X := n])(\texttt{BAnd}(b_1, b_2))$
   $= I(\texttt{BAnd})(eval(I, \zeta[X := n])(b_1), eval(I, \zeta[X := n])(b_2))$
   $= I(\texttt{BAnd})(eval(I, \zeta)(b_1\sigma), eval(I, \zeta)(b_2\sigma))$
   $= eval(I, \zeta)(\texttt{BAnd}(b_1, b_2)\sigma)$

   □

Lemme 2. *Soit P un programme ne contenant pas d'affectation sur X et $\sigma = [X := \overline{n}]$. Soit ζ une valuation. Si $P\sigma : \zeta \rightsquigarrow \zeta'$ alors $P : \zeta[X := n] \rightsquigarrow \zeta'[X := n]$.*

Démonstration. Par induction sur la dérivation de $P\sigma : \zeta \rightsquigarrow \zeta'$, par cas sur la règle utilisée pour obtenir la conclusion :

— (Imp_{Skip}) : alors $\zeta = \zeta'$ et $P = \texttt{CSkip} = P\sigma$. On peut donc appliquer (Imp_{Skip}) :

$$\frac{}{P : \zeta[X := n] \rightsquigarrow \zeta[X := n]}\ (Imp_{Skip})$$

— (Imp_{Seq}) : alors $P\sigma = \texttt{CSeq}(P_1\sigma, P_2\sigma)$ et il existe ζ'' telle que $P_1\sigma : \zeta \rightsquigarrow \zeta''$ et $P_2\sigma : \zeta'' \rightsquigarrow \zeta'$.
Par hypothèse d'induction sur la dérivation de $P_1\sigma : \zeta \rightsquigarrow \zeta''$, on obtient une dérivation de $P_1 : \zeta[X := n] \rightsquigarrow \zeta''[X := n]$.
Par hypothèse d'induction sur la dérivation de $P_2\sigma : \zeta'' \rightsquigarrow \zeta'$, on obtient une dérivation de $P_2 : \zeta''[X := n] \rightsquigarrow \zeta'[X := n]$.
On peut donc appliquer (Imp_{Seq}) :

$$\frac{P_1 : \zeta[X := n] \rightsquigarrow \zeta''[X := n] \quad P_2 : \zeta''[X := n] \rightsquigarrow \zeta'[X := n]}{\texttt{CSeq}(P_1, P_2) : \zeta[X := n] \rightsquigarrow \zeta'[X := n]}\ (Imp_{Seq})$$

— (Imp_{Ass}) : alors $P\sigma = \texttt{CAss}(Y, e\sigma)$, avec $Y \neq X$. L'application de (Imp_{Ass}) a la forme suivante :

$$\frac{}{\texttt{CAss}(Y, e\sigma) : \zeta \rightsquigarrow \zeta[Y := eval(I, \zeta)(e\sigma)]}\ (Imp_{Ass})$$

On peut faire les remarques suivante :

— $P = \texttt{CAss}(Y, e)$

— $\zeta' = \zeta[Y := eval(I, \zeta)(e\sigma)]$

— $\zeta'[X := n] = \zeta[Y := eval(I, \zeta)(e\sigma)][X := n] = \zeta[X := n][Y := eval(I, \zeta)(e\sigma)]$

D'après la question 1, $eval(I, \zeta)(e\sigma) = eval(I, \zeta[X := n])(e)$. Donc $\zeta'[X := n] = \zeta[X := n][Y := eval(I, \zeta[X := n])(e)]$. Or, en appliquant ($Imp_{Ass}$), on peut obtenir :

$$\frac{}{\texttt{CAss}(Y, e) : \zeta[X := n] \rightsquigarrow \zeta[X := n][Y := eval(I, \zeta[X := n])(e)]}\ (Imp_{Ass})$$

— (Imp_{IfTrue}) : on a

— $P\sigma = \texttt{CIf}(b\sigma, P_1\sigma, P_2\sigma)$ et $P = \texttt{CIf}(b, P_1, P_2)$

— $eval(I, \zeta)(b\sigma) = true$, et donc par le lemme 1 $eval(I, \zeta[X := n])(b) = true$.

— On a une dérivation de $P_1\sigma : \zeta \rightsquigarrow \zeta'$.

Par induction sur la dérivation de $P_1\sigma : \zeta \rightsquigarrow \zeta'$, on obtient une dérivation de $P_1 : \zeta[X := n] \rightsquigarrow \zeta'[X := n]$. On peut donc appliquer (Imp_{IfTrue}) :

$$\frac{P_1 : \zeta[X := n] \rightsquigarrow \zeta'[X := n]}{\texttt{CIf}(b, P_1, P_2) : \zeta[X := n] \rightsquigarrow \zeta'[X := n]}\ (Imp_{IfTrue})\ eval(I, \zeta[X := n])(b) = true$$

— ($Imp_{IfFalse}$) : ce cas est similaire au précédent.

— ($Imp_{WhileFalse}$) : on a

— $P\sigma = \texttt{CWhile}(b\sigma, P_1\sigma)$ et $P = \texttt{CWhile}(b, P_1)$

— $eval(I, \zeta)(b\sigma) = false$, et donc par le lemme 1 $eval(I, \zeta[X := n])(b) = false$.

— $\zeta = \zeta'$

On peut donc appliquer $(Imp_{WhileFalse})$:

$$\frac{}{\texttt{CWhile}(b, P_1) : \zeta[X := n] \rightsquigarrow \zeta[X := n]}\ (Imp_{WhileFalse})\ eval(I, \zeta[X := n])(b) = false$$

— $(Imp_{WhileTrue})$: il existe ζ'' tel que :

— $P\sigma = \texttt{CWhile}(b\sigma, P_1\sigma)$ et $P = \texttt{CWhile}(b, P_1)$

— $eval(I, \zeta)(b\sigma) = true$, et donc par le lemme 1 $eval(I, \zeta[X := n])(b) = true$.

— on a une dérivation de $P_1\sigma : \zeta \rightsquigarrow \zeta''$.

— on a une dérivation de $\texttt{CWhile}(b\sigma, P_1\sigma) : \zeta'' \rightsquigarrow \zeta'$.

Par induction sur la dérivation de $P_1\sigma : \zeta \rightsquigarrow \zeta''$, on obtient une dérivation de $P_1 : \zeta[X := n] \rightsquigarrow \zeta''[X := n]$.
Par induction **sur la dérivation de** $\texttt{CWhile}(b\sigma, P_1\sigma) : \zeta'' \rightsquigarrow \zeta'$, on obtient une dérivation de $\texttt{CWhile}(b\sigma, P_1\sigma) : \zeta''[X := n] \rightsquigarrow \zeta'[X := n]$.
On peut donc appliquer $(Imp_{WhileTrue})$:

$$\frac{P_1 : \zeta[X := n] \rightsquigarrow \zeta''[X := n] \quad \texttt{CWhile}(b\sigma, P_1\sigma) : \zeta''[X := n] \rightsquigarrow \zeta'[X := n]}{\texttt{CWhile}(b, P_1) : \zeta[X := n] \rightsquigarrow \zeta'[X := n]}\ (Imp_{WhileTrue})$$

car $eval(I, \zeta[X := n])(b) = true$

Remarque : on peut voir ici que faire une induction sur P ne fonctionnerait pas, car la deuxième hypothèse d'induction s'appuie sur le même P. La dérivation, elle, est différente (les dérivation des hypothèses d'induction servent bien à fabriquer celle du cas).

□

Reste la preuve de la question :

Démonstration. Supposons qu'il existe une dérivation de $P\sigma : \zeta \rightsquigarrow \zeta'$. D'après le lemme 2, il existe une dérivation de $P : \zeta[X := n] \rightsquigarrow \zeta'[X := n]$. On peut alors construire la dérivation suivante :

$$\frac{\dfrac{}{\texttt{CAss}(X, \overline{n}) : \zeta \rightsquigarrow \zeta[X := n]}\ (Imp_{Ass}) \quad P : \zeta[X := n] \rightsquigarrow \zeta'[X := n]}{\texttt{CSeq}(\texttt{CAss}(X, \overline{n}), P) : \zeta \rightsquigarrow \zeta'[X := n]}\ (Imp_{Seq})$$

□