Université Bordeaux 1
Licence, module Compilation, 2017/2018

TD5-Sémantique

Sémantique de IMP

Exercice 5.1

On considère les deux définitions de sémantique pour le langage IMP :
- la sémantique opérationnelle à petits pas : $[\![c]\!]_p : \mathbb{Z}^V \to \mathbb{Z}^V$
- la sémantique opérationnelle à grands pas $[\![c]\!]_g : \mathbb{Z}^V \to \mathbb{Z}^V$.

1- On introduit la relation $\vdash_p$ définie par :

$$\rho/e \vdash_p \rho/v \Leftrightarrow [\![e]\!]\rho = v$$

$$\rho/c \vdash_p \rho' \Leftrightarrow (c,\rho) \to^* (\varepsilon,\rho')$$

Montrer que $\vdash_p$ vérifie les neuf clauses (fin de la page 3 du document "sémantique") qui définissent la sémantique à grands pas. Que peut-on en déduire sur $[\![c]\!]_p$ et $[\![c]\!]_g$?

2- Montrer par récurrence sur n que :

$$(c,\rho) \to^n (\varepsilon,\rho') \Rightarrow \rho/c \vdash \rho'.$$

Que peut-on en déduire sur $[\![c]\!]_p$ et $[\![c]\!]_g$?

3- Montrer que, pour toute commande c de IMP, $[\![c]\!]_p = [\![c]\!]_g$.

Exercice 5.2

Pour chacune des instructions c suivantes sur l'ensemble de variables $V = \{x\}$, quelle est la fonction $[\![c]\!] : \mathbb{Z}^V \to \mathbb{Z}^V$?

```
if 1 then x := 1 else Skip   while 1 do Skip   if 1 then x := 1 else while 1 do Skip
if 0 then x := 1 else while 1 do Skip   while 1 do Skip Se if 1 then x := 1 else x:=1

        x:=10 Se  while x do x := x+1   x:=10 Se while x do x := x-1
   while x do x := x+1   while x do x := x-1   if 1 then x := 1 else Skip
```

Exercice 5.3

Parmi les programmes c_i ($i \in [1,10]$) de l'exercice précédent, lesquels sont *équivalents* i.e. définissent la même fonction $[\![c_i]\!]$?

Sémantique de Léa

Exercice 5.4

On considère trois programmes qui comportent la déclaration de fonction puis la déclaration de variables globales :

```
function foo ( z: integer ) : type integer
    begin  while true do z := z+1; end
var x,y:integer;
```

1- Quelle sera la sémantique du programme, si le corps est contitué de :

```
begin
x:= 10; if false then x:= foo(0) else y:=x+1; write(y);
end
```
2- Reprendre la question 1 avec

```
begin
x:= 10; if false then y:=x+1 else x:= foo(0); write(y);
end
```
3-Reprendre la question 1 avec

```
begin
x:= 10; y:=x+1; if (foo(2)= 3) then write(x) else write(y);
end
```

Attributs

Exercice 5.5

Soient D, I deux ensembles. Soit $\mathcal{F}(D, I)$ l'ensemble des fonctions de D dans I. On définit une relation binaire $\sqsubseteq$ sur $\mathcal{F}(D, I)$ par

$$f \sqsubseteq g \Leftrightarrow (\mathrm{Dom}(f) \subseteq \mathrm{Dom}(g) \text{ et } \forall x \in \mathrm{Dom}(f), f(x) = g(x))$$

Vérifier que $\sqsubseteq$ est une relation d'ordre sur $\mathcal{F}(D, I)$. On considère l'ensemble ordonné $(\mathcal{F}(D, I), \sqsubseteq$).
1- Est-ce un treillis ?
2- Est-ce que toute suite croissante admet une borne supérieure ?
3- Montrer que toute application continue $F : \mathcal{F}(D, I) \mapsto \mathcal{F}(D, I)$ admet un plus petit point fixe.

Exercice 5.6

Soit $\mathcal{G} := \langle X, N, P, \mathcal{A}, (V_\alpha)_{\alpha \in \mathcal{A}}, R \rangle$ une grammaire d'attributs. Soit T un arbre de dérivation qui induit un graphe de dépendance (des attributs) sans-cycle.
1- Montrer que le système d'équations (4) [1] admet une *plus petite solution* $(\hat{\alpha})_{\alpha \in \mathcal{A}}$.
Cette plus petite solution est appelée *la* pp-décoration de T.
2- Donner un exemple de grammaire d'attributs et d'arbre de dérivation tel que le système d'équations (4) admet plusieurs solutions.
Admet-il une plus grande solution ? Plusieurs solutions maximales ?
3- Soit T un arbre de dérivation de $\langle X, N, P \rangle$, soit $S = T(\varepsilon)$ (i.e. S est le non-terminal qui étiquette la racine de T) et $\alpha_0 \in \mathcal{A}_0(S)$ (i.e. un attribut synthétisé de S).
Comment peut-on tester, par un calcul sur le graphe de dépendance des attributs de T, si dans la pp-décoration $(\hat{\alpha})_{\alpha \in \mathcal{A}}$, $\varepsilon \in \mathrm{Dom}(\hat{\alpha}_0)$? (i.e. si l'attribut α_0 est "bien-défini" à la racine ?)
*4- Connaissant la grammaire d'attributs $\mathcal{G}$, le non-terminal S et l'attribut α_0, peut-on tester que *tout* arbre de dérivation de racine S satisfait la condition de la question 3 ?
NB : C'est un problème difficile ; on ne demande que des idées (informelles) de départ.

Exercice 5.7

Soit $G\langle X, N, P, \sigma \rangle$ une grammaire en forme normale quadratique de Chomsky. On suppose

1. donné en annexe

que σ n'apparait dans aucun membre droit de règle. Un non-terminal $S \in N$ est dit binaire (resp. unaire) ssi il est membre gauche d'une règle de la forme $S \to TU$ où $T, U \in N$ (resp. $S \to s$ où $s \in X$). On suppose qu'aucun non-terminal n'est à la fois binaire et unaire et que σ est binaire.
Ecrire des règles de calcul d'attributs qui permettent de calculer, en chaque noeud, son numero dans l'ordre de parcours en profondeur, de gauche à droite.

Exercice 5.8

Considérons le schéma de traduction suivant pour les If-Then ["Le dragon", page 403].
Règle syntaxique : `S -> if(B)S`
Règle sémantique :
l'attribut `trad` de `structured_statement` est défini comme le code :

```
B.true     =      newlabelQ
B.false     =     Si.next     =     S.next
S.code    =    B.code   ||   label(B .true) ||  Si.code
```

"... We assume that newlabelQ creates a new label each time it is called, ..."

1- Ces règles sont-elles conformes a la définition [Knuth 68] d'une grammaire d'attributs ?
2- Pouvez-vous, en ajoutant un attribut supplémentaire, obtenir une grammaire d'attributs, au sens de [Knuth 68] ?

Exercice 5.9

Une instruction
`if (A and B) then I else J` où `A,B` sont des expressions et `I,J` des instructions, est traduite en code à trois adresses par :

```
debA  : trad(A)
        ...
afterA:JumpZ resA  debJ
debB  :trad(B)
        ...
afterB:JumpZ resB  debJ
debI  :trad(I)
        ...
afterI:Jump afterJ
debJ  :trad(J)
        ...
afterJ:
```

où `trad(A)`,`trad(B)`, `trad(I)`,`trad(J)` sont les traductions de A, B, I, J et `resA`,`resB` sont les adresses où sont stockés les résultats des évaluations de `A` (resp. `B`).
1- Cette traduction est-elle correcte si le langage de départ est `IMP`, dans lequel on a ajouté un opérateur `and` qui est interprété comme le produit des entiers ?
2- Cette traduction est-elle correcte si le langage de départ est `Léa` ?

Bisimulations

Exercice 5.10

Parmi les automates de la Figure 1, lesquels sont bisimilaires ?

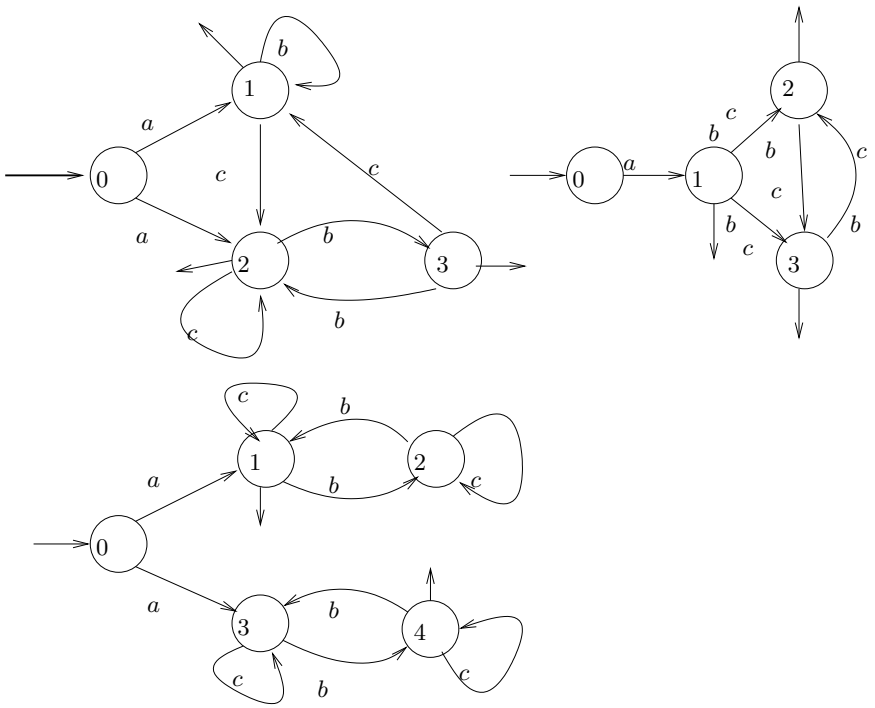

FIGURE 1 – exercice 5.10 : trois automates

Exercice 5.11

1- Montrer que, si deux états d'un automate sont bisimilaires, alors ils reconnaissent le même langage.

2- La réciproque est-elle vraie ?

3- Soit $\mathcal{A}$ un automate *déterministe*. Reprendre la question 2 avec cette hypothèse supplémentaire.

Exercice 5.12

1- Montrer que tout automate (non-déterministe) a une *plus grande* bisimulation.

2- Proposer, pour les automates finis, non-déterministes, un algorithme de calcul de cette plus grande bisimulation.

3- Reprendre la question 1 pour une structure de Kripke.

4- Reprendre la question 2 pour une structure de Kripke, finie, non-déterministe.

5- Calculer la plus grande bisimulation du premier automate de l'exercice 5.10.

Exercice 5.13

Soit V un ensemble fini. On considère l'ensemble d'actions

$$A := \{c \mid c \text{ programme IMP sur l'ensemble de variables } V\}$$

et la signature propositionnelle $\mathcal{P}$ formée des expressions de IMP n'utilisant que des variables de V. On définit une structures de Kripke $\mathcal{K}$ sur A et $\mathcal{P}$ par :

$$Q := \mathbb{Z}^V, \quad \delta = \{(\rho, c, \rho') \mid \rho, \rho' \in \mathbb{Z}^V, c \in A, [\![c]\!]\rho = \rho',$$

$$e^{\mathcal{K}}(\rho) = 0 \Leftrightarrow [\![e]\!]\rho = 0, \quad e^{\mathcal{K}}(\rho) = 1 \Leftrightarrow [\![e]\!]\rho \neq 0.$$

On définit une structure de Kripke $\mathcal{K}'$ sur A et $\mathcal{P}$ par : $\mathcal{E}$ est l'ensemble des expressions correctes en IMP (i.e. l'ensemble des mots engendrées par le non-terminal `E` de la grammaire

de IMP dans lequel les occurrences du symbole `I` sont remplacées par des suites de 0 et de 1). On note $\mu : \mathbb{N} \to \{0\} \cup \{1, \ldots, 9\}\{0, \ldots, 9\}^*$ l'écriture d'un nombre entier naturel en base 2. On étend μ à $\mathbb{Z}$ en posant, pour tout $n \in \mathbb{N} : \mu(-n) := 0 - \mu(n)$ (il s'agit de la concatenation du mot de longueur 2 ,"0 -", avec le mot $\mu(n)$).
$Q' = \mathcal{E}^V$. Les transitions de $\mathcal{K}'$ sont définies par l'ensemble de règles :

$$\text{Parenthèses}\ \frac{\rho/c \vdash \rho'}{\rho/(c) \vdash \rho'} \qquad \text{Séquence}\ \frac{\rho/c_1 \vdash \rho_1,\ \ \rho_1/c_2 \vdash \rho_2}{\rho/c_1\ \texttt{Se}\ c_2 \vdash \rho_2} \qquad \text{Skip}\ \frac{}{\rho/\ \texttt{Sk}\ \rho}$$

$$\text{Affectation de valeur}\ \frac{[\![e]\!]\rho = v}{\rho/x\ \texttt{Af}\ e\ \vdash\ \rho[x \mapsto \mu(v)]} \qquad \text{Affectation d'expression}\ \frac{}{\rho/x\ \texttt{Af}\ e\ \vdash\ \rho[x \mapsto e]}$$

$$\text{If true}\ \frac{[\![e]\!]\rho \neq 0,\ \ \rho/c_1\ \vdash\ \rho_1}{\rho/\texttt{If}\ e\ \texttt{Th}\ c_1\ \texttt{El}\ c_2\ \vdash\ \rho_1} \qquad \text{If false}\ \frac{[\![e]\!]\rho = 0,\ \ \rho/c_2\ \vdash\ \rho_2}{\rho/\texttt{If}\ e\ \texttt{Th}\ c_1\ \texttt{El}\ c_2\ \vdash\ \rho_2}$$

$$\text{While true}\ \frac{[\![e]\!]\rho \neq 0,\ \ \rho/\texttt{Wh}\ e\ \texttt{Do}\ c \vdash\ \rho_1}{\rho/\texttt{Wh}\ e\ \texttt{Do}\ c \vdash\ \rho_1} \qquad \text{While false}\ \frac{[\![e]\!]\rho = 0,}{\rho/\texttt{Wh}\ e\ \texttt{Do}\ c \vdash\ \rho}$$

$$e^{\mathcal{K}}(\rho) = 0 \Leftrightarrow [\![e]\!]\rho = 0,\ \ e^{\mathcal{K}}(\rho) = 1 \Leftrightarrow [\![e]\!]\rho \neq 0.$$

1- $\mathcal{K}'$ est-elle déterministe ?
2- Montrer que, pour tout $\rho \in \mathbb{Z}^V$, il existe une bisimulation R de $\mathcal{K}$ vers $\mathcal{K}'$ telle que $(\rho, \rho \circ \mu) \in R$ [2].
3- Soit $\mathcal{K}''$ obtenue à partir de $\mathcal{K}'$ en faisant un choix déterministe de transition à chaque action d'affectation (selon une stratégie quelconque, que l'on se fixe). Montrer que, pour tout $\rho \in \mathbb{Z}^V$, il existe une bisimulation R de $\mathcal{K}$ vers $\mathcal{K}''$ telle que $(\rho, \rho) \in R$.
4- Que peut-on en conclure sur ce que calculent $\mathcal{K}, \mathcal{K}''$?
5- On enrichit le langage IMP d'un opérateur unaire `tow` dont la sémantique est donnée par :

$$\text{si } n < 0,\ [\![\text{tow}]\!](n) = 0,\ \ [\![\text{tow}]\!](0) := 1, \quad \text{si } n \geq 0\ [\![\text{tow}]\!](n+1) = 2^{[\![\text{tow}]\!](n)}.$$

Un interpréteur qui simule une structure $\mathcal{K}''$ peut-il présenter un intérêt ? (du point de vue de la complexité du calcul).

2. où $\forall x \in V, \rho \circ \mu(x) = \mu(\rho(x))$

ANNEXE

Grammaire d'attributs.
Une grammaire d'attributs [Knuth, 68] est un tuple

$$\mathcal{G} := \langle X, N, P, \mathcal{A}, (V_\alpha)_{\alpha\in\mathcal{A}}, R\rangle$$

où
- $\langle X, N, P\rangle$ est une grammaire algébrique,
- $\mathcal{A}$ est un ensemble, appelé ensemble des *attributs* ;
- pour chaque $\alpha \in \mathcal{A}$, V_α est un ensemble, appelé l'ensemble des *valeurs* de l'attribut α
- l'ensemble R est constitué des *règles sémantiques*, décrites ci-dessous.

À chaque non-terminal $S \in N$ sont associés deux ensembles disjoints d'attributs $\mathcal{A}_0(S)$, $\mathcal{A}_1(S) \subseteq \mathcal{A}$; les membres de $\mathcal{A}_0(S)$ (resp. $\mathcal{A}_1(S)$) sont appelés attributs *synthétisés* (resp. *hérités*) du non-terminal S.
À chaque règle (syntaxique) $p \in P$

$$S_{p,0} \to S_{p,1}S_{p,2}\dots S_{p,n_p} \tag{1}$$

est associé un ensemble de règles (sémantiques) $R_{p,\alpha,i}$, pour $i \in [0, n_p], \alpha \in \mathcal{A}(S_{p,i})$; chacune de ces règles est de la forme

$$(R_{p,\alpha,i}) : \quad (\alpha, i) = f_{p,\alpha,i}((\alpha_1, i_1), \dots, (\alpha_t, i_t)) \tag{2}$$

avec $t \in \mathbb{N}, i_j \in [0, n_p], \alpha_j \in \mathcal{A}(S_{p,i_j})$, et d'applications

$$f_{p,\alpha,i} : V_{\alpha_1} \times V_{\alpha_2} \times \ldots \times V_{\alpha_t} \to V_\alpha \tag{3}$$

Les indices doivent respecter la convention que :
- si $i = 0$, alors $\alpha \in \mathcal{A}_0(S_{p,0})$ et $\forall j \in [1, t], i_j \in [1, n_p]$
- si $i \in [1, n_p]$, alors $\alpha \in \mathcal{A}_1(S_{p,i})$.

R est l'ensemble de ces règles $(R_{p,\alpha,i})$.
L'idée intuitive est que la valeur de l'attribut α de la variable $S_{p,i}$:
- sur un noeud père de la règle p (i.e. $i = 0$) et si α est synthétisé, est l'image par $f_{p,\alpha,0}$ des attributs de ses fils
- sur un noeud fils de la règle p (i.e. $i \geq 1$), et si α est hérité, est l'image par $f_{p,\alpha,i}$ des attributs de son père, de ses frères et de lui-même.

NB1 : Pour un attribut α, le fait d'être "synthétisé" ou "hérité" dépend du non-terminal $S \in V$ considéré, mais pas de la règle p où apparait S.
NB2 : Il est possible que $\mathcal{A}_0(S_1) \cap \mathcal{A}_1(S_2) \neq \emptyset$ i.e. que α soit synthétisé pour un non-terminal S_1 mais hérité pour un autre non-terminal S_2 ; on peut cependant normaliser l'ensemble des attributs de façon que tout attribut α soit :
- membre d'au moins un ensemble $\mathcal{A}(S)$
- synthétisé (pour *tous* les non-terminaux S tels que $\alpha \in \mathcal{A}(S)$)
- ou bien hérité (pour *tous* les non-terminaux S tels que $\alpha \in \mathcal{A}(S)$).

Une *décoration* d'un arbre de dérivation T par la grammaire $\mathcal{G}$ est une famille de fonctions

$$\hat{\alpha} : \mathrm{Dom}(T) \to V_\alpha$$

vérifiant, pour tout noeud $u \in \mathrm{Dom}(T) \subseteq (\mathbb{N} \setminus \{0\})^*$, tel que u possède n_p fils et

$$p = (T(u), T(u1)T(u2)\dots T(un_p))$$

on a

$$\hat{\alpha}(u \odot i) = f_{p,\alpha,i}(\hat{\alpha_1}(u \odot i_1), \dots, \hat{\alpha_t}(u \odot i_t)) \tag{4}$$

où $u \odot 0$ dénote le mot u et $u \odot i$ dénote le mot u suivi de la lettre i (si $i \neq 0$).
Structure de Kripke.
Une structure de Kripke, sur l'alphabet d'actions A et la signature propositionnelle $\mathcal{P}$ est un tuple

$$\mathcal{K} := \langle A, Q, \delta, (P^{\mathcal{K}})_{P \in \mathcal{P}} \rangle$$

tel que

- A est un ensemble, l'ensemble des actions
- Q est un ensemble, l'ensemble des états
- $\delta \subset Q \times A \times Q$ est l'ensemble des transitions
- pour tout $P \in \mathcal{P}$, $P^{\mathcal{K}} : Q \to \{0, 1\}$ est une application à valeurs booléennes.

Un automate $\mathcal{A}$ sur l'alphabet d'entrée A et l'ensemble d' états Q peut être vu comme une structure de Kripke en choisissant une structure propositionnelle à un seul prédicat F, interprété par

$$F^{\mathcal{A}}(q) := 1 \text{ ssi } q \text{ est final .}$$

Bisimulation.
Soit A un alphabet, $\mathcal{P}$ une signature propositionnelle et, pour chaque $i \in \{0, 1\}$, $\mathcal{K}_i := \langle A, Q, \delta, (P)^{\mathcal{K}_i}_{P \in \{\mathcal{P}\}} \rangle$ une structure de Kripke sur $A, \mathcal{P}$. Une relation binaire $R \subseteq Q_0 \times Q_1$ est une *bisimulation* de $\mathcal{K}_1$ vers $\mathcal{K}_2$ ssi :
(B1)$\forall (p, q) \in R, \forall P \in \mathcal{P}, P^{\mathcal{K}_0}(p) = P^{\mathcal{K}_1}(q)$
(B2)$\forall (p, q) \in R, \forall (p, a, p') \in \delta_0, \exists q' \in Q_1, (q, a, q') \in \delta_1$ et $(q, q') \in R$
(B3)$\forall (p, q) \in R, \forall (q, a, q') \in \delta_1, \exists p' \in Q_0, (p, a, p') \in \delta_0$ et $(p, p') \in R$