

**Exercice 1 (Interprétation des nombres binaires)**

On considère le langage des nombres en représentation binaire dont les mots (ou expressions) sont les théorèmes du système d'inférence suivant :

$$(R_1)_{\overline{0}} \quad (R_2)_{\overline{1}} \quad (R_3)_{\frac{n}{n\overline{0}}} \quad (R_4)_{\frac{n}{n\overline{1}}}$$

On notera  $\mathcal{NB}$  l'ensemble de ces théorèmes.

1. Définir une fonction d'interprétation  $\llbracket \_ \rrbracket : \mathcal{NB} \rightarrow \mathbb{N}$  des expressions de ce langage.
2. En utilisant cette fonction, calculer  $\llbracket 1011 \rrbracket$ .

**Exercice 2 (Induction)**

On considère le sous-ensemble  $D$  de  $\mathbb{N} \times \mathbb{N}$  défini inductivement par le système d'inférence :

$$(R_1)_{\frac{\quad}{(n, 0)}} \quad (R_2)_{\frac{(n, n')}{(n, n + n')}}}$$

1. Donner quelques éléments de  $D$ .
2. Montrer que si  $(n, n') \in D$ , alors pour  $k \in \mathbb{N}$ , on a  $n' = kn$ .
3. Montrer par récurrence sur  $k$  que si  $n' = kn$  alors  $(n, n') \in D$ .

**Exercice 3 (Substitutions)**

L'ensemble  $E_A$  des expressions arithmétiques peut être vu comme l'ensemble de termes  $T_{\mathbb{Z} \cup \{+, -, \times, /\}}[V]$ . Dans ce contexte, une substitution est une fonction de  $V$  dans  $T_{\mathbb{Z} \cup \{+, -, \times, /\}}[V]$ . Dans cet exercice, on considère uniquement les substitutions dont le domaine est réduit à un singleton. On écrira donc ces substitutions sous la forme  $[x/e]$  où  $e$  est l'expression arithmétique à substituer à la variable  $x$ . De plus, on étend la notion de valuation en considérant l'ensemble :

$$\mathcal{V}[\mathbb{Z} \cup \{\text{Err}\}] = \{V \rightarrow \mathbb{Z} \cup \{\text{Err}\}\}$$

et on définit une opération sur ces valuations qui consiste à modifier la valeur associée à une variable. Cette opération est définie comme suit :

$$\sigma[x \leftarrow v](y) = \begin{cases} v & \text{si } y = x \\ \sigma(y) & \text{sinon} \end{cases}$$

où  $v \in \mathbb{Z} \cup \{\text{Err}\}$ . Montrer que :

$$\forall e, e' \in E_A, \forall x \in V, \forall \sigma \in \mathcal{V}[\mathbb{Z} \cup \{\text{Err}\}] \quad \mathcal{A}[\llbracket e[x/e'] \rrbracket]_{\sigma} = \mathcal{A}[\llbracket e \rrbracket]_{\sigma[x \leftarrow \mathcal{A}[\llbracket e' \rrbracket]_{\sigma}]}$$

où  $e[x/e']$  dénote le résultat de l'application de la substitution  $[x/e']$  à  $e$ .

**Exercice 4** Montrer que :

$$\forall \sigma \in \mathcal{V}[\mathbb{Z}] \quad \forall e \in E_A \quad \forall v \in \mathbb{V} \quad \langle e, \sigma \rangle \rightsquigarrow v \Leftrightarrow \mathcal{A}[\llbracket e \rrbracket]_{\sigma} = v$$

### Exercice 5 (Variables locales)

On ajoute à présent une construction permettant la présence de variables locales dans les expressions arithmétiques. Par exemple, on veut pouvoir considérer les expressions de la forme  $\text{let}(x, 3 + z, x + y)$ . Dans cette expression, les valeurs de  $z$  et  $y$  sont obtenues à partir de la valuation utilisée pour évaluer l'expression, tandis que  $x$  est une variable locale (re)définie lors de l'évaluation. On dit qu'une telle variable est liée, tandis que  $z$  et  $y$  sont libres. On ajoute donc au système d'inférence définissant  $E_A$  la règle :

$$(\mathbb{A}_7) \frac{a_1 \quad a_2}{\text{let}(x, a_1, a_2)}$$

permettant de considérer des expressions de la forme  $\text{let}(x, a_1, a_2)$ , où  $x \in V$ .

1. Donner un schéma d'interprétation des expressions de la forme  $\text{let}(x, a_1, a_2)$ .
2. Spécifier l'évaluation des expressions de la forme  $\text{let}(x, a_1, a_2)$  à l'aide de règles d'inférence.
3. En utilisant ces règles, montrer que  $\langle \text{let}(x, 3 + z, x + y), \sigma \rangle \rightsquigarrow 12$  où  $\sigma$  est une valuation vérifiant  $\sigma(x) = 2$ ,  $\sigma(y) = 5$  et  $\sigma(z) = 4$ .
4. L'ensemble  $\mathcal{F}(a) \subseteq V$  des variables libres d'une expression arithmétique  $a \in E_A$  correspond à l'ensemble des variables dont la valeur sera obtenue à partir de la valuation  $\sigma$  utilisée lors de l'évaluation. Par exemple, lors de l'évaluation de  $\text{let}(x, 3 + z, x + y)$ , on utilisera  $\sigma(z)$  et  $\sigma(y)$  mais pas  $\sigma(x)$ . Donner une définition formelle de l'ensemble  $\mathcal{F}(a) \subseteq V$  des variables libres apparaissant dans une expression arithmétique  $a \in E_A$ .
5. Existe-t-il un entier  $n$  tel qu'il existe un arbre de preuve de  $\langle \text{let}(y, y + 4, 3 + y), \sigma \rangle \rightsquigarrow n$  où  $\sigma$  est une valuation telle que  $\sigma(y) = 2$ ? Si oui, construire cet arbre et donner la valeur de  $n$ . Que pouvez-vous dire de la variable  $y$ ? Est-ce une variable libre? Est-ce une variable liée?
6. Prouver que le résultat de l'évaluation d'une expression arithmétique  $a$ , étant donnée une valuation  $\sigma$ , ne dépend que de la valeur associée par  $\sigma$  aux variables libres présentes dans  $\mathcal{F}(a)$ .

$$\forall a \in E_A, \forall \sigma_1, \sigma_2 \in \mathcal{V}[\mathbb{Z}], \quad \sigma_1 =_{\mathcal{F}(a)} \sigma_2 \Rightarrow (\langle a, \sigma_1 \rangle \rightsquigarrow n \Leftrightarrow \langle a, \sigma_2 \rangle \rightsquigarrow n)$$

7. Les résultats affirmant l'existence d'un résultat, et le déterminisme pour l'évaluation d'une expression arithmétique sont-ils toujours vérifiés? Si oui, compléter les preuves.

### Exercice 6 (Sémantique d'un petit langage d'expressions)

On considère dans cet exercice un langage d'expressions construites à partir :

- de symboles de variables  $x, y, \dots$  pris dans un certain ensemble  $V$
- d'une constante littérale  $Id$
- d'un opérateur binaire de construction de paires (on notera les paires sous la forme  $\langle e_1, e_2 \rangle$ )
- de deux opérateurs unaires correspondant aux projections sur une paire, notés  $fst$  et  $snd$
- d'un opérateur binaire  $\oplus$

La grammaire des expressions peut donc s'exprimer par :

$$e ::= x \mid Id \mid \langle e, e \rangle \mid fst(e) \mid snd(e) \mid e \oplus e$$

1. Définir l'ensemble des expressions à l'aide d'un système d'inférence. De quel ensemble de termes s'agit-il ?
2. Quel est le schéma d'induction associé à cette définition inductive ?
3. Peut-on construire un arbre de dérivation pour  $fst(\langle x, Id \oplus y \rangle) \oplus z$  et pour  $snd(Id)$  ?
4. On dira qu'une expression est "mal formée" si l'arité des opérateurs n'est pas respectée ou si les opérateurs  $fst$  et  $snd$  ne sont pas appliqués à des paires. Définir un système d'inférence qui élimine les expressions "mal-formées". Les jugements manipulés par ce système sont notés  $e \text{ is } WT$  (et expriment que l'expression  $e$  est "bien formée"). Démontrer que, quelle que soit la stratégie d'application des règles de ce système, l'algorithme de vérification de "bonne formation" termine.
5. On souhaite maintenant simplifier les expressions bien formées. Pour cela on souhaite remplacer toute sous-expression de la forme  $fst(\langle e_1, e_2 \rangle)$  par  $e_1$  et toute sous-expression de la forme  $snd(\langle e_1, e_2 \rangle)$  par  $e_2$ . Exprimer cette transformation à l'aide d'un système d'inférence décrivant le processus de simplification à grands pas et démontrer que ce processus termine. Utiliser ce système pour simplifier l'expression  $fst(\langle Id \oplus snd(\langle x, Id \rangle), snd(\langle Id, Id \rangle) \rangle)$ .
6. Démontrer par induction structurelle que pour toute expression  $e$  "bien formée", il existe une unique expression  $e'$  telle que  $e'$  soit le résultat de la simplification de  $e$ .
7. Décrire précisément les expressions ne pouvant plus être simplifiées que nous appellerons *formes normales* puis proposer un système d'inférence définissant l'ensemble des formes normales. De quel ensemble de termes s'agit-il ?
8. Démontrer par induction structurelle que le processus de simplification défini à la question 5. transforme toute expression "bien formée" en une expression en forme normale. En déduire que toute expression "bien formée" admet une forme normale unique.
9. Les preuves demandées dans les questions 6. et 8. auraient aussi pu être obtenues par induction bien-fondée. Comment ? Quel aurait été l'avantage de procéder ainsi ?
10. On souhaite à présent interpréter les expressions en forme normale. Pour cela, on se donne l'ensemble de valeurs  $X = \{A, B, C, I\}$  pour domaine d'interprétation. Cet ensemble est muni de deux opérations  $+$  et  $*$  définies par les tables suivantes :

$+$	$A$	$B$	$I$	$*$	$A$	$B$	$I$
$A$	$B$	$B$	$A$	$A$	$I$	$A$	$B$
$B$	$A$	$A$	$B$	$B$	$I$	$A$	$A$
$I$	$A$	$B$	$I$	$I$	$A$	$B$	$I$

La constante  $Id$  sera interprétée par  $I$ , et on interprétera l'opérateur  $\oplus$  par  $+$  et le constructeur de paire par  $*$ . Définir un schéma d'interprétation des expressions en forme normale dans ce modèle.

11. Définir à l'aide d'un système d'inférence un évaluateur à grands pas d'expressions en formes normales. On notera  $\rho(x)$  la valeur associée à la variable  $x$  dans un environnement  $\rho$  et on notera  $\rho \vdash e \rightsquigarrow v$  le jugement exprimant que l'expression  $e$  dans l'environnement  $\rho$  s'évalue en la valeur  $v \in X$ .

12. Démontrer que pour toute expression en forme normale il existe une et une seule valeur  $v \in X$  telle que  $\rho \vdash e \rightsquigarrow v$ .

### Exercice 7 (Expressions booléennes)

1. Enoncer le schéma d'induction engendré par le système d'inférence définissant  $E_B$ .
2. Prouver les résultats d'existence et de déterminisme pour l'évaluation des expressions booléennes :

$$\forall b \in E_B \forall \sigma \in \mathcal{V}[\mathbb{Z}]$$

$$(i) \quad \exists v \in \mathbb{B} \cup \{\text{Err}\} \quad \langle b, \sigma \rangle \rightsquigarrow v$$

$$(ii) \quad \forall v_1, v_2 \in \mathbb{B} \cup \{\text{Err}\} \quad (\langle b, \sigma \rangle \rightsquigarrow v_1 \text{ et } \langle b, \sigma \rangle \rightsquigarrow v_2) \Rightarrow v_1 = v_2$$

### Exercice 8 (Equivalence de programmes)

Montrer que `while b do c`  $\equiv$  `if b then c; while b do c else skip`.

**Exercice 9** Dans cet exercice, on considère le langage impératif vu en cours dont la sémantique opérationnelle à grands pas contient les trois règles :

$$(C_2) \frac{\langle a, \sigma \rangle \rightsquigarrow n}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \leftarrow n]} \quad n \in \mathbb{Z}$$

$$(C_6) \frac{\langle b, \sigma \rangle \rightsquigarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}$$

$$(C_7) \frac{\langle b, \sigma \rangle \rightsquigarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma_1 \quad \langle \text{while } b \text{ do } c, \sigma_1 \rangle \rightarrow \sigma_2}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma_2}$$

Montrer que pour toute expression booléenne  $b$ , toute variable  $x$  et toute valuation  $\sigma$ , si  $\sigma(x)$  est pair et s'il existe un arbre d'inférence du jugement  $\langle \text{while } b \text{ do } x := x + 2, \sigma \rangle \rightarrow \sigma'$ , alors  $\sigma'(x)$  est pair. Dans le cas d'une démonstration par induction, indiquer explicitement sur quel objet l'induction est faite.

### Exercice 10 (Boucle repeat)

On souhaite étendre le langage des instructions afin de permettre l'écriture d'instructions de la forme : `repeat c until b` où  $c \in E_C$  et  $b \in E_B$ . La sémantique informelle de cette instruction est : "exécuter  $c$  jusqu'à ce que  $b$  prenne la valeur `true`".

1. Donner les règles d'inférence permettant de spécifier la sémantique de cette construction.
2. On définit l'instruction `foo` par : `repeat z := z * x; x := x - 1 until x = 0`. Si  $\sigma$  est un état vérifiant  $\sigma(z) = 3$  et  $\sigma(x) = 2$ , quel est l'état  $\sigma'$  tel que  $\langle \text{foo}, \sigma \rangle \rightarrow \sigma'$ ? Construire l'arbre de preuve.
3. Montrer que `repeat c until b`  $\equiv$  `c; if b then skip else repeat c until b`.
4. Montrer que :

$$\begin{aligned} \text{while } b \text{ do } c &\equiv \text{while not not } b \text{ do } c \\ \text{repeat } c \text{ until } b &\equiv \text{repeat } c \text{ until not not } b \end{aligned}$$

5. En déduire que :

$$\begin{aligned}\text{while } b \text{ do } c &\equiv \text{if } b \text{ then repeat } c \text{ until not } b \text{ else skip} \\ \text{repeat } c \text{ until } b &\equiv c; \text{while not } b \text{ do } c\end{aligned}$$

*Indication.* En notant :

$$\begin{aligned}c_1 : \text{ while } b \text{ do } c & \quad c_2 : \text{ if } b \text{ then repeat } c \text{ until not } b \text{ else skip} \\ c_3 : \text{ repeat } c \text{ until } b & \quad c_4 : c; \text{while not } b \text{ do } c\end{aligned}$$

il suffira de démontrer :

- (1)  $\forall \sigma, \sigma' \in \Sigma \quad \langle c_1, \sigma \rangle \rightarrow \sigma' \Rightarrow \langle c_2, \sigma \rangle \rightarrow \sigma'$
- (2)  $\forall \sigma, \sigma' \in \Sigma \quad \langle c_2, \sigma \rangle \rightarrow \sigma' \Rightarrow \langle c_1, \sigma \rangle \rightarrow \sigma'$
- (3)  $\forall \sigma, \sigma' \in \Sigma \quad \langle c_3, \sigma \rangle \rightarrow \sigma' \Rightarrow \langle c_4, \sigma \rangle \rightarrow \sigma'$
- (4)  $\forall \sigma, \sigma' \in \Sigma \quad \langle c_4, \sigma \rangle \rightarrow \sigma' \Rightarrow \langle c_3, \sigma \rangle \rightarrow \sigma'$

Les preuves de (1) et (3) vont s'obtenir par induction structurelle sur un arbre d'inférence, les implications (2) et (4) se prouveront très facilement en utilisant (1) et (3).