

UPMC/master/info/4I503 APS

EXAMEN RÉPARTI 2

Mai 2016

Les documents autorisés sont vos notes de cours manuscrites et les notes fournies par votre enseignant de cours (<http://www.pps.univ-paris-diderot.fr/~eleph/Enseignement/2015-16/APS>). Et ce, à l'exclusion de tout autre document: feuilles de TD/TME, notes manuscrites de TD/TME.

EXERCICE I

Typage et sémantique opérationnelle 1

À côté de la boucle *while-do* certains langages fournissent une boucle *repeat-until*. C'est comme la boucle *while-do* une structure d'itération qui exécute son corps jusqu'à ce qu'une condition booléenne soit vérifiée.

On ajoute à la syntaxe le mot clé REPEAT et l'instruction

```
Stat ::= ...
      | 'REPEAT' Prog Expr
```

Exemples:

- le programme [VAR x int; SET x 0; REPEAT [SET x (add x 1)] (eq x 1)] termine avec 1 pour valeur de x.
- le programme [VAR x int; SET x 1; REPEAT [SET x (add x 1)] (eq x 1)] ne termine pas.

QUESTION (I.1) Donnez la règle de typage et les règles de sémantiques opérationnelles de cette instruction.

EXERCICE II

Typage et sémantique opérationnelle 2

On a défini la structure de liste:

```
Syntaxe: mots clef 'list' 'nil' 'cons' 'car' 'cdr'
Type ::= .. | '(' 'list' Type ')'
Expr ::= .. | 'nil' | '(' 'cons' Expr Expr ')'
        | '(' 'car' Expr ')' | '(' 'cdr' Expr ')'
```

Typage: pour tout type t,

```
G |- nil:(list t)
si G |- e1:t et G |- e2:(list t) alors G |- (cons e1 e2):(list t)
si G |- e:(list t) alors G |- (car e):t
si G |- e:(list t) alors G |- (cdr e):(list t)
```

Sémantique: une constante d'adresse non allouable: a0

```
r,m |- nil ~> (a0,m)
Si r,m |- e1 ~> (v1,m') et r,m' |- e2 ~> (v2,m'')
alors r,m'' |- (cons e1 e2) ~> (a, m''[a=v1][a+1=v2])
Si r,m |- e ~> (a,m') avec a <> a0 alors r,m' |- (car e) ~> (m(a), m')
Si r,m |- e ~> (a,m') avec a <> a0 alors r,m' |- (cdr e) ~> (m(a+1), m')
```

On veut réaliser une structure de contrôle *filtrage* des cas de construction des listes (comme en ML). On adopte la syntaxe suivante:

```
Stat ::= ...
      | 'CASE' Expr '[' Prog '[' Ident ',' Ident ']' Prog
```

Intuitivement, dans `CASE e [] blk1 [x , xs] blk2`, l'expression *e* doit avoir pour valeur une liste; *blk₁* est le bloc exécuté si la liste est vide et *blk₂*, le bloc exécuté si la liste est non vide: dans ce cas, la valeur du premier élément de la liste est liée à la variable *x* et la valeur de sa suite, à la variable *xs*.

QUESTION (II.1) Donnez la règle de typage et les règles de sémantique opérationnelle de cette instruction.

EXERCICE III Sémantique dénotationnelle (*APS1*)

Soit le programme

```
[
  CONST f int->int [n:int](add n 10);
  VAR x int;
  SET x 32;
  WHILE (lt x 42)
    [
      SET x (f x)
    ]
]
```

QUESTION (III.1) En vous appuyant sur les équations sémantiques, donnez les étapes d'évaluation de ce programme.