UPMC/master/info/4I503 APS

Examen de 2nde session

Juin 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 *until-do*. C'est, comme la boucle *while-do*, une structure d'itération qui exécute son corps tant qu'une condition n'est pas vérifiée.

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

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

Exemples:

- le programme `[ VAR x int; SET x 0; UNTIL (eq x 1) [ SET x (add x 1) ] ]` termine avec 1 pour valeur de `x`;
- le programme `[ VAR x int; SET x 1; UNTIL (eq x 1) [ SET x (add x 1) ] ]` termine sans changer la valeur de `x`;
- le programme `[ VAR x int; SET x 2; UNTIL (eq x 1) [ SET x (add x 1) ] ]` ne termine pas.

Question (I.1) Donnez la règle de typage et les règles de sémantique opérationnelle 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 ::= ...
     |   'MATCH' Expr 'nil' Prog '(cons' Ident  Ident ')' Prog
```

Intuitivement, dans `MATCH` e `nil` blk_1 `(cons` x xs`)` blk_2, l'expression e doit avoir pour valeur une liste; blk_1 est le bloc exécuté si la liste est vide et blk_2, 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 20);
  VAR x int;
  SET x 22;
  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.