Master INIS 2ème année
Analyse statique SOM4IF15

Devoir de programmation.

Réalisation d'un analyseur pour le langage WHILE

Spécification lexicale :

- *Valeurs et types :* on considèrera uniquement les variables et les constantes de type entier, et les constantes booléennes true et false. Toutes les informations sont de type entier et toutes les bexpressions sont de type booléen.
- La constante représente n'importe quel entier qui peut être codé sur 32 bits.
- L'identificateur appartient au langage exprimé par l'expression régulière $(a..z)\,((a..z) \mid \_ \mid (0..9))^*$.

Spécification syntaxique :

Les mots clé sont **en gras**, et ce qui est entre crochets est [optionnel].

Les notations A^* (resp. A^+) dénotent une liste, éventuellement vide, (resp. non vide) de A.

program	::=	**program** [identifier] declaration* **begin** lDeclVariables statements **end**
declaration	::=	**proc** identifier (lDeclIdent [,**res** type identifier]) **begin** statements **end**
lDeclIdent	::=	type identifier [, type identifier]*
lDeclVariables	::=	declVariables lDeclVariables*
declVariables	::=	type lIdentifier **;**
lIdentifier	::=	identifier [, identifier]*
type	::=	**int** | **boolean**
block	::=	statement
	|	**(** statements **)**
statements	::=	statement [**;** statements]*
statement	::=	**skip**
	|	identifier **:=** aexpression
	|	**if** bexpression **then** block [**else** block]
	|	**while** bexpression **do** block
	|	**call** identifier **(** lAexpression **)**
lAexpression	::=	aexpression [, aexpression]*
aexpression	::=	identifier | constant
	|	aexpression op_a aexpression
	|	**-** aexpression
	|	**(** aexpression **)**
op_a	::=	$+ \mid - \mid * \mid /$
bexpression	::=	**true** | **false**
	|	aexpression op_r aexpression
	|	**not** bexpression
	|	**(** bexpression **)**
op_r	::=	$< \mid <= \mid > \mid >= \mid = \mid <>$

Conventions : Dans le langage WHILE, les variables sont toutes globales. Toute variable utilisée est automatiquement visible dans tout le programme excepté les procédures. Une procédure a des paramètres, qui sont aussi ses variables locales. Une variable locale masque la variable globale de même nom.

Les procédures ne peuvent pas être imbriquées. Elles doivent être toutes définies l'une après l'autre avant le programme principal. Elles peuvent avoir une seule valeur de retour.
Les procédures peuvent être mutuellement récursives. Toute procédure est visible depuis n'importe quelle autre.

Question 1 : en utilisant les outils jflex et Cup, ou ANTLR, on vous demande de programmer une analyse lexicale et une analyse syntaxique du langage WHILE.

Préciser les types de l'arbre abstrait engendré par votre analyse.

Question 2 : écrivez un "module" qui permet d'imprimer les programmes que vous analysez en utilisant l'arbre abstrait.

Vous pourrez tester votre analyseur sur n'importe quel exemple simple comme ceux donnés en annexe ou ceux traités en cours et en TD.

Question 3 : Implantez le cadre monotone qui constitue l'algorithme générique pour les quatre méthodes d'analyse statique étudiées en cours et en TD. À l'issue de cette partie, il devra être possible de choisir une méthode pour analyser un programme.

Il est important qu'on puisse visualiser tous les éléments calculés lors de l'analyse.

Annexe :

Exemple 1

```
program p1
begin
int x;
x := 100;
while ( 0 < x ) do (
  x := x -1
)
end
```

Exemple 2

```
program p2
begin
int x, y, z;
y := x; z := 1;
while (y > 1) do (
  z := z*y; y := y-1
);
y := 0
end
```

Exemple 3

```
program
proc fib(int z, int u, res int v)
  begin
    if z < 3 then v := u+1
    else (call fib(z-1, u, v); call fib(z-2, v, v))
  end
begin
int x, y;
call fib(x, 0, y)
end
```

Exemple 4

```
program p4
begin
int x, y;
int[10] t;
x := 0; y := 0;
while (x < 10) do (
  y := y + t[x]; x := x+1
)
end
```