Examen de Compilation

Licence Sciences Pour Ingénieur

13 Mai 2014

Le sujet est composé d'exercices indépendants. Aucun document n'est autorisé. Durée de l'épreuve : 0x78 minutes. La présentation de la copie entre en compte dans la note finale.

1 Expression régulière

L'extrait de la page `regex` du manuel (`man 7 regex`) signale la possibilité de décrire des motifs avec des "références arrières".

1. Décrire le langage des mots correspondants au motif $(a+)b\backslash 1c$.
2. Est-il légitime de parler d'expressions régulières ?

```
REGEX(7)    Manuel du programmeur Linux
NOM         regex − Expressions rationnelles POSIX.2
DESCRIPTION
  la référence arrière: «\» suivi d'un chiffre décimal non−nul n corres−
−pond à la même séquence de caractères que ceux mis en correspondance avec
la n−ième sous−expression entre parenthèses. (les sous−expressions sont
numérotées par leurs parenthèses ouvrantes, de gauche à droite),
ainsi «([bc])\1» correspond à « bb » ou « cc » mais pas à « bc ».
```

2 Théorie des langages

Dans le cours, nous avons étudié la stabilité par intersection de deux classes de langages. Rappeler en quelques mots pourquoi :

1. l'intersection de deux langages rationels est rationel ;
2. en général, l'intersection de deux langages algébriques n'est pas algébrique.

3 Automate

On considère $A = \{a, b\}$.

1. Quel est le langage reconnu par U_3 ?
2. Déterminiser l'automate U_3.
3. Construire un automate non déterministe à $n + 1$ états qui reconnait A^*aA^n.
4. Montrer que les résiduels des mots de moins $n - 1$ lettres sont distincts.
5. Quelle conclusion en tirer sur l'algorithme de déterminisation ?

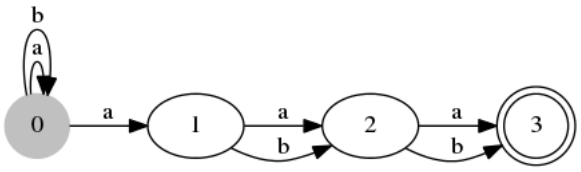

FIG. 1 – U_3 un automate non déterministe à 4 états.

TAB. 1 – opérations de la machine

STO r	stocke la valeur de ACC dans r.
VAL v	charge la valeur v dans l'accumulateur.
MEM m	charge le contenu de la mémoire m.
ADD r	ACC := ACC + registre[r].
MUL r	ACC := ACC + registre[r].

4 flex-bison

On considère une machine fictive possèdant 26 mémoires (A,B,C,...) et 16 registres (0,1,...), un accumulateur ACC et des opérations : STO, MEM, VAL, ADD, MUL.

Un fragment d'analyseur syntaxique `mini.y` est fourni. Il s'agit de le compléter pour obtenir l'ébauche d'une commande `mini.exe`.

1. Préciser la nature de la commande `mini.exe` : compilateur, interpréte ?
2. Ecrire un analyseur lexical adéquat en `flex`.
3. Quelle compilation permet d'obtenir `mini.exe` ?
4. Quel est le résultat de

```
echo -n 'A+B+C+7;' | ./mini.exe
```

5. Comment modifier l'analyseur syntaxique pour gérer la multiplication ?
6. Compléter l'analyseur syntaxique pour gérer une affectation.

```
%{
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
int use[16] = { 0 }, mem[26], tmp;
int yylex( void ), yyerror( char *x );
int reg( void )
{ int i = 0;
  while ( i < 16 && use[i] ) i++;
  if ( i == 16 ) {
  printf("\ndepassement de capacite!");
  exit(1);
}
use[i] = 1;
return i;
}
%}
```

```
18 %token  MEM NB SEP
19 %left   PLUS
20 %%
21 PROG  :  PROG INSTR SEP
22       |   /* epsilon */
23       ;
24 INSTR : EXP
25       ;
26 EXP :  EXP PLUS {
27             $1 = reg( );
28             printf("\nSTO %d;", $1);
29          } EXP {
30          printf("\nADD %d;", $1 );
31          use[ $1 ] = 0;
32          }
33     | MEM {
34          printf("\nMEM %c;", $1 );
35        }
36     | NB {
37          printf("\nVAL %d;", $1 );
38        }
39     ;
40
41 %%
42 int main(void) { yyparse(); return 0; }
```