Théorie des Langages et Compilation

L3 Sciences Pour Ingénieur

25 juin 2014

Le sujet est composé de quatre 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

Le programme `regex.c` ci-dessous est compilé en une commande `regex.exe`.

```
1  #include <regex.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4  int main(int argc, char **argv)
5  { regex_t regex;
6    if ( argc < 2) return 1;
7    argv++;
8    if (regcomp(&regex, *argv, REG_EXTENDED) != 0)
9      return 2;
10   while ( * ++argv != NULL )
11      if ( 0 == regexec(&regex, *argv, 0, NULL, 0) )
12         puts(*argv);
13   regfree(&regex);
14   return 0;
15 }
```

1. Commenter avec soin les lignes 8,11, 13 du code de `regex.c`.
2. Quel est le résultat de `./regex.exe '1{2}1' 1 11 21 1211 111221 312211` ?

2 Théorie des langages

$A := \{a, b\}$. On considère le langage X des mots de A^* qui contiennent autant de a que de b. On rappelle que le résiduel de $u \in A^*$ est $u^{-1}X = \{y \in A^* \mid uy \in X\}$.

1. Soit $x \in X$. Quel est le résiduel de x ?
2. Soit n un entier. Quel est le résiduel de a^n ?
3. Montrer que le langage n'est pas régulier.
4. Montrer que si x est non vide dans X alors $x = aubv$ ou $x = buav$ avec u et v dans X.
5. Montrer le langage est algébrique.

3 Automate

Conformément à l'usage, un mot binaire x de $\{0,1\}^+$ représente un nombre entier en base 2. Par exemple, 0001 représente 1, 101 représente 5 etc... Par convention, le mot vide représente 0.

1. Donner un automate pour reconnaître les mots binaires représentant des multiples de 5.
2. L'automate obtenu est-il déterministe ? minimal ?
3. D'aprés l'automate, quels mots de 1^* représentent des multiples de 5 ?
4. Interpréter le résultat à l'aide de sommes géométriques de raison 2 et du petit théorème de Fermat !

4 flex-bison

On considère le langage des expressions rationnelles sur l'alphabet $\{a, b\}$. On représente les opérations union, produit et étoile par les symboles usuels de la théorie des langages : (,), +, ., $*$. Par exemble, le langage contient les mots : a, $a+b$, $a+b.a^*$, $(a+b)^*$... Mais pas le mot ab.

L'objectif de l'exercice est de modifier la source `bison` ci-dessous pour réaliser un analyseur syntaxique qui imprime le numéro de la première ligne de l'entrée standard qui ne correspond pas à une expressions rationnelles. L'analyseur lexical sera écrit en `flex`.

1. Le langage est-il régulier ?
2. Ecrire une grammaire pour reconnaître ce langage.
3. Détailler les modifications.
4. Donner l'analayseur lexical.

```
1  %{
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <ctype.h>
5  int yylex( void ), yyerror( char *x );
6  int lino = 1;
7  %}
8  %token   - - -
9  %left    - - -
10 %left    - - -
11 % - - -
12 %%
13 LST :  LST EXP NL  - - -
14     |   /* epsilon */
15     ;
16 EXP :   - - -
17
18 %%
19 int main(void) {
20   yyparse(); return 0;
21 }
22
23 int yyerror( char*x) {
24    printf("%s:%d\n", x , lino); return 0;
25 }
```