I53- Compilation et théorie des langages
-TP 4 (partie 2)-

Licence 3 - 2020/2021

Compilation d'expressions régulières en automate

Le but de cette partie est d'écrire un programme similaire à `grep` capable de lire une expression régulière dans l'entrée standard et une chaîne de caractères quelconque et d'afficher si oui ou non la chaîne appartient au langage dénoté par l'expression régulière (en construisant un automate acceptant le langage correspondant). On concidère ici les expression régulières définies par la grammaire suivante:

Exreg $\rightarrow$ *Exreg* + *Exreg* (union)
Exreg $\rightarrow$ *Exreg* . *Exreg* (concaténation)
Exreg $\rightarrow$ *Exreg* $*$ (fermeture de Kleene)
Exreg $\rightarrow$ `Char` —*(Exreg)*

Ici le terminal `Char` représente n'importe quel caractère ascii alphanumérique. En particulier on impose ici à tous les automates de travailler sur le même alphabet à savoir `&abc...xyzABC...XYZ01...89`

Exemple d'utilisation:

```
$ ./mygrep
usage: ./mygrep <exreg> <chaine>
$ ./mygrep '(a+b).(a+c).b*' 'acbbbbb'
acceptee
$ ./mygrep '(a+b).(a+c).b*' 'bbbbbb'
rejetee
```

1. Écrire une fonction `void afn_char(afn *C, char c)` qui contruit un AFN acceptant le langage constitué du seul symbole `c`.

2. Écrire les trois fonctions suivantes

```
void afn_union(afn *C, afn A, afn B);
void afn_concat(afn *C, afn A, afn B);
void afn_kleene(afn *C, afn A);
```

   qui construisent respectivement les automates reconnaissant l'union, la concaténation et la fermeture de Kleene des langages acceptés par les automates A et B.

3. Construire une grammaire décrivant la structure des expressions régulières sur 3 caractères (ex: `a.(b+c)*.a.b*`.

4. Dans trois nouveaux fichiers (`compregex.h`, `compregex.c` et `mygrep.c` qui produiront un programme `mygrep`), écrire un traducteur dirigé par la syntaxe (sur le modèle du TP 2) qui construit un AFD reconnaissant le langage décrit par une expression régulière lue dans l'entrée standard.

5. (facultatif) On pourra améliorer le programme en rajoutant les opérateurs suivants:

   - $Exreg\{n\}$ qui dénote la répétition exactement n fois d'une expression régulière
   - $[c_1c_2...c_n]$ qui réprésente exactement un des caractères c_i.