I53- Compilation et théorie des langages
-TP 4 (partie 1)-

Licence 3 - 2019/2020

Simulation d'automates finis

Dans ce TP on se propose d'implanter un simulateur d'automates finis. On considérera ici des automates reconnaissant des mots à valeurs dans un alphabet quelconque où & représentera les ϵ-transitions. On limite de plus le nombres d'états des AFN à 64 pour simplifier leur représentation. Pour cela on utilisera les structures suivantes:

```
typedef unsigned int uint;
typedef unsigned long long int ullong;

typedef struct {
  uint nbetat, nbsymb, nbfinal;
  uint init;
  char * alphabet;
  uchar tsymb[SYMB_NB_MAX];
  ullong *finals;
  uint **delta;
} afd;

typedef struct{
  uint nbstate, nbsymb;
  ullong init, finals;
  char * alphabet;
  uchar tsymb[SYMB_NB_MAX];
  ullong **delta;
} afn;
```

Les états d'un AF seront simplement représenté par des entiers de 0 à $n-1$ (où n est le nombre d'états). Le champ `tsymb` représente un dictionnaire entre les valeurs ascci

Le nombre d'état d'un AFN est limité à 64. Pour représenter l'ensemble des états que peux prendre un AFN on utilisera un entier de 64 bits dont chaque bit représente un état. Par exemple, si les états initiaux de l'AFN sont 1 et 3, l'attribut `init` vaudra 5 (car $5 = 00\ldots00101$).

Le nombre d'états d'un AFD est limité à $2^{32}-2$, la valeur $2^{32}-1$ servant de valeur 'vide'. L'ensemble des états finals est représenté par un entier multi-précision dont chaque vide à 1 représente un final final pour l'indice correspondant.

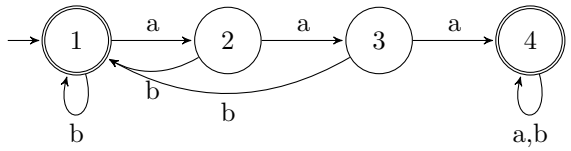
1. Récupérer les fichiers `afd.h, afnd.c, simul.c, makefile`, compiler et tester le programme.

2. Écrire une fonction `void afd_finit(char *nomfichier, afd *A)` qui initialise un afd à partir d'un fichier texte au format suivant:

   ```
   nbr_d_etats alphabet nbr_etats_finals
   etat_initial
   etat_final_1 etat_final_2 ... etat_final_n
   etat_i_11 symb_j_1 etat_i_12
   ```

```
etat_i_21 symb_j_2 etat_i_22
   .
   .
   .
etat_i_n1 symb_j_n etat_i_n2
```

par exemple le graphe suivant sera représenté par le fichier ci-contre:




```
5 ab 2
1
2
1 4
1 a 2
1 b 1
2 a 3
2 b 1
3 a 4
3 b 1
4 a 4
4 b 4
```

3. Compléter la fonction `int afd_simul(char *s, afd A)` qui simule le traitement d'une chaîne de caractère par un AFD.
4. Compléter le fichier `afn.c` afin de pouvoir manipuler les AFN.
5. Écrire une fonction `ullong afn_epsilon_fermeture(afn A, ullong R)`
6. Écrire une fonction `void afn_determinisation(afn A, afd *D);` qui construit une version determinisée de l'AFN `A`.