# I61- Compilation et théorie des langages

Licence 3 - 2016/2017

## 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 l'alphabet $\{a, b, c, \ldots, z, -\}$ où $-$ représentera les $\epsilon$-transitions. Pour cela on utilisera la structure suivante:

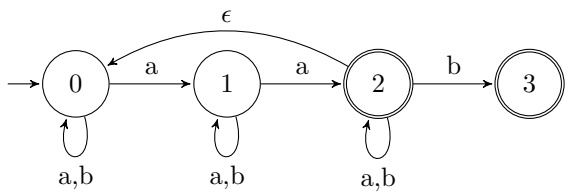
```
typedef unsigned long long int ullong;
typedef struct {
  ullong init, final;
  ullong trans[ETAT][ALPHA];  // q -l-> trans[q][l]
} afnd;
```

On se limite ici à des automates avec au plus 64 états. Un ensemble d'états est représenté par un `ullong`, un mot de 64 bits, donc chaque bit à 1 représente la présence de l'état correspondant dans l'ensemble. L'attribut `init` donc un nombre comprenant un seul bit à un correspondant à l'état initial et `final` un nombre dont chaque bit à 1 représente un état final. De même chaque case de la table de transition

1. Récupérer les fichiers `afnd.h, afnd.c, simul.c, makefile`, compiler et tester le programme.
2. Écrire un fonction `void finitafn(afnd *A, char * f)` qui initialise un afn à partir d'un fichier texte au format suivant:

   ```
   etat initial
   liste des etats finaux
   liste des transitions
   ```

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




```
0
2 3
0 a 0
0 b 0
0 a 1
1 a 1
1 b 1
1 a 2
2 a 2
2 b 2
2 b 3
2 - 0
```

3. Écrire une fonction `ullong epsilon(int s, afnd * A)` qui retourne l'$\epsilon$ fermeture de l'état `s`.

4. Écrire une fonction `int utile(int s, afnd *A)` qui retourne 1 si `s` est un état utile (c'est-à-dire connecté à un autre état) et 0 sinon.

5. Écrire une fonction `afd * determinisation(anfd *A)` qui retourne un pointeur vers une version determinisée de l'afn `A`. On se limitera à des afn d'au plus 25 états et on créera un tableau de $2^{20}$ cases. On devra créer une nouvelle structure de données `afd` comme suit:

```
typedef struct etat{
  int mark;
  int final;
  int *trans[27];
} etat;

typedef struct etat * afd;

afd D;
D = calloc(1<<25, sizeof(*afd))
```

   L'attibut `mark` doit être initialisé à 0 pour signifié que l'état en question n'a pas été ajouté à l'automate. L'attribut `final` vaut 1 si l'état est final et 0 sinon et la table `trans` est la table des transitions de l'état en question.

6. Écrire une fonction `afd *minimisation(afd *D)` qui retourne une version minimisée de l'automate `D`.

7. Réécrire la fonction `main` pour que le programme produit prenne en paramètre un nom de fichier et une chaîne de caractères et simule le déroulement de l'automate correspondant.