Projet - Grammaires et langages

Transformation et minimisation d'automate

Faculté des Sciences et Techniques

Licence 3 Informatique

Encadré par :
Pr Philippe GABORIT

Réalisé par :
Nicolas NAN
Jérémy DRON

1 Algorithme de transformation d'un AFN en AFD

En théorie des automates, il existe plusieurs types d'automates ayant des caractèristique bien défini :

- **AFD [Automate Fini Déterministe] :** Une seule transition au plus pour chaques symboles depuis un état
- **AFN [Automate Fini Non-déterministe] :** Automate pouvant posséder plusieurs transitions pour chaques symboles depuis un état
- **AFN-ϵ [Automate Fini Non-Déterministe avec ϵ transition] :** Automate possédant des transitions avec le symbole du vide

Il est possible de passer d'un type d'automate à un autre en lui appliquant quelques modifications. Pour passer d'un AFN à un AFD, il faut créer des super-états qui seront la combinaisons de plusieurs états. Un état sera final si un des états qui le compose l'est.Notre algorithme ne prendra en entrée que les automates suivant :

A={Q,ϵ,δ,q_0,F}
Q => L'ensembles des états de l'automate avec $|Q| <= 10$
ϵ => L'alphabet de symbole fini avec $\epsilon = a, b$
δ => Transitions de l'automate
q_0 => L'etat initial de l'automate
F => L'ensemble des états finaux

1.1 Représentation d'un automate et structure de données

Dans un premier temps, il nous a fallu pouvoir représenter un automate de manière que notre programme puisse le comprendre et l'utiliser. Pour cela, nous avons réaliser pour chaques symboles une matrice. Ces matrices continennent uniquement des "0" ou des "1" et chaques lignes indiquent l'état de départ d'une transition et chaque colonne l'état d'arrivé.

	0	1	2	3
0	1	1	0	0
1	1	1	1	0
2	1	1	1	0
3	1	1	0	0

Le tableau ci-dessus représente un matrice concernant un symbole d'un automate.

1.2 Principe de l'algorithme

1.3 Programme et execution

2 Algorithme de minimisation d'un automate

A présent, nous cherchons à minimiser un AFD, cest-à-dire que nous cherchons à avoir un automate avec le minimum d'état qui a le même comportement que celui actuel. En effet, certains automates ont des états qui peuvent être combiner entre-eux. Notre algorithme ne prendra en entrée que les automates suivant :

A={Q,ϵ,δ,q_0,F}
Q => L'ensembles des états de l'automate avec $|Q| <= 10$
ϵ => L'alphabet de symbole fini avec $\epsilon = a, b$
δ => Transitions de l'automate
q_0 => L'etat initial de l'automate
F => L'ensemble des états finaux

2.1 Représentation d'un automate et structure de données

Premièrement, nous avons cherché une représentation, pour notre AFD, que notre programme pourrait facilement utiliser. Nous nous sommes alors basés de ce que nous avions trouvé dans la première partie.
Nous avons pour chaque symbole de notre alphabet une matrice lui correspondant. Dans le cas précédent notre

matrice contenait l'information de quels états de départ pointaient sur quels états d'arrivés. Dans ce cas présent, c'est l'inverse, nous cherchons à savoir quels états est pointés par quels état. Du au fait, que nous travaillons uniquement avec des AFDs nous pouvons concidérer que :

Soit M = "La matrice des transitions de départ vers arrivé" alors M^t = "La matrice des transitions des arrivés vers les départs"

Par ailleurs, pour minimiser notre automate nous avons utilisé l'algorithme vu en cours. Dans notre programme nous utiliserons une file qui contiendra les couples d'états à vérifier. Nous avons fait le choix de faire notre programme dans le paradygme orienté objet pour une optimisation de la compléxité cognitive. Nous avons réaliser le diagramme de classe de notre programme pour faciliter la compréhension de la structure du programme.

2.2 Principe de l'algorithme

En premier lieu, l'utilisateur doit créer un objet de type *Automate* et un objet de type *Minimiseur* puis il peut utiliser deux méthodes : une permettant de récuperer les états non distingables et l'autre le nombre d'état de l'automate minimisé.

Concretement, l'objet *Automate* prend un fichier en entrée qui contient des informations sur l'automate ainsi la représentation de l'automate exprimé plus haut. Le fichier doit contenir sur la premier ligne le nombre d'état de l'automate, sur la deuxieme ligne les états terminaux et finalement les matrices de transitions sur les lignes suivantes. Le fichier ressemble peut ressembler à ça :

```
 1  7
 2  2
 3  0 1 0 0 0 0 0
 4  0 0 0 0 0 1 0
 5  0 0 0 0 0 0 0
 6  0 0 0 0 0 0 1
 7  0 0 1 0 0 0 0
 8  0 0 0 0 0 1 0
 9  0 0 0 0 0 1 0
10  0 0 0 0 1 0 0
11  0 0 1 0 0 0 0
12  0 0 1 0 0 0 0
13  0 0 0 0 1 0 0
14  0 0 0 0 0 1 0
15  0 0 0 1 0 0 0
16  0 0 1 0 0 0 0
```

2.3 Programme et execution