Projet Grammaire et Langages

Nicot Fabien & Thuillier-Le Gac/Le Gac Pierre

2015-2016

I. Algorithme de transformation AFN sans epsilon transition vers AFD

Cet algorithme fonctionne de la manière suivante : on passe un automate AFN sans epsilon transition au programme et il nous rend un AFD.

L'automate AFN que l'on souhaite passer au programme doit être inscrit dans le fichier afn.csv en suivant les directives suivantes (un exemple est déjà présent dans le fichier et c'est celui que nous allons utiliser pour présenter l'algorithme) :

-une ligne représente un état et chaque information est séparé par une ,.

-tout d'abord il faut saisir le numéro de l'état. Dans notre cas on saisit tout d'abord 1

-ensuite il faut saisir une liste de nombre valant 0 ou 1. La liste contient autant de nombres qu'il y a d'états et chaque nombre est séparé par un ;. Le 1 indique que l'état dans lequel on est permet d'aller à l'état de la liste par la lettre a. Dans notre exemple on sait qu'il y a quatre état et qu'à partir de l'état 1 on a deux possibilités pour a : soit on reste sur l'état 1, soit on va à l'état 2.

-on fait de même mais pour la lettre b.

-on indique ensuite par 0 (faux) ou 1 (vrai) si l'état est un état final

-et on répète ces instructions pour chaque état

Notre exemple nous donne donc ceci :

1,1;1;0;0,1;0;0;0,0
2,0;0;1;0,0;0;0;0,0
3,0;0;0;0,0;0;0;1,0
4,0;0;0;0,0;0;0;0,1

Maintenant que nous avons inscrit notre AFN dans le fichier nous pouvons lancer le programme. Celui-ci va d'abord récupérer les données de notre AFN. Pour chaque état qu'il trouve, le programme crée un objet Etat qu'il ajoute dans un vector.

Une fois que l'on a récupéré tous les états le programme crée un objet SuperEtat qu'il ajoute dans un vector. Il parcourt ensuite tout les états du vector Etat :

-il crée deux vectors : vA et vB qui sont initialisés avec autant de valeurs qu'il y a d'états, tous ayant pour valeur 0

-il parcourt une nouvelle fois le vector Etat du premier état à celui en cours. Cette boucle permet de mettre les valeurs de vA et vB nécessaire. La boucle récupère pour chaque état les valeurs des listes a et b. Dès qu'il trouve un 1 dans une des listes il met à jour le vecteur vA ou vB.

-on affiche le contenu de vA et vB.

-on effectue un test afin de savoir si le dernier état auquel peut accéder l'état en cours est final. Si c'est le cas tous les SuperEtat qui seront crée à ce passage de là seront finaux.

- on parcourt ensuite la liste des SuperEtats. Pour chaque SuperEtat on parcourt une autre fois la liste des SuperEtats et on essaye de trouver si on trouve une correspondance entre vA/vB et le vecteur du SuperEtat. Si c'est le cas on change la direction du SuperEtat si

celui-ci n'a pas déjà été changé. Si on ne trouve pas de correspondance on crée un SuperEtat et on change la direction du SuperEtat en cours par celui crée.

Une fois que cela est fait on obtient notre automate AFD. Le programme nous l'affiche. Dans l'exemple que nous avons utilisé au début voici ce qui s'affiche à la fin :
1000 ; A : 2 ; B : 1 ; Final : 0
1100 ; A : 3 ; B : 1 ; Final : 0
1110 ; A : 3 ; B : 4 ; Final : 0
1001 : A : 3 ; B : 4 ; Final : 1
La première suite de 1 et de 0 correspond au vecteur de l'état. La valeur représentant A et B correspond au prochain état qui doit être atteint. La valeur de Final est à 1 si l'état est un état final.

II. Algorithme de minimisation

1/ Explication de l'algorithme

Cet algorithme permet de fusionner les états qui sont identiques.

Cet algorithme prend en entrée une AFD, mais ressort un AFD minimiser au maximum, c'est-à-dire qu'il n'y a que des états différenciables à la fin.

L'AFD qui est passé en entrée et stocké sous la forme d'un fichier .csv, ce fichier contient une ligne par état, ensuite chaque colonne correspond à une information concernant un état. La première colonne correspond aux numéros de l'état, la deuxième correspond à l'état vers lequel la transition par a et effectue (0 s'il n'y a pas de transition en a), la troisième correspond à l'état vers lequel la transition par b est effectué (0 s'il n'y a pas de transition en a) et la dernière indique si l'État est final (1) ou non (0).

Lignes sont délimitées par un retour à la ligne tandis que les colonnes sont séparées par des virgules. Vu que l'on prend un AFD en entrée il n'y a qu'une seule transition pour un symbole contrairement à l'AFN de l'exercice précédent.

Voici l'exemple que nous avons pris :
1,2,5,0
2,6,3,0
3,0,3,1
4,7,5,0
5,3,6,0
6,6,4,0
7,6,3,0

Donc la première étape de l'algorithme est de lire le fichier et de créer les classes états correspondants, comme dans l'exercice précédent à l'exception que notre état dit EtatAFD n'a pas de vecteurs d'entiers pour les transitions de a vers b mais juste un entier.

Voici le résultat que l'on obtient :

```
Numéro : 1
a : 2
b : 5

Numéro : 2
a : 6
b : 3

Numéro : 3
a : 0
b : 3
Final

Numéro : 4
a : 7
b : 5

Numéro : 5
a : 3
b : 6

Numéro : 6
a : 6
b : 4

Numéro : 7
a : 6
b : 3
```

Ensuite il a fallu initialiser le tableau des Distinguables. Pour ça j'ai d'abord créé une matrice carrée de caractère de la même dimension que le nombre de nos états. La lettre D correspondant pour les couples distinguables et N pour les autres
Pour initialiser le tableau, il faut trouver nos états finaux afin de les marquer Distinguable avec un non final.

On obtient donc se tableau car l'état 3 est terminal :

```
1
2
3 D  D
4       D
5       D
6       D
7       D
  1  2  3  4  5  6  7
```

Après cela il faut initialiser la file des couples distinguables. Nous avons utilisé le type queue de la bibliothèque du même nom de C++. Nous avons aussi créé un type couple afin de stocker les deux états du couple. Nous avons donc fait une file de couple.

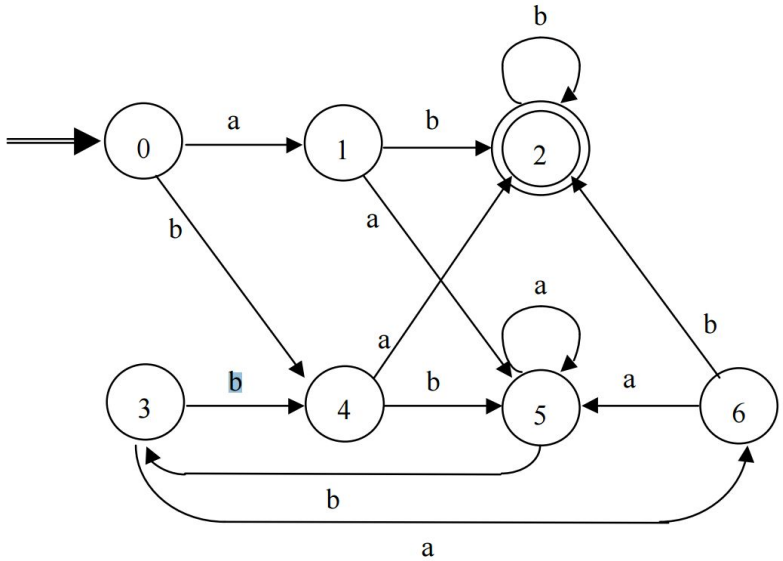
Dans cette deuxième partie il nous a fallu pour chaque élément de la file retrouvé les antécédent des états du couple dépilé car ils sont distinguables entre eux, c'est-à-dire si le couple d'états 1 et 2 sont distinguables alors les antécédents de 1 en a et les antécédents de 2 en s sont distinguables aussi, c'est pareil pour b. Il faut aussi rajouter les nouveaux couples distinguables dans la file et continuer jusqu'à ce que la file soit vide.

On obtient se tableau :

```
1
2  D
3  D  D
4     D  D
5  D  D  D  D
6  D  D  D  D  D
7  D     D  D  D  D
   1  2  3  4  5  6  7
```

Les couples qui ne sont pas désignés d'un D ne sont donc pas distinguable et peuvent donc être fusionner.

2/ Exemple



Sources : http://www-igm.univ-mlv.fr/~desar/Cours/automates/ch2.pdf

Initialisation :

0							
1							
2	X	X					
3			X				
4			X				
5			X				
6			X				
	0	1	2	3	4	5	6

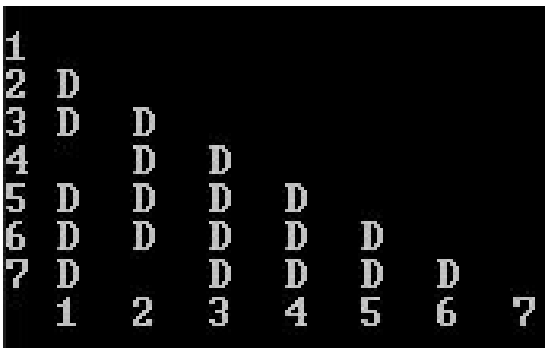
File :

2	2	3	4	5	6	0	5	5	0	0	3	3	4	4	4	3	0	3
0	1	2	2	2	2	4	1	6	1	6	1	6	1	5	6	4	5	5

0							
1	X						
2	X	X					
3		X	X				
4	X	X	X	X			
5	X	X	X	X	X		
6	X		X	X	X	X	
	0	1	2	3	4	5	6

Les états 3 et 0 peuvent être fusionner et 6 et 1 peuvent aussi être fusionner

Maintenant comparons avec le résultat obtenu :



C’est le même exemple sauf que les états commencent à 1 et pas 0.

Nous remarquons que nous avons le même tableau de distinguables.