

Mathématiques pour l'Informatique

Format du fichier

L'automate est entré dans le fichier sous forme de tableau de transitions. Cela nous permet d'avoir à la fois les états existants, leurs statuts (Entrée / Sortie / Rien) et les transitions.

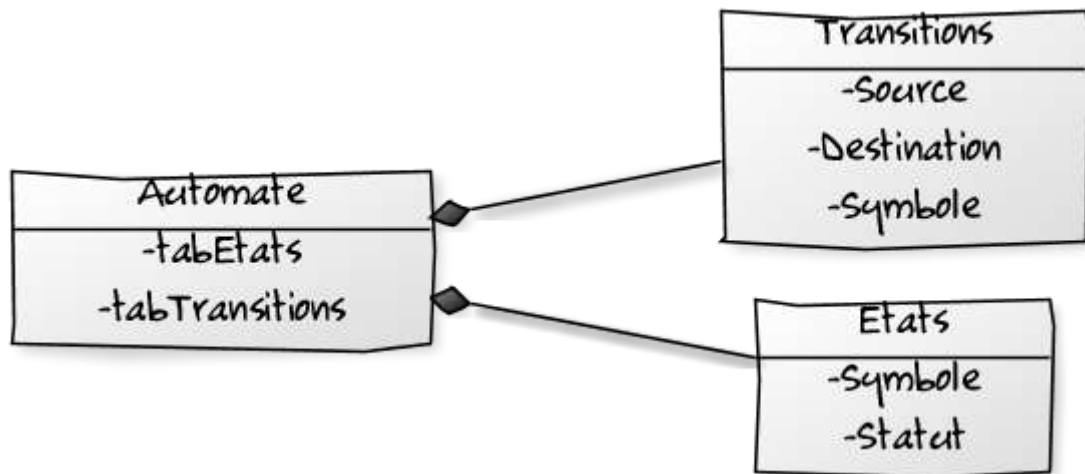
	Etat		a	b	c	d
E	1	1	2	3	4	
-	2	5	2	3	4	
-	3	5	5	3	4	
S	4	5	5	5	4	
-	5	5	-	-	-	

1. Exemple de fichier d'automate

Mécanisme de lecture

1. On récupère le contenu (Vérification si le fichier est vide)
2. On compte le nombre de lignes (Vérification nombre d'états & si le fichier est vide)
3. On parcourt chaque ligne
 - a. On récupère les informations
 - b. On compte le nombre de transitions (Vérification s'il n'y a pas de transitions)
 - c. On crée les transitions
 - d. On crée les états (Vérification entrées / sorties)

Représentation en mémoire des automates



2. Structure du programme

Standardisation

```
nouvel etat « I » (Entrée)
pour chaque etat existant
    si statut = « Entrée » ou « Entrée / Sortie »
        pour chaque transition ayant pour source cette entrée
            si la transition n'existe pas déjà
                On crée la transition
            fin
        fin
    statut = « Rien » ou « Sortie »
fin
fin
```

Détermination

etatsATraiter

0234	0	2	3	4
24	2	4		

```
nouvel automate « AFD »
nouvel etat « I » (Entrée)
nouveau tableau etatsATraiter[]

// Création de l'entrée de l'AFD
pour l'entrée de l'AFS
    pour chaque symbole de transition
        etatsATraiter[].taille = etatsATraiter[].taille + 1
        pour chaque transition ayant pour source cette entrée
            On copie la transition dans l'AFD
        fin
    fin
    si on a ajouté une transition
        etatsATraiter.symbole = transition.derniere.symbole
        etatsATraiter.taille = etatsATraiter.taille + 1
        AFD.transitions.taille = AFD.transitions.taille + 1
    fin
fin

// Mise en place de la liste des etatsATraiter
pour chaque symbole de transition
    pour chaque transition de ce nouvel etat
        etatsATraiter[].taille = etatsATraiter[].taille + 1
    fin
fin

// Traitement des etats etatsATraiter
tant que etatsATraiter[0] n'est pas vide
    si l'état n'existe pas déjà
        On crée le nouvel état
        pour chaque symbole de transition
            pour chaque composant etatsATraiter[]
                pour chaque transition ayant le composant en
                    source
                    On copie la transition dans l'AFD
                fin
            fin
            On crée le nouvel état regroupant les composants
        fin
    fin
    On supprime etatsATraiter[0]
fin
```

Minimisation

On travaille avec 4 tableaux :

- **Etats** et **transition** : correspond à l'automate déterministe et complété)
- **etatsNew** et **transitionNew** : présentent la même structure qu'**etats** et **transitions**. Ils représentent l'automate en cours de minimisation et une fois l'algorithme finis, l'automate minimisé.
- **Etats_temp** : tableaux temporaire qui servira pour la formation des groupes et l'affichage de l'automate minimisé.

Déroulement de la minimisation :

- **Initialisation de départ** : remplir pour la 1^{ère} fois **etatsNew** et **transitionNew**.

```
Pour compteur de 0 au nombre d'état
  Rempli les symboles du tableau etatNew avec les symboles du
  tableau etats
  Si valeur du statut etats = 2 OU 3 (état sortie)
    Valeur du statut etatsNews sera 1 (groupe I)
  Sinon
    Valeur du statut etatsNews sera 2 (groupe II)
```

On remplit ensuite le tableau **transitionNew**.

- **remplirTransition** : remplit le tableau **transitionNew** avec les nouveaux groupes formé précédemment (**etatsNew**)

```
Pour compteur de 0 au nombre total de transitions
  Remplir la valeur de l'état source du tableau transitionNew
  Remplir le symbole du tableau transitionNew
  Pour compteur de 0 au nombre d'états
    Si symbole de etatsNew = desti de transition
      Remplir desti de transitionNew avec le statut du
      tableau etatsNew (groupe)
    On sort de la boucle
```

Viens ensuite la boucle principale de la fonction qui minimise

- **Boucle principale** : de la minimisation tant que la fonction **minimiser** = **true**
- **Minimiser** : Si il y a deux groupes qui n'ont pas les mêmes transitions, la minimisation n'est pas finis on renvoi true.

```
Pour compteur de 0 à nombre d'états
  Pour compteur de 0 à nombre d'états
    Si 2 groupes sont identiques
      On compare toutes les transitions des deux groupes
    Si les 2 groupes ne sont pas identiques
      Renvoi true
Renvoi false
```

- **formationGroupes** : forme les différents groupes

```
Pour compteur de 0 à nombre d'états
  Pour compteur de 0 à nombre d'états
    Si 2 groupes sont identiques
      On compare leurs transitions
    Si transitions égaux
      Si pas encore attribué un groupe
        Donne un numéro de groupe à etats_temp
aux id des groupes qu'on parcourt
On incrémente groupe
```

On remplit **etatsNew** avec valeurs **etats_temp**

- **VerifMinimisation** : vérifie si l'automate sortie a été minimisé ou n'a pas été modifié
 - ➔ Si 2 groupes sont identiques renvoi **true**
 - ➔ Sinon renvoi false

Si **true** affiche le tableau de transitions de l'automate minimisé

Reconnaissance de chaîne de symboles

Automate : lectureMot(chaine mot)

```
pour chaque les etats
    si l'état est une entrée
        si etat.lectureMot(mot) = vrai
            retourner vrai
        fin
    fin
fin
retourner faux
```

Etat : lectureMot(chaine mot)

```
si le mot est vide ou si c'est le mot vide
    si l'état est une entrée
        retourner vrai
    fin
fin

si le mot est vide
    retourner faux
fin

pour chaque les transitions
    si le mot n'est pas vide
        si le symbole de la transition = première lettre du mot
            si etat.lectureMot(mot sans la première lettre) = vrai
                retourner vrai
            fin
        fin
    fin
fin
retourner faux
```