

TAI de Mathématiques pour l'Informatique

TAI de Mathématiques pour l'Informatique

Sommaire

Introduction.....	3
Un peu de vocabulaire	4
Les structures de données	5
La classe Automate	5
La structure Etat	6
La structure Transition	6
Les fonctions de manipulation	7
Initialiser un Automate.....	7
L'affichage d'un Automate	8
L'automate et son état.....	9
Est-il complet ?	9
Est-il déterministe ?	10
Initialiser un Etat.....	11
Initialiser une Transition.....	11
Faire des opérations sur les automates	12
Déterminiser.....	12
L'automate déterminisé	12
L'état déterminisé.....	13
La transition déterminisée.....	13
Compléter.....	14
Reconnaître un mot	14
Conclusion	15

TAI de Mathématiques pour l'Informatique

Introduction

Dans le cadre de notre cours de Mathématiques pour l'Informatique, nous avons pu, tout au long de ce semestre, découvrir les automates finis et leur fonctionnement. Nous avons eu l'occasion d'employer nos nouvelles connaissances d'abord sur papier, puis sur ordinateur avec ce TAI.

Ainsi, il nous a été demandé de coder en programme en C ou en C++ qui soit capable, à partir d'un fichier de données de pouvoir déterminer, compléter et minimiser les automates finis (synchrone et asynchrone) qui nous étaient fournis. De plus, nous devons pouvoir, à partir de n'importe quel mot saisi, vérifier si l'automate pouvait le lire.

Dans ce rapport, nous allons vous présenter le programme, codé en C++, que nous avons produit afin de répondre aux attentes de nos professeurs.

TAI de Mathématiques pour l'Informatique

Un peu de vocabulaire

Avant de nous intéresser au programme en lui-même, présenter les notions que nous allons étudier nous a paru pertinent.

On appelle automate fini une machine abstraite, constitué d'un nombre d'états fini et de transitions reliant ces états. Il reconnaît un langage rationnel c'est-à-dire les langages que l'on peut obtenir à partir du mot vide et des lettres auxquels on applique les opérations rationnels (union, produit et étoile de Kleene).

Déterminer un automate fini (donc obtenir son automate fini déterministe) consiste, à partir d'un automate fini non déterministe, à créer un automate reconnaissant le même langage n'ayant qu'un seul état initial et qui, pour chaque état et chaque lettre, a au plus une transition partant de l'état étiquetée par la lettre.

Compléter un automate fini déterministe c'est, pour chaque lettre d'un état n'ayant pas de transition, lui rajouter une transition vers un état « poubelle » (généralement noté P), état non co-accessible (qui ne peut pas arriver dans un état terminal) ayant comme transitions en chaque lettre une boucle sur lui-même.

Minimiser un automate fini déterministe complet signifie qu'à partir de l'automate fini déterministe complet, on va déterminer l'automate équivalent (c'est-à-dire qui reconnaît le même langage) avec le nombre minimal d'état. Pour un automate fini déterministe, il existe un et un seul automate fini déterministe complet minimal.

Enfin, on dit qu'un automate reconnaît un mot si, à partir de l'état initial, on peut, en prenant les lettres une par une et dans leur ordre dans le mot, arriver à un état terminal.

TAI de Mathématiques pour l'Informatique

Les structures de données

Pour ce projet, nous avons eu besoin de 3 structures de données différentes : une pour l'automate, une pour l'état et une pour la transition. Nous vous présenterons par la suite les fonctions et méthodes liées à ces structures

La classe Automate

Voyons d'abord comment est représenté un automate « physiquement ». Celui-ci a un nom, un nombre d'états, un nombre de transitions, un ou des état(s) initial(aux), un ou des état(s) terminal(aux). Une fois cela mis en lien avec la syntaxe des fichiers que les professeurs nous ont demandé, nous avons pu mettre en classe une classe que nous jugeons pertinente.

Ligne 1 : nombre de symboles dans l'alphabet de l'automate.

Ligne 2 : nombre d'états.

Ligne 3 : nombre d'états initiaux, suivi de leurs numéros.

Ligne 4 : nombre d'états terminaux, suivi de leurs numéros.

Ligne 5 : nombre de transitions.

Lignes 6 et suivantes : transitions sous la forme

<état de départ> <symbole> <état d'arrivée>

Voilà la syntaxe qui nous était demandée de respecter. Et voilà comment nous l'avons traduit dans une classe en C++ :

Ligne 1	→	Entier
Ligne 2	→	Entier
Ligne 3	→	Entier + tableau dynamique de chaîne de caractère
Ligne 4	→	Entier + tableau dynamique de chaîne de caractère
Ligne 5	→	Entier
Ligne 6 ..	→	Liste doublement chaînée d'Etat, une structure nouvellement créée

Nous stockons donc les informations principales dans des entiers afin qu'elles soient simples à manipuler. Pour les états initiaux et terminaux, nous choisissons d'utiliser en complément de l'entier représentant leur quantité des tableaux dynamiques d'entier, que l'on définit respectivement grâce au nombre d'états initiaux et terminaux afin de stocker les noms de ces états initiaux et terminaux.

TAI de Mathématiques pour l'Informatique

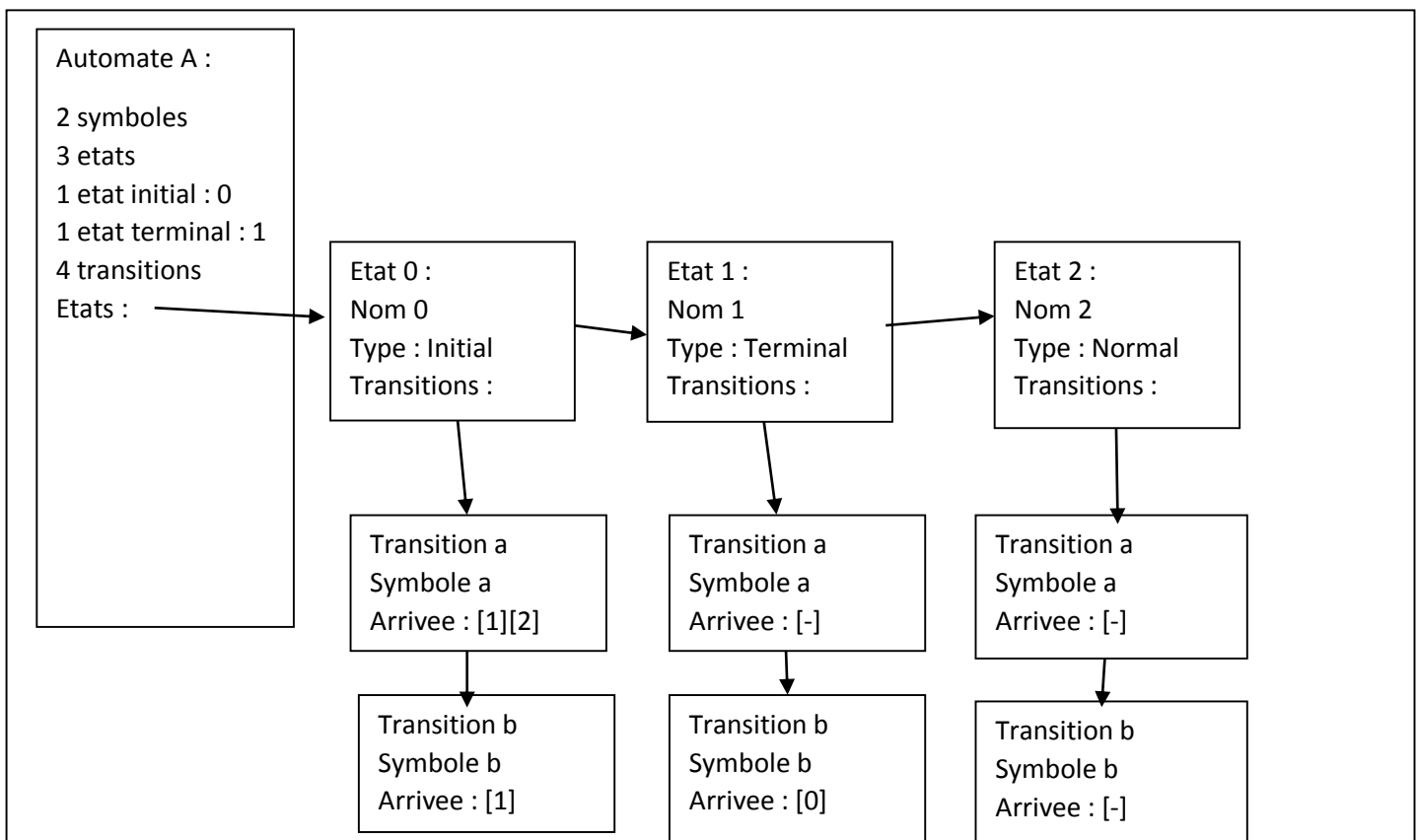
Enfin, nous avons défini la structure Etat, pour pouvoir stocker nos états de la façon que nous avons jugé la meilleure. Nous employons une liste doublement chaînée afin de pouvoir facilement manipuler les états (ajouts, suppressions), sans avoir besoin de connaître préalablement le nombre qu'en contient l'automate.

La structure Etat

Ici, on a 2 chaînes de caractères, une stockant le nom de l'état, la seconde stockant son type (initial, terminal, etc ...). De plus, une liste doublement chaînée de Transition permet, pour chaque état, de définir ses transitions.

La structure Transition

Enfin, la structure Transition comporte 2 champs : une chaîne de caractère qui stocke le symbole de la transition et un tableau dynamique de chaîne de caractère qui va stocker l'ensemble des états d'arrivées de la transition à partir d'un état et d'un symbole donnée.



TAI de Mathématiques pour l'Informatique

Les fonctions de manipulation

Maintenant que nous savons comment sont stockés les automates et leurs composants en mémoire, voyons comment le programme les manipule. Pour chacune des structures de données, il y a deux façons de l'initialiser, une pour l'initialisation à partir du fichier, la seconde pour la détermination. Nous aborderons ici uniquement les initialisations à partir du fichier, la seconde façon sera abordée lorsqu'on s'intéressera à la détermination.

Initialiser un Automate

Grâce la structure de fichier demandée par nos professeurs, l'initialisation d'un automate à partir d'un fichier est particulièrement simple. En effet, pour initialiser les 7 premières données membres, il va suffire de lire et d'interpréter le fichier. A l'aide de la fonction `getline`, on attribue à chaque donnée sa valeur et on remplit également les tableaux contenant les noms des états initiaux et finaux. Puis, on va initialiser les différents états de l'automate.

```
Pour l'ensemble des états
{
    Si l'état est initial, on le crée comme étant initial
    Si l'état est terminal, on le crée comme étant terminal
    Si l'état est terminal et initial, on le crée comme étant terminal et initial
    Sinon, on crée l'état comme étant normal
}
```

Pour les automates, nous avons également fait le choix de surcharger l'opérateur d'affichage.

TAI de Mathématiques pour l'Informatique

L'affichage d'un Automate

Grâce aux possibilités du C++, nous avons choisi de surcharger l'opérateur d'affichage pour que l'affichage se fasse simplement. On va d'abord afficher les informations basiques sur l'automate, c'est-à-dire les informations stockées dans les entiers et dans les tableaux dynamiques. Ensuite, on va afficher la table de transition de l'automate. Pour cela, on va d'abord afficher l'alphabet de l'automate. Ensuite, en parcourant les listes chaînées d'Etats et de Transitions au fur et à mesure, on va, pour chaque état de l'automate, afficher les états d'arrivés en dessous du symbole qui lui correspond. Si il n'y a pas de transitions, on affiche « - ».

Exemple d'affichage pour l'automate n°12

```
Automate n : 12
Alphabet de 2 symboles
Automate avec 6 etats
1 etat initial qui est 0
2 etats terminaux qui sont 0 5
Cet automate a 8 transitions

  a    b
0     1  -
1     2  2
2     3  -
3     4  4
4     5  -
5     1  -
```

On va également voir les fonctions permettant de savoir si les automates sont complets et/ou déterministes.

TAI de Mathématiques pour l'Informatique

L'automate et son état

Ici, nous allons présenter les fonctions qui permettent de savoir si un automate est complet et/ou déterministe.

Est-il complet ?

Un automate est complet à condition que chaque transition ait au moins un état d'arrivée (ce qui, dans notre programme, se traduit par une chaîne de caractère différente de « - »). Ici, on utilise une fonction qui retourne un booléen selon si l'automate est complet ou non. Même si c'est une contre-performance algorithmique, nous avons choisi de vérifier toutes les transitions de tous les états de l'automate. Ainsi, l'algorithme est le suivant :

```
Pour l'ensemble des états
{
    Pour l'ensemble des transitions
    {
        Si la transition a pour état d'arrivée « - »,
        alors l'automate n'est pas complet
    }
}

Si l'automate n'est pas complet, on retourne faux
Si il est complet, on retourne vrai
```

TAI de Mathématiques pour l'Informatique

Est-il déterministe ?

Un automate est déterministe a deux conditions : qu'il n'ait qu'un seul état initial et que, pour chaque état et chaque lettre, il est au plus une transition étiquetée par cette lettre (dans le code, on traduit cela grâce au tableau contenu dans la structure transition que l'on initialise préalablement intégralement à « - ». Ainsi, si, à l'indice [1], le contenu n'est pas « - », l'automate n'est pas déterministe). Encore une fois, on a choisi une fonction retournant un booléen, et on vérifie de nouveau l'intégralité des états et des transitions. Ainsi, la fonction suit l'algorithme suivant :

```
Si l'automate a plus d'un état initial, on retourne faux

Pour l'ensemble des états
{
    Pour l'ensemble des transitions
    {
        Si la transition a plus d'un état d'arrivée, alors
        l'automate n'est pas déterministe
    }
}

Si l'automate n'est pas déterministe, on retourne faux
Si il est déterministe, on retourne vrai
```

Retour des fonctions appliquées à l'automate 12 :

```
L'automate est deterministe
L'automate n'est pas complet
```

TAI de Mathématiques pour l'Informatique

Initialiser un Etat

Initialiser notre structure Etat revient à affecter à l'état son nom, son type et l'intégralité de ses transitions, symbole par symbole. Une fois l'état initialisé, on va afficher son nom et son type avant l'affichage de la table de transition. Voyons ce que cela donne pour l'automate 12 :

```
Nom : 0
Type : Initial et terminal
Nom : 1
Type : Normal
Nom : 2
Type : Normal
Nom : 3
Type : Normal
Nom : 4
Type : Normal
Nom : 5
Type : Terminal
```

Initialiser une Transition

Voilà l'initialisation la plus technique. On commence par affecter à la transition son symbole et à allouer de la mémoire pour son tableau dynamique de chaîne de caractère qui contiendra le nom des états d'arrivée. On initialise ensuite ce tableau intégralement avec « - ». Puis, on va ouvrir le fichier en mémoire afin de saisir les transitions :

```
On se déplace jusqu'à la 6ème ligne du fichier (le début des transitions)
Tant que l'on est pas à la fin du fichier (signalée par un « ! »)
{
    On récupère les caractères qui représentent l'état de départ et le symbole
    Si on est sur le bon état de départ et le bon symbole
    {
        On récupère le caractère suivant et on le rentre dans le tableau à la 1ère case libre
        Puis on passe à la ligne suivante
    }
    Sinon, on passe à la ligne suivante
}
```

TAI de Mathématiques pour l'Informatique

Faire des opérations sur les automates

Cette partie va présenter les opérations de détermination, de complétion et de reconnaissance de mots.

Déterminiser

On arrive ici sur la présentation de la seconde initialisation existante dans notre programme, l'initialisation de la détermination. Nous verrons d'abord comment on initialise l'automate, puis l'état et enfin la transition.

L'automate déterminisé

On commence par lui affecter son nom et son alphabet, qui sont les mêmes que pour l'automate non déterministe. Son nombre d'état initial est de 1 vu que l'automate est déterministe. On va d'abord créer l'état initial de l'automate déterministe :

```
On concatène le nom de tous les états initiaux de l'automate non déterministe pour avoir le nom
Pour chaque lettre dans l'alphabet
{
    On récupère les états initiaux de l'automate non déterministe
    On va stocker dans un tableau les états d'arrivés de l'état pour cette lettre (à condition qu'il soit initialisé)
    Quand on a parcouru toutes les lettres, on va créer l'état initial de l'automate déterministe en passant le tableau en
paramètre
}
```

TAI de Mathématiques pour l'Informatique

Ici, nous avons donc volontairement traité l'état initial du nouvel automate différemment. Voyons maintenant comment nous initialisons les autres états :

```
On stocke le nom de l'état initial pour ne pas le retraiter
On stocke dans un tableau le nom de chaque état que l'on a à traiter, qui sont les états d'arrivée de l'état précédent
{
    On va récupérer le nom du premier état à traiter
    On va séparer cet état en un ou plusieurs sous-états selon si il est nouveau ou non et on stocke ces sous-états
    Tant qu'ils ne sont pas tous traités
    {
        On récupère le symbole de chaque transition
        Ainsi que son ou ses états d'arrivée que l'on stocke dans un tableau
        On supprime alors le sous-état que l'on vient de traiter
    }
    On crée alors notre nouvel état
    Puis, pour chacun des états d'arrivée du nouvel état
    {
        Si il n'est pas déjà dans la liste pour être traité ou qu'il a été traité, on le rajoute à la liste des états à déterminer
    }
    Enfin, on supprime le macro-état que l'on vient de traiter
}
```

Quand on a créé tous les états, on affecte les différents champs encore vide de notre automate

Maintenant, voyons comment on crée un état lors de la détermination.

L'état déterminisé

Ici, rien de bien compliqué, on affecte à l'état son nom et son type, puis, comme pour l'initialisation de base, pour chaque symbole, on crée la transition correspondante.

La transition déterminisée

On a passé un paramètre de la fonction le tableau qui a stocké les états d'arrivée des différentes transitions. Il ne reste qu'à concaténer l'ensemble des états d'arrivée présent dans chaque ligne du tableau pour pouvoir les affecter à leur lettre (en première colonne de ce même tableau) dans le tableau d'arrivée de la transition.

TAI de Mathématiques pour l'Informatique

Compléter

C'est une manipulation relativement simple que celle de compléter un automate déterministe.

```
Pour chacun des états
{
    Et pour chacune de ses transitions
    {
        Si la transition ne point vers aucun état d'arrivée, on la fait pointer vers la poubelle
    }
}
Si on a fait pointer un des états vers la poubelle, on ajoute l'état poubelle et on met son nombre de
transitions à jour
```

Reconnaître un mot

Encore une fois, on est ici sur une manipulation simple, notamment parce que l'on travaille sur un automate déterministe complet. Une fois le mot saisi par l'utilisateur, le programme va parcourir les transitions au fur et à mesure des lettres pour vérifier si, une fois au bout du mot, l'état est terminal.

```
L'utilisateur saisit un mot
Si c'est le mot vide, on vérifie si l'état initial est également terminal
{
    Si oui, le mot vide est reconnu
}
Sinon
{
    Pour chaque caractère du mot
    {
        On lit la lettre de chaque transition de l'état actuel (l'état initial pour le 1er caractère)
        {
            Si la lettre est similaire au symbole, on récupère l'état d'arrivée de la transition et
        }
    }
}
Si on arrive sur un état terminal, le mot est reconnu
```

TAI de Mathématiques pour l'Informatique

Conclusion

Dans ce projet, on aura eu l'occasion d'appliquer en informatique toute la théorie des mathématiques pour l'informatique. Cependant, même si nous avons commencé tôt à préparer nos algorithmes, nous nous sommes mis au code beaucoup trop tard pour pouvoir faire correctement tout ce qu'il nous a été demandé.