Les automates à états finis

Problématique

Problème

Le problème qui se pose est de pouvoir reconnaître si un mot donné appartient à un langage donné. Un **reconnaisseur** pour un langage est un programme qui prend en entrée une chaîne x et répond oui si x est une phrase (un mot) du langage et non sinon.

Théorème

***Les automates à états finis** (A.E.F.) sont des reconnaisseurs pour les langages réguliers.*

Automates à états finis

Définitions

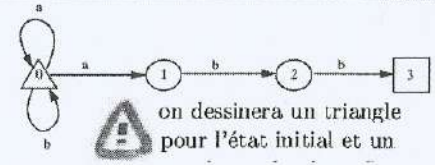
Un **automate à états finis (AEF)** *est défini par*

- *un ensemble* **fini** E *d'états*
- *un état* $e_0 \in E$ *distingué comme étant l'***état initial**
- *un ensemble fini* T *(*T *inclus dans* E*) d'états distingués comme* **états finaux** *(ou* **états terminaux***)*
- *un alphabet* Σ *des symboles d'entrée*
- *une* **fonction de transitions** Δ *qui à tout couple formé d'un état et d'un symbole de* Σ *fait correspondre un ensemble (éventuellement vide) d'états :* $\Delta(e_i, a) = \{e_{i_1}, \ldots, e_{i_n}\}$

Exemple

$\Sigma = \{a, b\}$, $E = \{0, 1, 2, 3\}$, $e_0 = 0$, $T = \{3\}$
$\Delta(0, a) = \{0, 1\}$, $\Delta(0, b) = \{0\}$, $\Delta(1, b) = \{2\}$, $\Delta(2, b) = \{3\}$ (et $\Delta(e, l) = \emptyset$ sinon)

Représentation graphique



on dessinera un triangle pour l'état initial et un

Représentation par une table de transition

état	a	b
0	0,1	0
1	-	2
2	-	3
3	-	-

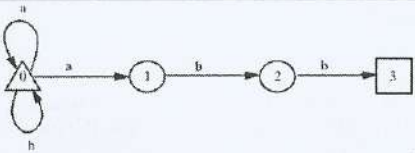
$e_0 = 0$ et $T = \{3\}$

Automates à états finis

Définition

Le **langage reconnu par un automate** *est l'ensemble des chaînes qui permettent de passer de l'état initial à un état terminal.*

Exemple



L'automate de l'exemple précédent accepte le langage régulier (l'expression régulière) .....

Remarque

Un automate peut très facilement être simulé par un algorithme (et donc on peut écrire un programme simulant un AEF). C'est encore plus facile si l'automate est **déterministe**, c'est à dire lorsqu'il n'y a pas à choisir entre 2 transitions). Ce que signifie donc le théorème 3.1 c'est que l'on peut écrire un programme reconnaissant tout mot (toute phrase) de tout langage régulier. Ainsi, si l'on veut faire l'analyse lexicale d'un langage régulier, il suffit d'écrire un programme simulant l'automate qui lui est associé.

Construction d'un AFN à partir d'une E.R.

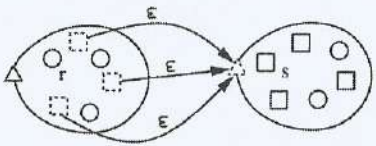
Définitions

On appelle **ε-transition**, *une transition par le symbole ε entre deux états.*
Pour une expression régulière s, on note $A(s)$ un automate reconnaissant cette expression.

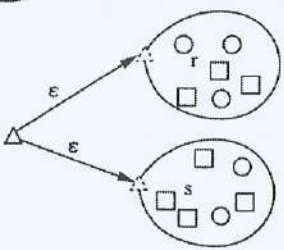
- automate acceptant la chaîne vide 

- automate acceptant la lettre a 

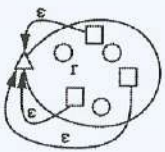
- automate acceptant $(r)(s)$
  1. mettre une ε-transition de chaque état terminal de $A(r)$ vers l'état initial de $A(s)$
  2. les états terminaux de $A(r)$ ne sont plus terminaux
  3. le nouvel état initial est celui de $A(r)$
  4. (l'ancien état initial de $A(s)$ n'est plus état initial)



- automate reconnaissant $r|s$
  1. créer un nouvel état initial q
  2. mettre une ε-transition de q vers les états initiaux de $A(r)$ et $A(s)$
  3. (les états initiaux de $A(r)$ et $A(s)$ ne sont plus états initiaux)



- automate reconnaissant r^+
  mettre des ε-transition de chaque état terminal de $A(r)$ vers son état initial



Construction d'un AFN à partir d'une E.R.

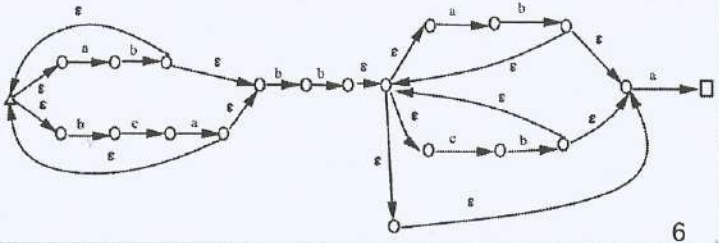
Exemple1

Donner un AFN des expressions régulières suivantes :

- a*
- a+
- a|b
- a*|b
- a*|b+
- a+|b+
- (a|b)+
- (a|b)*

Exemple1

Donner une ER que reconnaît l'automate suivant :



Automates finis déterministes (AFD)

Définitions

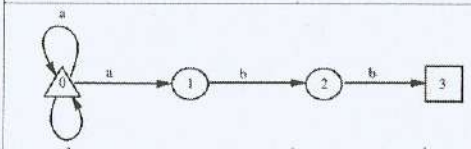
On appelle ε-**transition**, *une transition par le symbole* ε *entre deux états.*
Un automate fini est dit **déterministe** *lorsqu'il ne possède pas de* ε*-transition et lorsque pour chaque état* e *et pour chaque symbole* a*, il y a* **au plus un** *arc étiqueté* a *qui quitte* e

Remarques

L'automate donné en exemple précédemment qui reconnait $(a|b)^*abb$ n'est pas déterministe, puisque de l'état 0, avec la lettre a, on peut aller soit dans l'état 0 soit dans l'état 1.

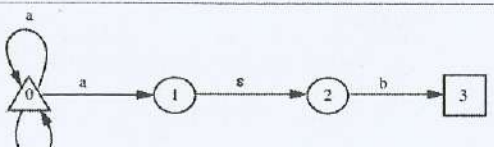
Les AFD sont plus faciles à simuler (pas de choix dans les transitions, donc jamais de retours en arrière à faire). Il existe des algorithmes permettant de **déterminiser** un automate non déterministe (c'est à dire de construire un AFD qui reconnait le même langage que l'AFN[1] donné). L'AFD obtenu comporte en général plus d'états que l'AFN, donc le programme le simulant occupe plus de mémoire.

Exemple1



AF-ND car à partir de l'état 0 il y a deux arcs étiquetés a

Exemple2



il existe une ε-transition

Déterminisation d'un AFN ne contenant pas de Ɛ_transition

Algorithme

1. Partir de l'état initial : $E = \{e_0\}$
2. Construire $E^{(1)}$ l'ensemble des états obtenus à partir de E par la transition a : $E^{(1)} = \Delta(E, a)$
3. Recommencer 2 pour toutes les transitions possibles et pour chaque nouvel ensemble d'état $E^{(i)}$
4. Tous les ensemble d'états $E^{(i)}$ contenant au moins un état terminal deviennent terminaux
5. Renuméroter alors les ensemble d'états en tant que simples états .

Exemple

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

$e_0 = 0$ et $T = \{2, 3\}$

Déterminisation d'un AFN ne contenant pas de Ɛ_transition

Solution

Définition

On appelle ε-**fermeture** *de l'ensemble d'états* $T = \{e_1, \ldots, e_n\}$ *l'ensemble des états accessibles depuis un état* e_i *de* T *par des* ε*-transitions*

ε-fermeture$(\{e_1, \ldots, e_n\} = \{e_1, \ldots, e_n\} \cup \{e \text{ tq } \exists e_i \text{ avec } i = 1, 2, \ldots, n \text{ tq } e_i \xrightarrow{\varepsilon}\xrightarrow{\varepsilon}\xrightarrow{\varepsilon} \ldots \xrightarrow{\varepsilon} e\}$

Calcul de Ɛ transition

Mettre tous les états de T dans une pile P
Initialiser ε-fermeture(T) à T
Tant que P est non vide faire
  Soit p l'état en sommet de P
  dépiler P
  Pour chaque état e tel qu'il y a une ε-transition entre p et e faire
    Si e n'est pas déja dans ε-fermeture(T)
      ajouter e à ε-fermeture(T)
      empiler e dans P
    finsi
  finpour
fin tantque

Exemple

état	a	b	c	ε
0	2	-	0	1
1	3	4	-	-
2	-	-	1,4	0
3	-	1	-	-
4	-	-	3	2

$e_0 = 0$ et $T = \{4\}$

Déterminisation d'un AFN contenant de Ɛ_transition

Algorithme

1. Partir de l'ϵ-fermeture de l'état initial
2. Rajouter dans la table de transition toutes les ϵ-fermetures des nouveaux "états" produits, avec leurs transitions
3. Recommencer 2 jusqu'à ce qu'il n'y ait plus de nouvel "état"
4. Tous les "états" contenant au moins un état terminal deviennent terminaux
5. Renuméroter alors les états.

Exemple

état	a	b	c	ε
0	2	-	0	1
1	3	4	-	-
2	-	-	1,4	0
3	-	1	-	-
4	-	-	3	2

$e_0 = 0$ et $T = \{4\}$

Minimisation d'un AFN

Objectif

obtenir un automate ayant le minimum d'états possible.
on définit des classes d'équivalence d'états par raffinements successifs. Chaque classe d'équivalence obtenue forme un seul même état du nouvel automate.

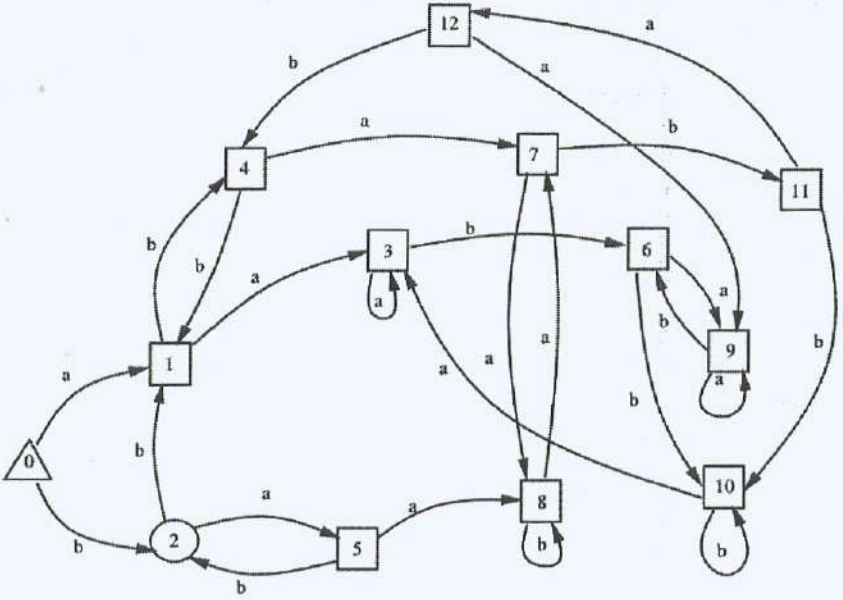
Algorithme

1 - Faire deux classes : A contenant les états terminaux et B contenant les états non terminaux.
2 - S'il existe un symbole a et deux états e_1 et e_2 d'une même classe tels que $\Delta(e_1, a)$ et $\Delta(e_2, a)$ n'appartiennent pas à la même classe, alors créer une nouvelle classe et séparer e_1 et e_2. On laisse dans la même classe tous les états qui donnent un état d'arrivée dans la même classe.
3 - Recommencer 2 jusqu'à ce qu'il n'y ait plus de classes à séparer.
4 - Chaque classe restante forme un état du nouvel automate

Exemple (voir diapo 13)

Minimisation d'un AFN

Exemple



Calcul d'une ER à partir d'un AEF

Définition

On appelle L_i le langage que reconnaîtrait l'automate si e_i était son état initial. On peut alors écrire un système d'équations liant tous les L_i :

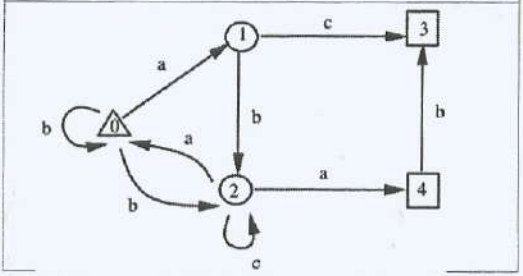
- chaque transition $\Delta(e_i, a) = e_j$ permet d'écrire l'équation $L_i = aL_j$
- pour chaque $e_i \in T$ on a l'équation $L_i = \varepsilon$
- les équations $L_i = \alpha$ et $L_i = \beta$ se regroupent en $L_i = \alpha|\beta$

On résoud ensuite le système (on cherche à calculer L_0) en remarquant juste que

Propriété

si $L = \alpha L|\beta$ *alors* $L = \alpha^*\beta$

Exemple (voir Slide 13)



Déterminisation d'un AFN (cas général)

Définition

On appelle ε-**fermeture** *de l'ensemble d'états* $T = \{e_1, \ldots, e_n\}$ *l'ensemble des états accessibles depuis un état* e_i *de* T *par des* ε*-transitions*

ε-fermeture$(\{e_1, \ldots, e_n\} = \{e_1, \ldots, e_n\} \cup \{e \text{ tq } \exists e_i \text{ avec } i = 1, 2, \ldots, n \text{ tq } e_i \xrightarrow{\varepsilon}\xrightarrow{\varepsilon}\xrightarrow{\varepsilon} \ldots \xrightarrow{\varepsilon} e\}$

Exemple