CH.1 Automates finis

- 1.1 Les automates finis déterministes
- 1.2 Les automates finis non déterministes
- 1.3 Les automates avec ε-transitions
- 1.4 Les expressions régulières
- 1.5 L'équivalence des modèles

1.1 Les automates finis déterministes

M = (Q, Σ, δ, q_0, F) est un AFD

Q ensemble fini d'états
Σ alphabet fini
q_0 état initial
F ensemble des états terminaux
δ fonction de transition Q x Σ → Q

Fonction de transition étendue aux mots :

1– δ'(q, ε) = q
2– δ'(q, wa) = δ(δ'(q, w), a)

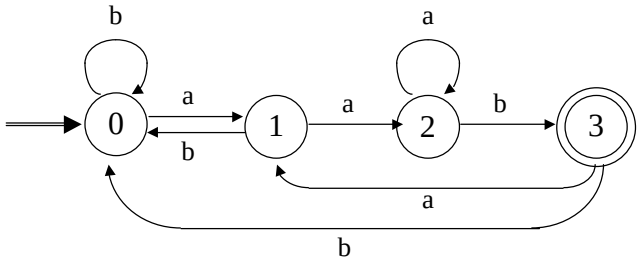
Les deux coïncident sur les lettres

Mot x **accepté ou reconnu** si $\delta'(q_0, x)$ est terminal.
Reconnaissance de x en temps O(n).

Langage accepté L(M) = ensemble des mots acceptés.

Langage reconnaissable = langage accepté par un AFD

Exemple L(M) = mots se terminant par aab



1.2 Les automates finis non déterministes

A priori plus général. Utilité théorique, mais aussi
plus facile à construire. Mais plus coûteux à utiliser.

$M = (Q, \Sigma, \delta, q_0, F)$ est un AFN

Seule différence :
δ fonction de transition $Q \times \Sigma \rightarrow 2^Q$

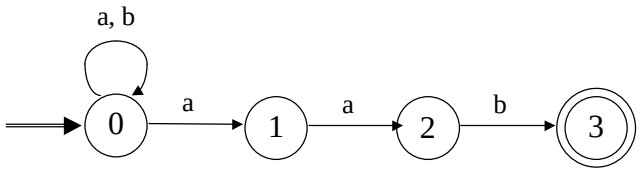
Fonction de transition étendue aux mots :

1– $\delta'(q, \varepsilon) = \{q\}$
2– $\delta'(q, wa) = \{p : \exists r \in \delta'(q, w) \text{ et } p \in \delta(r, a)\}$

Les deux coïncident sur les lettres

3– $\delta'(P, w) = \cup\, \delta'(q, w)$

Exemple L(N) = mots se terminant par aab



Mot x **accepté ou reconnu** si $\delta'(q_0, x)$ contient un état terminal = il existe un chemin de q_0 vers un état terminal dont la suite des étiquettes vaut x.
Reconnaissance de x en temps $O(2^n)$.

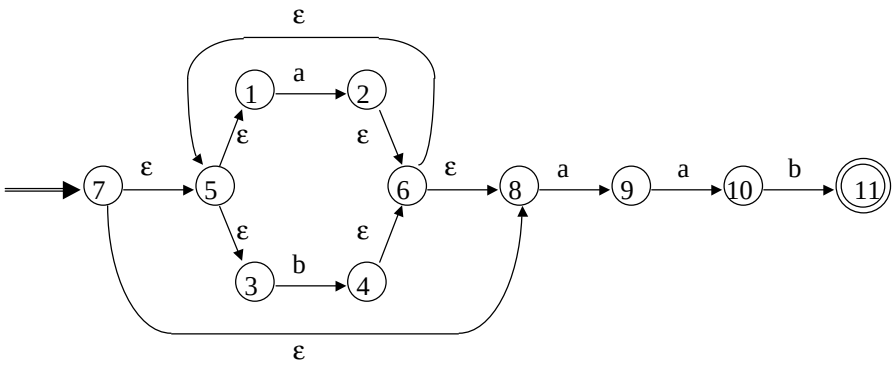
Langage accepté L(N) = ensemble des mots acceptés.

Théorème : Si L est accepté par un AFN, alors il est accepté par un AFD.

1.3 Les automates avec e**-transitions**

Comme les AFN, mais il peut aussi y avoir des transitions sur le mot vide ε.

Exemple L(N) = mots se terminant par aab



Mot x **accepté ou reconnu** s'il existe un chemin de q_0 vers un état terminal dont la suite des étiquettes vaut x.
Le mot vide est neutre pour la concaténation, donc tous les chemins ne sont pas de longueur n.
Reconnaissance de x en temps $O(2^n)$.

Langage accepté L(N) = ensemble des mots acceptés.

Notion importante : la ε**-clôture**.

clot(q) = {p : il existe un chemin étiqueté ε de q à p}

La détermination de la ε-clôture se fait par un algorithme d'exploration du graphe obtenu en ne conservant que les transitions vides.

Théorème : Si L est accepté par un AFN avec ε-transitions, alors il est accepté par un AFN sans ε-transitions.

Corollaire : Les langages acceptés par les AFD, les AFN sans ε-transitions et les AFN avec ε-transitions coïncident.

1.4 Les expressions régulières

Elles permettent de spécifier a priori un langage, alors que les automates permettent de tester l'appartenance d'un mot à un langage déjà spécifié.

Opérations régulières sur les langages :

- réunion ensembliste $\cup$ ou +
- produit de concaténation
- fermeture transitive *

Les expressions régulières sont des chaînes de caractères contituées de lettres et de ε, de parenthèses et de symboles opératoires +, . ou <rien>, *. Cette chaîne peut être vide, notée $\varnothing$.

Expressions régulières :

i) L'expression régulière ε représente{ε} ;
ii) Si a est une lettre, alors c'est une expression régulière qui représente{a} ;
iii) Si r et s sont des expressions régulières qui représentent L(r) et L(s), alors :
- r + s représente L(r) $\cup$ L(s) (ou r | s)
- rs représente L(r)L(s)
- r* représente L(r)*.

Langage régulier = langage représenté par expression régulière.

Tous les langages finis sont réguliers.

Exemples :
(a + b)* représente tous les mots sur {a, b}
a*(ba*)* représente le même langage
(a + b)*aab représente les mots se terminant par aab.

Théorème : Tout langage régulier est reconnaissable par un AFN avec ε-transitions.

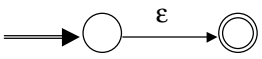
Corollaire : Tout langage régulier est reconnaissable.

Démonstration : Algorithme de Thompson

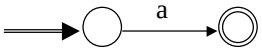
Décrit l'assemblage des automates correspondant aux opérations sur les expressions régulières.

Exemple : on retrouve l'AFN avec ε-transitions de la p.6 à partir de r = (a + b)*aab.

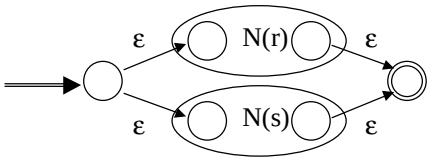
• pour ε : N(ε)



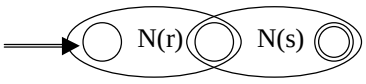
• pour a : N(a)



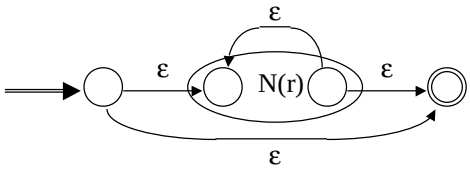
• pour r + s : N(r + s)



• pour rs : N(rs)



• pour r* : N(r*)



1.5 L'équivalence des modèles

Démonstrations des théorèmes des paragraphes 2 et 3.

Théorème : Si L est accepté par un AFN, alors il est accepté par un AFD.

Démonstration :

Donnée : un AFN sans ε-transition N.

Résultat : un AFD M acceptant le même langage.

Les états de M sont des ensembles d'états de N.

L'état initial de M est $[q_0]$.

Si P est un ensemble d'états de N, on définit

$\delta'(P, a) = \cup_{p \in P}\{q : q \in \delta(p, a)\}$.

Ceci définit récursivement un l'ensemble des états de M.

Un état P de M est terminal s'il contient un état terminal de N.

Remarques :

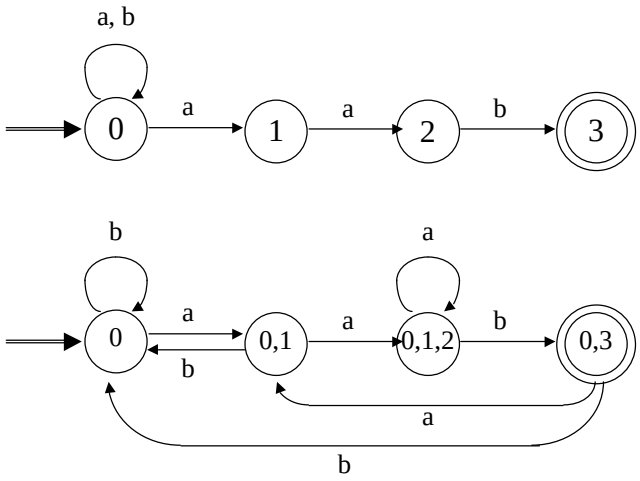
Si $P = \varnothing$, c'est un état "poubelle" de M.

Si $\delta'(P, a) = Q$, alors tout état de N qui apparaît dans Q peut être atteint à partir d'un état qui apparaît dans P par une flèche étiquetée a.

Cette dernière remarque permet d'établir que $\delta'([q_0], w)$ est final dans M si et seulement si il existe dans N un chemin étiqueté w de q_0 à un état terminal de N.

Ceci établit le théorème.

Exemple : automate du paragraphe 2.



On retrouve l'AFD du paragraphe 1.

Théorème : Si L est accepté par un AFN avec ε-transitions, alors il est accepté par un AFD.

Démonstration :

Donnée : un AFN avec ε-transition N.

Résultat : un AFD M acceptant le même langage.

Les états de M sont des ensembles d'états de N.

Il faut faire attention aux ε-clôtures.

L'état initial de M est Clot($[q_0]$).

Si P est un ensemble d'états de N, on définit

$\delta'(P, a) = \cup_{p \in P} \text{Clot}(\{q : q \in \delta(p, a)\})$.

Ceci définit récursivement un l'ensemble des états de M.

Un état P de M est terminal s'il contient un état terminal de N.

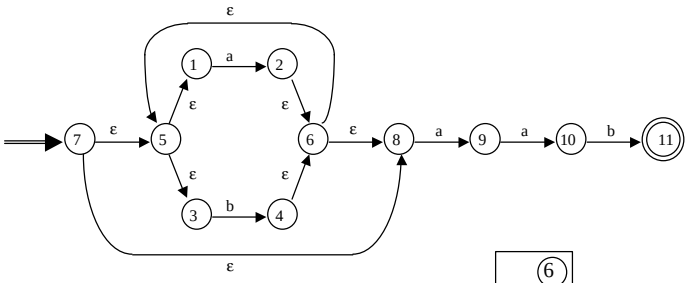
Remarques :

Si δ'(P, a) = Q, alors tout état de N qui apparaît dans Q peut être atteint à partir d'un état qui apparaît dans P par une flèche étiquetée a ou par un chemin étiqueté aεεεε…

Cette dernière remarque permet d'établir que δ'(Clot([q_0]), w) est final dans M si et seulement si il existe dans N un chemin étiqueté w de q_0 à un état terminal de N.

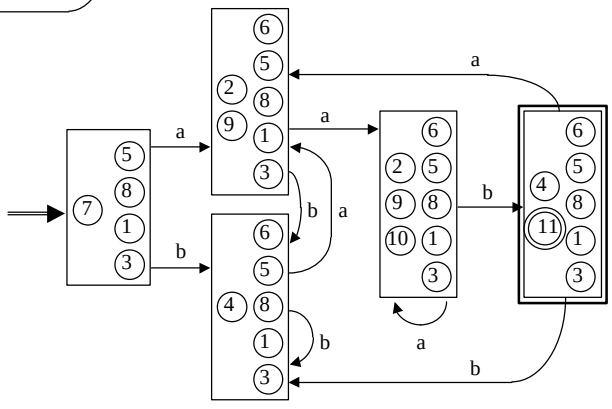
Ceci établit le théorème.

Exemple : automate du paragraphe 3, construit par l'algorithme de Thompson à partir de l'expression (a + b)*aab.



Les états sont constitués des états de N atteints et de leur ε-clôture.

L'AFD ainsi obtenu est un peu plus compliqué que précédemment.



On a ainsi établi que pour tout langage représenté par une expression régulière on peut trouver un AFD qui le reconnaît.

On va maintenant montrer la réciproque. On aura ainsi établi le

Théorème de Kleene : Les langages réguliers coïncident avec les langages reconnaissables.
Démonstration :
Il reste à montrer que tout langage reconnaissable par un automate fini peut être décrit par une expression régulière.
Ceci est établi par un algorithme.
Donnée : Un automate fini M.
Résultat : Une expression régulière décrivant L(M).

L'algorithme ne suppose pas que l'automate soit déterministe.

On définit un automate généralisé comme un graphe avec un seul état initial α et un seul état terminal ω, dont les flèches sont étiquetées par des expressions régulières. Un mot w est reconnu par un automate généralisé s'il existe un chemin allant de α à ω tel que w appartient au langage décrit par le produit des expressions régulières apparaissant comme étiquettes de ce chemin. Le langage reconnu est l'ensemble des mots reconnus. Deux automates sont équivalents lorsqu'ils reconnaissent le même langage.

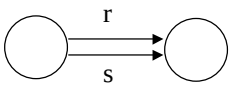
On peut facilement transformer un automate M ordinaire en automate généralisé : il suffit d'ajouter les états α et ω et des ε-transitions de α vers l'état inital de M et des états terminaux de M vers ω.

Le langage reconnu est bien le même selon les deux définitions.

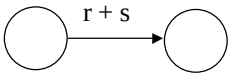
Etant donné un automate généralisé, on va trouver un automate équivalent ayant moins de flèches ou d'états, en appliquant les transformations ci-dessous.

Réduction des flèches :

Avant

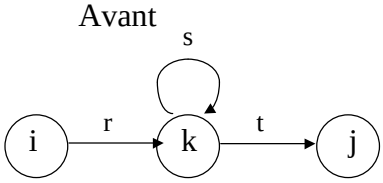


Après

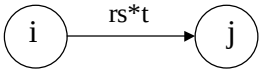


Réduction des états : on enlève k ; pour chaque couple d'un prédécesseur de k et d'un successeur de k, faire :

Avant



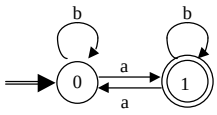
Après



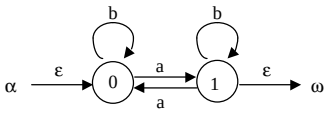
Le langage reconnu n'est pas modifié. En partant d'un automate ordinaire, les états ajoutés α et ω restent toujours respectivement sans prédécesseur ni sans successeur.
A la fin on obtient donc un automate avec deux états α et ω et une seule flèche. Cette flèche est constituée d'une expression régulière. Comme l'automate est équivalent à M, c'est une expression régulière décrivant le langage L(M). Ceci termine la démonstration du théorème de Kleene.

Remarques :
L'expression régulière obtenue dépend du choix des états à réduire.
Les expressions obtenues à partir d'AFN sont souvent plus simples surtout s'il y a moins de flèches.

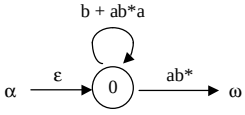
Exemple : L(M) = nombre impair de a



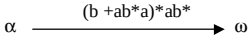
Adjonction de α et ω



Suppression de l'état 1



Suppression de l'état 0



Donc L(M) peut être représenté par (b + ab*a)*ab*.