Chapitre 1

Automates à États Finis

1.1 Définitions

Déf. 1 (Automate fini déterministe - AFD)

Un automate à nombre fini d'états (automate fini) déterministe $\mathcal{A}$ est défini par :

$$\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$$

Q est un ensemble fini d'états
Σ est un vocabulaire (ou alphabet)
q_0 est un élément de Q, appelé état initial
F est un sous-ensemble de Q, dont les éléments sont appelés états terminaux
δ est une **fonction** de $Q \times \Sigma$ dans Q. On écrit $\delta(q, a) = r$.

Déf. 2 (Reconnaissance)

Un mot $a_1 a_2 ... a_n$ est **reconnu** par l'automate si et seulement si il existe une suite $k_0, k_1, ..., k_n$ d'éléments de Q (ensemble d'états) telle que

$$k_0 = q_0$$
$$k_n \in F$$
$$\forall i \in [1, n], \quad \delta(k_{i-1}, a_i) = k_i$$

Déf. 3 (AFD complet)

Un automate à nombre fini d'états déterministe complet $\mathcal{A}$ est défini par :

$$\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$$

Q est un ensemble fini d'états
Σ est un vocabulaire (ou alphabet)
q_0 est un élément de Q, appelé état initial
F est un sous-ensemble de Q, dont les éléments sont appelés états terminaux
δ est une **fonction** de $Q \times \Sigma$ dans Q, qui vérifie la propriété :

$$\forall (q, a) \in Q \times \Sigma, \ \exists q' \in Q \text{ t.q. } \delta(q, a) = q'$$

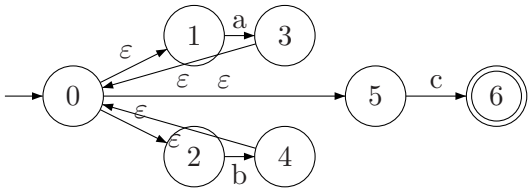
Déf. 4 (Automate fini non déterministe - AFnD)

Un automate à nombre fini d'états (automate fini) non déterministe $\mathcal{A}$ est défini par :

$$\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$$

Q est un ensemble fini d'états
Σ est un vocabulaire (ou alphabet)
q_0 est un élément de Q, appelé état initial
F est un sous-ensemble de Q, dont les éléments sont appelés états terminaux
δ est une **fonction** de $Q \times \Sigma \cup \{\varepsilon\}$ dans 2^Q.

1.1.1 Élimination des ε-transitions



Étant donné un ε-automate $\langle X, Q, I, F, \delta \rangle$, on peut construire un automate non déterministe qui reconnaît le même langage.

Pour cela, on définit l'application ε^+ de la manière suivante :

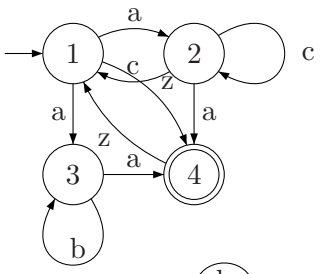
1. Si $q_j \in \delta(q_i, \varepsilon)$ alors $q_j \in \varepsilon^+(q_i)$
2. Si $q_j \in \varepsilon^+(q_i)$ et $q_k \in \delta(q_j, \varepsilon)$ alors $q_k \in \varepsilon^+(q_i)$

On construit l'automate non déterministe $\langle X, Q, I, F', \delta' \rangle$ comme suit :

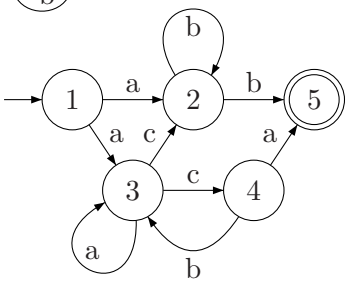
```
Début
    F' := F
    pour tous les q_i ∈ Q faire
        pour tous les x ∈ X faire
            δ'(q_i, x) := δ(q_i, x)
    pour tous les q_i tels que ε+(q_i) ≠ ∅ faire
        pour tous les q_j ∈ ε+(q_i) faire
            pour tous les x et q_r tels que q_r ∈ δ(q_j, x) faire
                δ'(q_i, x) := δ'(q_i, x) ∪ {q_r}
            si q_j ∈ F alors F' := F' ∪ {q_i}
Fin
```

.1 Exercices : Automates

1. Soit $\mathcal{A} = \{a, b, c\}$. Donner un automate déterministe complet qui reconnaît chacun des langages suivants :
   (a) L'ensemble des mots de longueur paire
   (b) L'ensemble des mots où le nombre d'occurrences de b est divisible par 3
   (c) L'ensemble des mots se terminant par b
   (d) L'ensemble des mots ne se terminant pas par b
   (e) L'ensemble des mots non-vides ne se terminant pas par b
   (f) L'ensemble des mots contenant au moins un b
   (g) L'ensemble des mots contenant au plus un b
   (h) L'ensemble des mots contenant exactement un b
   (i) L'ensemble des mots ne contenant aucun b
   (j) L'ensemble des mots contenant au moins un a et dont le première occurrence de a n'est pas suivie par un c
   (k) L'ensemble des mots comportant au moins 3 lettres et dont la troisième lettre à partir de la fin est un a ou un c
2. Pour chacun des langages rationnels suivants sur l'alphabet $\{0, 1\}$, construire un automate non déterministe le reconnaissant. Déterminiser les automates obtenus, soit *de visu*, soit en utilisant l'algorithme de déterminisation.
   (a) Ensemble des mots dont l'avant-dernière lettre est 0
   (b) Ensemble des mots contenant le facteur 001
   (c) Ensemble des mots qui débutent et terminent par la même lettre
3. Déterminiser l'automate suivant. On donnera la table de transition ainsi que la représentation graphique, en précisant les classes qui forment les nouveaux états. Trouver une expression rationnelle caractérisant ce langage.



4. Déterminiser l'automate suivant (on demande juste un automate déterministe reconnaissant le même langage). Vérifier que l'automate déterminisé reconnaît le même langage en testant 3 mots différents.



5. Proposer un automate minimal (en nombre d'états) qui reconnaisse le langage décrit par l'expression rationnelle $a * (c(ab|ba*)|cab|cb)$. On déduira de l'automate une expression rationnelle plus simple. On ne demande pas nécessairement d'appliquer les algorithmes vus en cours.
6. Dans les automates proposés aux exercices 4 et 3, remplacer le symbole z par c par une transition vide (ε-transition), et déterminiser l'automate ainsi obtenu.
7. Proposer un automate et une expression rationnelle pour le langage de tous les mots de $\{a, b, c\}*$ donc cac est un sous-mot.
8. Soit l'alphabet $X = \{a, b, c\}$. Proposer un automate fini déterministe qui reconnaît

le langage $L_2 = \{u \in X^* / \nexists v, w \in X^* | u = vbacaw\}$

9. Soit l'expression rationnelle $(a|bc)^*z(z|ba|ca^*)$.
   (a) Proposer un automate reconnaissant le même langage
   (b) Éliminer les ε-transitions
   (c) Déterminiser l'automate résultant
   (d) Minimiser l'automate résultant
10. Proposer un automate qui reconnaît $L_1 \cap L_2$, où L_i est le langage reconnu par $\mathcal{A}_i$.

$\mathcal{A}_1$	a	b	c
$\rightarrow 1$	2		4
2		3	
3	2		4
$\leftarrow 4$	2	4	

$\mathcal{A}_2$	a	b	c
$\rightarrow$ A	A	B	D
B		C	B
C	A		D
$\leftarrow$ D	C		

11. Donnez une structure de données (dans un langage orienté objet) propre à représenter et manipuler un automate à états fini. Vous décrirez notamment les attributs et méthodes de chacune des classes que vous définirez. Les structures que vous créez devront au minimum :
    (a) représenter tous les attributs formels d'un automate et permettre de les créer et les modifier
    (b) partager le même alphabet, celui de l'ensemble des caractères du jeu de caractères utilisé par le programme
    (c) offrir les méthodes permettant la reconnaissance d'un mot (et donc toutes les autres méthodes que cela présuppose)

    On ne vous demande pas de code effectif à ce stade, mais juste le détail de l'architecture que vous utiliserez dans votre projet.
12. Ecrire l'algorithme de parcours d'un automate déterministe mais non complet. On notera `q0` l'état initial, `x[1]...x[`n`]` la chaîne d'entrée, et on supposera que l'on dispose d'une fonction booléenne `existe_t`, à deux arguments `q` (état) et `c` (caractère) qui renvoie *true* si la fonction δ a une valeur pour (`q`,`c`). On dispose aussi d'une fonction `delta`, avec les mêmes arguments, qui retourne un état pour les valeurs pour laquelle elle est définie.
13. Proposer un algorithme de déterminisation d'un automate (sans ε-transition, mais pas nécessairement complet). On fera les hypothèses suivantes :
    - Le type `état` est défini
    - On dispose des types et primitives nécessaires pour manipuler (1) des listes, (2) des ensembles[1].
    - Un automate est déterminé par les fonctions suivantes :
      - `terminal(état q)` : renvoie `vrai` si `q` est un état terminal
      - `initial()` : renvoie l'état initial
      - `delta(état q, lettre x)` : renvoie un ensemble d'états

[1] En particulier, on suppose que l'on dispose d'une instruction qui permet de parcourir un ensemble : quelque chose comme `Pour tout x appartenant à X faire`...