IN 101 - TD 06
22 octobre 2010

Énoncé

ENSTA ParisTech

Exercice 1 : Automates finis

Définition et représentation d'automates finis. Un automate fini déterministe est une machine (virtuelle) pouvant se trouver dans un état parmi un nombre fini. Cette machine, initialement dans un état donné, reçoit un message sous la forme d'une suite de caractères : à chaque caractère reçu, l'automate change d'état en suivant une règle appelée *fonction de transition*.

Plus formellement, un automate fini déterministe est la donnée :
- d'un alphabet fini Σ, *i.e.* d'un ensemble de caractères ;
- d'un ensemble fini d'états Q ;
- d'une fonction de transition $\delta : Q \times \Sigma \to Q$;
- d'un état initial p_0 ;
- d'un ensemble d'états finaux F.

On dessine habituellement un automate comme un graphe dont chaque sommet représente un état et dont chaque arc, étiqueté par un symbole de l'alphabet, représente une transition. L'état initial est désigné par une flèche sans origine, et les états finaux sont doublement cerclés (ou cerclés en gras).

Par exemple, le diagramme de la FIGURE 1 représente l'automate :
- $\Sigma = \{a, b\}$
- $Q = \{1, 2, 3, 4\}$
- $\delta(1, a) = 2$, $\delta(2, a) = 4$, $\delta(3, a) = 3$, $\delta(4, a) = 4$, $\delta(1, b) = 4$, $\delta(2, b) = 3$, $\delta(3, b) = 2$, $\delta(4, b) = 4$
- $p_0 = 1$
- $F = \{3\}$

On remarquera que l'état 4 est très particulier car une fois atteint, on ne peut plus le quitter. De plus, ce n'est pas un état final. Un tel état est appelé « rebut », et n'est pas représenté afin de simplifier les diagrammes. On obtient donc finalement le graphe de la figure 2 avec la convention que lorsqu'une transition n'est pas indiquée, elle mène à l'état rebut non représenté.

Langage reconnu par un automate fini. Un automate fini permet de définir des langages, c'est-à-dire des ensembles de mots formés de caractères issus de l'alphabet Σ. La règle du jeu est la

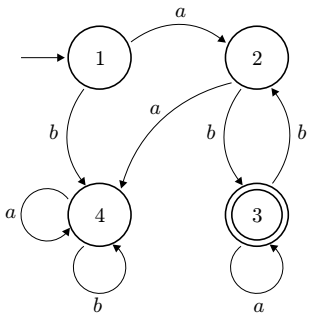


FIGURE 1 – Exemple de diagramme représentant un automate.

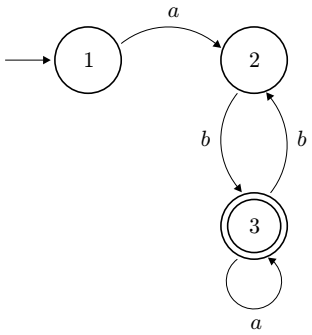


FIGURE 2 – Diagramme simplifié, sans état rebut.

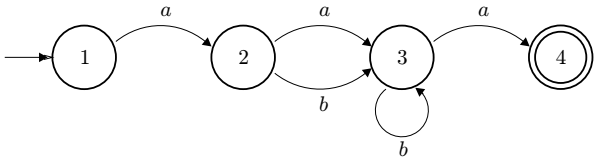


FIGURE 3 – Automate reconnaissant $L = a(a|b)b^*a$.

suivante : afin de savoir si un mot est *reconnu* par un automate, on initialise ce dernier dans l'état p_0, puis, en utilisant la fonction de transition, on fait évoluer l'état de l'automate en lisant les caractères du mot un par un. Lorsque tous les caractères ont été lus, l'automate se trouve dans un état p ; si p appartient à l'ensemble des états finaux, le mot est déclaré reconnu et appartient au langage défini par l'automate. Sinon, le mot n'est pas reconnu et n'appartient pas au langage. Ainsi, l'automate présenté en FIGURE 2 reconnaît les mots *ab*, *aba*, *abaa*, *abbb*, *abaabbbbaaa*..., mais aucun des mots *baa*, *abb*, *abba*...

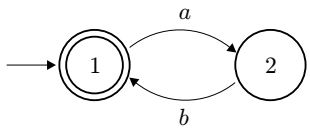
Afin de décrire le langage, c'est-à-dire l'ensemble (généralement infini) des mots reconnus par un automate, on utilise les notations suivantes : si t_1 et t_2 sont deux ensembles de mots :

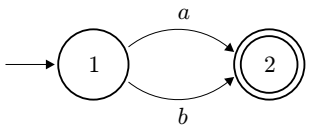
- $t_1|t_2$ désigne l'union des ensembles t_1 et t_2
- t_1t_2 désigne l'ensemble des mots formés par concaténation d'un mot de t_1 et d'un mot de t_2.
- ${t_1}^*$ désigne l'ensemble des mots obtenus par concaténation d'un nombre quelconque de mots de t_1 (éventuellement aucun).
- ${t_1}^+$ désigne l'ensemble des mots obtenus par concaténation d'un nombre non nul de mots de t_1.

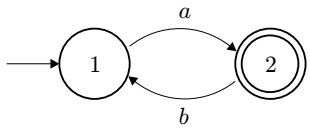
Enfin, on note ϵ le mot vide, dont le nombre de caractères est nul.

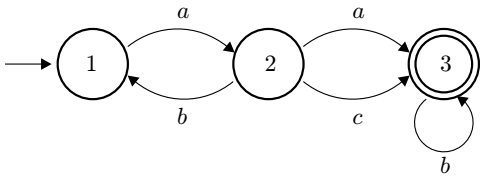
Exemple. Le langage $L = a(a|b)b^*a$ est formé de l'ensemble des mots sur l'alphabet $\{a, b\}$ commençant par un a, suivi d'un a ou d'un b, puis d'un nombre quelconque (éventuellement nul) de b, et se terminant par un a. Ainsi, le mot *aabbbba* appartient au langage L, car il se décompose en *a a bbbb a*. En revanche, le mot *ababbba* n'appartient pas au langage L. Le langage L est reconnu par l'automate représenté en FIGURE 3.

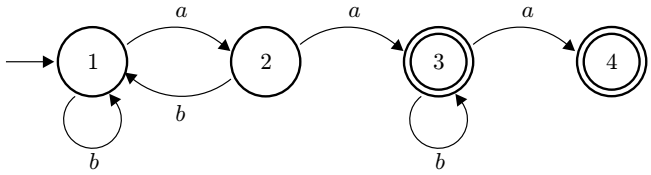
1.a] Quels sont les langages reconnus par les automates suivants ?

1. 

2. 

3. 

4. 

5. 

6. l'automate de la FIGURE 1

1.b] Construire des automates reconnaissant respectivement les langages suivants :

1. $(ba|a)^*$
2. $(ba|b)^*$
3. $(a|aa|aaa)$
4. $b^*(a|aa|aaa)b^*$

1.c] Construire des automates (avec pour alphabet les chiffres de 0 à 9) reconnaissant :

1. les nombres pairs,
2. les multiples de 3,
3. les multiples de 4.

1.d] On veut construire un automate reconnaissant un nombre complexe (à coefficients entiers). L'alphabet contient les caractères '+', '-', et 'i' et les chiffres de 0 à 9. On veut reconnaître les nombres de la forme '+3', '2i', '-4+5i', '3i-2', 'i+2'... Les nombres des formes suivantes ne doivent en revanche pas être reconnus : '-+3', 'i4+5', '2i+i', '5-3', '-'... Écrivez le langage correspondant à ces nombres complexes et construisez un automate déterministe reconnaissant ce langage. Pour simplifier, on reconnaitra aussi des entiers commençant par 0, par exemple, +05-0i sera reconnu.

Exercice 2 : Automates non-déterministes

Un automate fini non-déterministe est tel que partant d'un même état et pour un même symbole, plusieurs états peuvent être accessibles. Graphiquement, cela signifie qu'un sommet peut être origine de plusieurs flèches étiquetées par le même symbole. L'avantage de tels automates est que pour certains langages, il sont faciles à construire, alors que la conception d'un automate déterministe n'est généralement pas évidente.

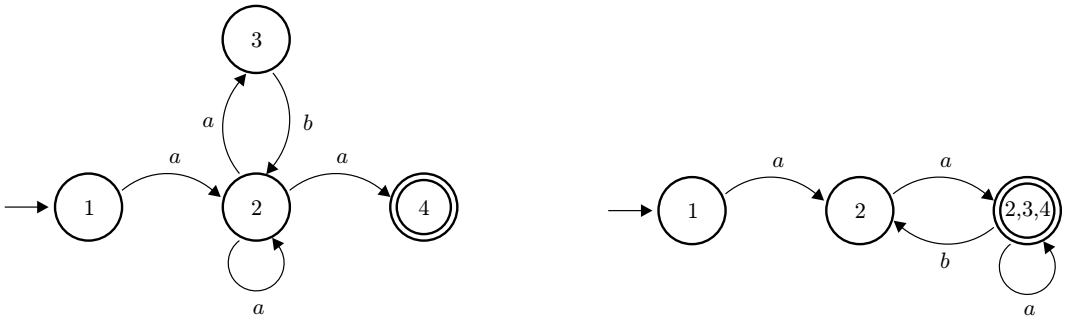
2.a] Construire un automate non déterministe reconnaissant le langage $L = (a|b)^*abb$.

Nous allons démontrer que tout langage reconnu par un automate non-déterministe peut également l'être par un automate déterministe. Cependant, le nombre d'états de l'automate déterministe peut être bien supérieur. Pour faire cette preuve, nous raisonnons de manière

constructive en proposant un algorithme permettant de construire un automate déterministe $\mathcal{D}$ à partir d'un automate non déterministe $\mathcal{A}$:

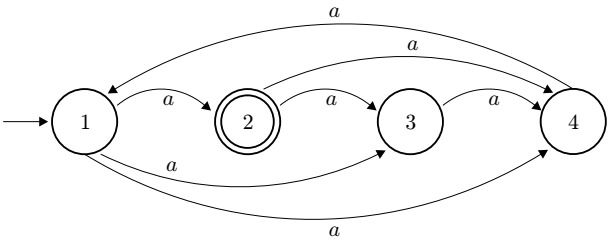
- l'alphabet Σ reste identique,
- si Q est l'ensemble des états de $\mathcal{A}$, l'ensemble des états de $\mathcal{D}$ est l'ensemble des parties de Q,
- si l'état initial de $\mathcal{A}$ est p_0, l'état initial de $\mathcal{D}$ est $\{p_0\}$,
- si l'ensemble des états finaux de $\mathcal{A}$ est F, l'ensemble des états finaux de $\mathcal{D}$ est l'ensemble des états f tels que $f \cap F \neq \emptyset$
- la fonction de transition δ de $\mathcal{D}$ fait passer l'automate déterministe de l'état $\{p_0, p_1, \ldots, p_k\}$ à l'état $\{q_0, q_1, \ldots, q_\ell\}$ en lisant le symbole s si et seulement si $\{q_0, q_1, \ldots, q_\ell\}$ est l'ensemble des états accessibles à partir d'au moins un des états p_0, p_1, …, ou p_k en lisant s pour l'automate non-déterministe $\mathcal{A}$.

Exemple. Ci-dessous, l'automate non-déterministe de gauche reconnaît le langage $L = a(a|ab)^*a$ et l'automate déterministe de droite a été obtenu par l'algorithme que nous venons de décrire.



2.b] Construire un automate déterministe reconnaissant le langage $L = (a|b)^*abb$.

2.c] Quel est le langage reconnu par l'automate suivant ? Quels mots ne sont pas reconnus par cet automate ?



Exercice 3 : *Pattern-matching* à base d'automates finis

Pour tout motif P de longueur m, on peut construire un automate M qui le reconnaisse et tel que :

- M ait $m + 1$ états : $\{0, 1, \ldots, m\}$,
- 0 soit l'état initial,
- m soit le seul état final.

Par exemple, avec l'alphabet $\Sigma = \{a, b\}$, l'automate de la FIGURE 4 dépourvu de ses flèches arrières reconnaît le mot *aabba* et seulement lui. L'automate complet accepte tout mot sur Σ, et atteint l'état 5 à chaque fois qu'il a reconnu une occurrence de *aabba* dans son entrée. On en déduit un algorithme très simple, de complexité optimale $\Theta(n)$ permettant de rechercher un motif dans un texte, une fois construit l'automate correspondant au motif :

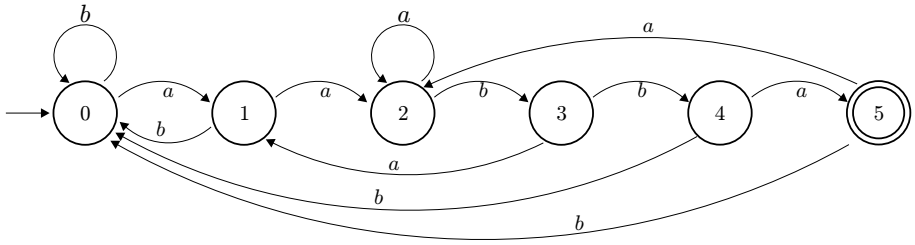


FIGURE 4 – Automate de recherche de motif

RECHERCHE-PAR-AUTOMATE(δ, T)
  $q \leftarrow 0$
  pour $s \leftarrow 1$ **à** n, **faire**
    $q \leftarrow \delta(q, T[s])$
    si $q = m$, **alors**
      classer $s - m$ comme un décalage valide

Calcul de l'automate de recherche. Le problème de la construction de l'automate de recherche n'est cependant pas simple. Il est fabriqué à partir d'un « squelette » linéaire de m transitions correspondant au motif P. La difficulté réside dans le calcul de la fonction de transition δ.

Soit P_q le préfixe de longueur q de P. La fonction de transition δ doit être telle que $\delta(q, x)$ soit égal à la longueur du plus long préfixe de P également suffixe de la concaténation de P_q et x.

Ainsi, dans l'exemple de la FIGURE 4, $\delta(3, a)$ est égal à la longueur du plus long préfixe de *aabba* également suffixe de *aaba*, c'est-à-dire égal à 1. De même, $\delta(5, a)$ est égal à la longueur du plus long préfixe de *aabba* également suffixe de *aabbaa*, c'est-à-dire égal à 2 (le tronçon commun étant *aa*).

Application au motif *aabba* :

$$\begin{array}{ll} \delta(0,a) = 1 & \delta(0,b) = 0 \\ \delta(1,a) = 2 & \delta(1,b) = 0 \\ \delta(2,a) = 2 & \delta(2,b) = 3 \\ \delta(3,a) = 1 & \delta(3,b) = 4 \\ \delta(4,a) = 5 & \delta(4,b) = 0 \\ \delta(5,a) = 2 & \delta(5,b) = 0 \end{array}$$

3.a] Construire l'automate de recherche du motif *abaabbaaa*.

3.b] On note Σ^* une suite quelconque d'éléments de Σ, Σ^+ une suite quelconque non vide d'éléments de Σ et Σ un élément quelconque de Σ. Dessinez les automates reconnaissants les langages Σ^*, Σ^+, et Σ.

3.c] On veut construire les automates permettant de reconnaître les motifs $ab\Sigma^*cd$ et $ab\Sigma^+cd$. Reconnaître l'un de ces motifs consiste à reconnaître un premier motif *ab* puis, dès que ce motif a été reconnu, de chercher à reconnaître le motif *cd*. En revanche, comme n'importe quel texte peut se trouver entre les deux motifs, une fois que le premier motif a été trouvé on ne revient plus jamais en arrière (avant le dernier état de l'automate reconnaissant le premier motif). Déduisez-donc des automates reconnaissant *ab* et *cd* les 2 automates reconnaissant les motifs $ab\Sigma^*cd$ et $ab\Sigma^+cd$ (on note $[\hat{}ab]$ un caractère quelconque autre que *a* ou *b*).

3.d] On veut construire l'automate reconnaissant la motif $ab\Sigma cd$. Contrairement aux deux motifs précédents, on cherche les deux motifs ab et cd séparés d'un seul caractère. Donc, une fois que l'on a vu le premier motif, si le motif cd n'apparaît pas un caractère après on repart du tout début. Attention, le caractère Σ peut aussi être un a, modifiant la façon dont on repart au début. Vérifiez en particulier que votre automate reconnaît le texte $ababzcd$. Pour simplifier, vous pourrez commencer par construire un automate non-déterministe, puis vous le déterminiserez.