Mémo n. 4

Les Automates

I32 Preuves et Analyses d'algorithmes

P. Véron

Problème : On recherche un certain motif dans un texte avec les contraintes suivantes : le texte est parcouru de gauche à droite, on ne connait à chaque instant que le caractère courant, il n'est pas possible de stocker les caractères précédemments lus et aucun retour en arrière n'est possible.

1. Généralités

Définition : *On appelle alphabet (et on note Σ) tout ensemble fini non vide de symboles.*
Exemple : $\Sigma = \{a, b, c, \ldots, z\}$.

Définition : *Les éléments d'un alphabet sont appelés lettres.*

Définition : *Un mot sur l'alphabet Σ est une suite finie de lettres.*
Exemple : $abfg$ est un mot sur l'alphabet $\Sigma = \{a, b, c, \ldots, z\}$.

Définition : *L'ensemble de tous les mots de l'alphabet Σ est noté Σ^*.*

Définition : *On appelle langage sur un alphabet Σ toute partie de Σ^*.*
Exemple : les mots du dictionnaire de la langue française constitue un langage sur l'alphabet $\Sigma = \{a, \ldots, z, A, \ldots, Z\}$. Ce langage est certainement différent de Σ^* !

Définition : *On définit sur les mots un opérateur dit de concaténation (et noté .) tel que si u et v sont 2 mots définis sur l'alphabet Σ, alors $u.v = uv$ est un mot sur Σ obtenu en concaténant les lettres de u et les lettres de v.*

Définition : *L'élément neutre pour l'opération de concaténation est noté ε. C'est le mot vide. Il vérifie pour tout mot m défini sur un alphabet Σ, $m.\varepsilon = \varepsilon.m = m$.*

Définition : *On appelle machine à caractères, un système composé d'un ruban (appelé ruban à caractères) comportant une suite de caractères accessibles un par un par l'intermédiaire d'une lucarne susceptible de se déplacer uniquement vers la droite.*

2. Principe d'un automate

Un automate est une machine pouvant se retrouver dans un nombre fini de configurations différentes : les états. Il reçoit des signaux qui au fur et à mesure le font basculer d'un état vers un autre : c'est l'étape de transition.

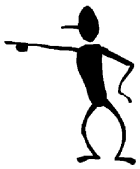
Définition : *Un automate fini M est un quintuplet $(Q, q_0, A, \Sigma, \delta)$ où*
- *. Q est un ensemble fini d'état,*
- *. q_0 est l'état initial,*
- *. $A \subset Q$ est un ensemble d'états dits terminaux,*
- *. Σ est un alphabet fini,*
- *. δ est une fonction de $Q \times \Sigma$ dans Q appelée fonction de transition.*

L'automate démarre dans l'état q_0 et à chaque fois qu'il lit un caractère $c \in \Sigma$, il bascule de l'état q courant vers l'état $\delta(q, c)$. Si l'état courant est un élément de A on dit alors qu'à cette étape la chaîne lue a été reconnue par l'automate.

3. Représentation d'un automate

Par sa fonction de transition

Exemple :
$Q = \{0, 1\}$, $q_0 = 0$, $A = \{1\}$, $\Sigma = \{a, b\}$.
$\delta(0, a) = 1$, $\delta(0, b) = 0$, $\delta(1, a) = 1$, $\delta(1, b) = 0$.



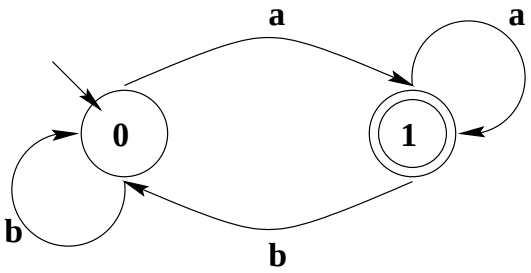
Par tableau

Les lignes sont indicées par les états et les colonnes par les lettres de l'alphabet. On trouve à la case d'indice (q, c) ($q \in Q$, $c \in \Sigma$) la valeur $\delta(q, c)$. La ligne correspondant à l'état initial est précédée d'une flèche, les états terminaux sont cerclés.

	a	b
→ 0	1	0
(1)	1	0

Par diagramme

Les états sont représentés par des cercles et les transitions par des flèches allant d'un état à un autre. Une flèche allant de l'état q à l'état q' est étiquettée par le symbole $c \in \Sigma$ si $\delta(q, c) = q'$. L'état initial est marqué par une flèche et les états finaux sont doublement cerclés.



Définition : *On note L_M le langage reconnu par l'automate M. $L_M = \{m \in \Sigma^*, m$ est reconnu par $M\}$.*

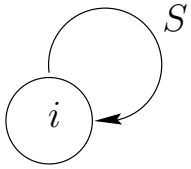
4. Calcul du langage reconnu par un automate M

Soit i un état quelconque on note L_i le langage reconnu à partir de l'état i. Soit $\Sigma = \{a_1, a_2, \ldots, a_p\}$ et soit S un sous-ensemble de Σ.

Définition : *Si $S = \{a_{i_1}, \ldots, a_{i_k}\}$ est un sous-ensemble de Σ, on note S^n l'ensemble des mots de longueur n construits à partir des lettres de S.*

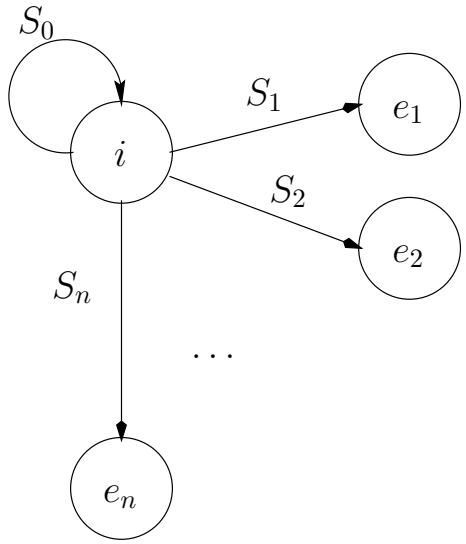
On a les cas suivants.

Cas 1



. Si i est terminal, alors $L_i = S^n = \{\{a_{i_1}, \ldots a_{i_k}\}^n, n \geqslant 0\}$,
. sinon $L_i = \emptyset$.

Cas 2



. S_j est le sous ensemble des symboles qui font basculer de l'état i à l'état e_j.

. Les S_j sont deux à deux disjoints.

. $\sum_{j=0}^{n} \text{card}(S_j) = \text{card}(\Sigma)$.

On a alors si i est terminal :

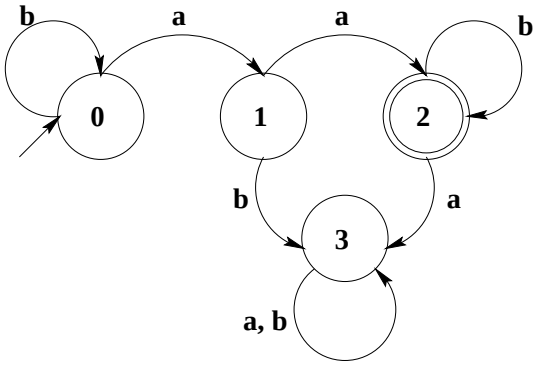
$$L_i = \underbrace{S_0.L_i}_{\text{si } S_0 \neq \emptyset} \cup S_1 L_{e_1} \cup \cdots \cup S_n L_{e_n} \cup \varepsilon$$

sinon

$$L_i = \underbrace{S_0.L_i}_{\text{si } S_0 \neq \emptyset} \cup S_1 L_{e_1} \cup \cdots \cup S_n L_{e_n}$$

Proposition : *Si $L_i = S.L_i \cup B$ où $S \subset \Sigma$ et $B \subset \Sigma^*$, alors si $B \neq \emptyset$, $L_i = S^n.B$ (n quelconque). Sinon on se retrouve dans le cas 1.*

Exemple : $\Sigma = \{a, b\}$



On a :

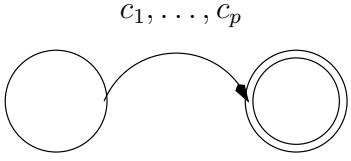
$$\begin{array}{rcccl} L_M = L_0 & = & b.L_0 \cup a.L_1 & & \text{(cas2)} \\ L_1 & = & a.L_2 \cup b.L_3 & & \text{(cas2)} \\ L_2 & = & b.L_2 \cup a.L_3 \cup \varepsilon & & \text{(cas2)} \\ L_3 & = & \emptyset & & \text{(cas1)} \end{array}$$

On en déduit :

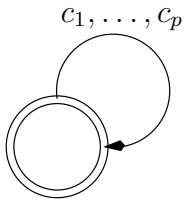
$$\begin{array}{rcll} L_2 & = & \{b^n, n \geqslant 0\}.\{\varepsilon\} & = \{b^n, n \geqslant 0\} \\ L_1 & = & a.L_2 & = \{ab^n, n \geqslant 0\} \\ L_0 & = & b.L_0 \cup \{aab^n, n \geqslant 0\} & = \{b^n, n \geqslant 0\}.\{aab^n, n \geqslant 0\} \\ & = & \{b^n aab^m, n \geqslant 0 \quad m \geqslant 0\} & \end{array}$$

5. Automate asssocié à un langage

. **Automate reconnaissant** $S = \{c_1, \ldots c_p\} \subset \Sigma$



. **Automate reconnaissant** $S^n, n \geqslant 0$

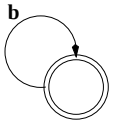


. **Automate reconnaissant** $L_1.L2$**,** $L_1, L_2 \subset \Sigma^*$

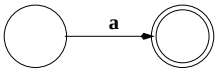
Soit M_1 l'automate reconnaissant L_1 et M_2 l'automate reconnaissant L_2, on identifie les états finaux de M_1 avec l'état initial de M_2 et on conserve comme états finaux ceux de M_2.

Exemple : Automate reconnaissant $\{b^n aab^m,\ n, m \geqslant 0\}$. Ce langage est constitué de la concaténation des langages suivants : $\{b^n, n \geqslant 0\}$, $\{a\}$, $\{a\}$, $\{b^m, m \geqslant 0\}$.

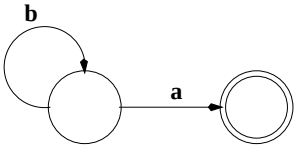
Automate reconnaissant $b^n, n \geqslant 0$:



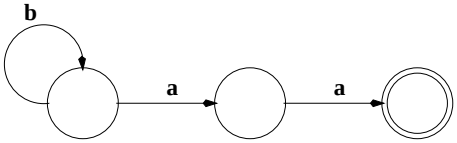
Automate reconnaissant a :



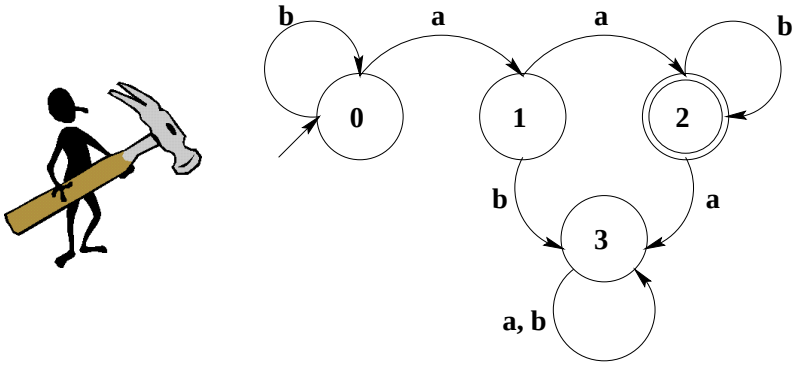
Automate reconnaissant $b^n a, n \geqslant 0$ (par concaténation des 2 automates précédents) :



Automate reconnaissant $b^n aa, n \geqslant 0$



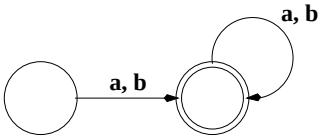
Automate reconnaissant $\{b^n aab^m, n, m \geqslant 0\}$ (ajout de l'état poubelle)



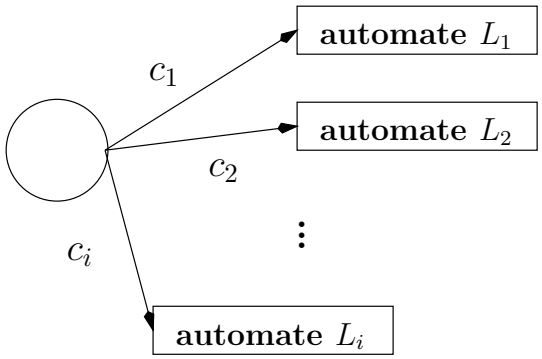
. **Automate reconnaissant** S^n**,** $n \geqslant j > 0$

Remarquons que $S^n = S^j.S^{n-j}$, $n - j \geqslant 0$. Il suffit donc de concaténer les automates reconnaissants S^j et S^{n-j}.

Exemple : $\{\{a,b\}^n, n \geqslant 1\}$. Ce langage peut s'écrire $\{\{a,b\}\{a,b\}^n, n \geqslant 0\}$. D'où l'automate :

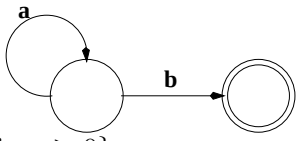


. **Automate reconnaissant** $c_1L_1 \cup \ldots c_iL_i$**,** $c_j \in \Sigma$**,** $L_j \in \Sigma^*$

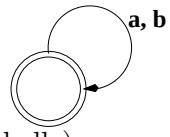


Exemple : $\{a^nb, n \geqslant 1\} \cup \{b\{a,b\}^m, m \geqslant 0\}$. Ce langage peut s'écrire aussi sous la forme suivante : $\{a\{a^nb\}, n \geqslant 0\} \cup \{b\{a,b\}^m, m \geqslant 0\}$.

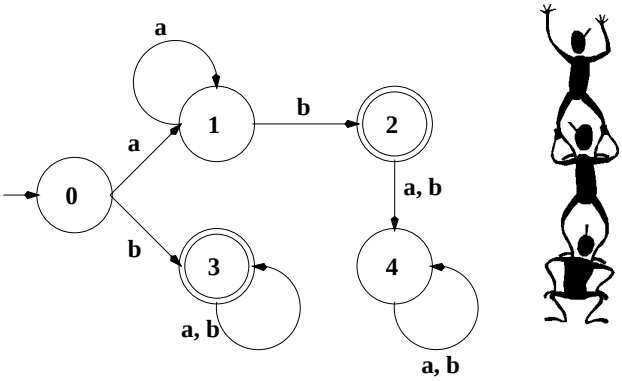
Automate reconnaissant $\{a^nb, n \geqslant 0\}$:



Automate reconnaissant $\{\{a,b\}^m, m \geqslant 0\}$:



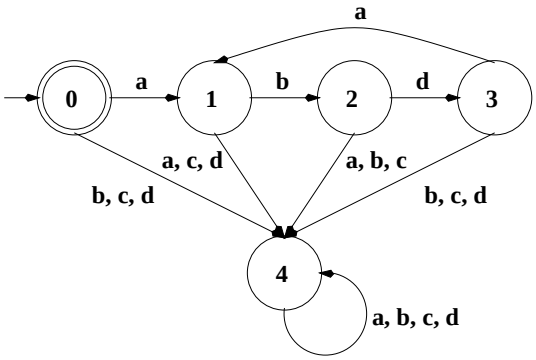
D'où l'automate final (avec l'état poubelle) :



. **Automate reconnaissant** $(c_1.c_2 \ldots c_p)^n$ $, n \geqslant 0,\ c_i \in \Sigma$

On construit l'automate reconnaissant $c_1.c_2 \ldots c_p$. L'état initial est terminal et on joint l'état final au 2ème état de l'automate en l'étiquetant par c_1.

Exemple : $\Sigma = \{a, b, c, d\}$ $L = \{(abd)^n, n \geqslant 0\}$

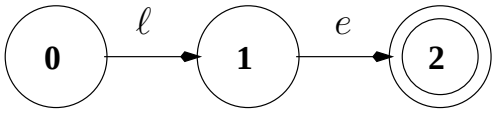


6. Application à la recherche de motif

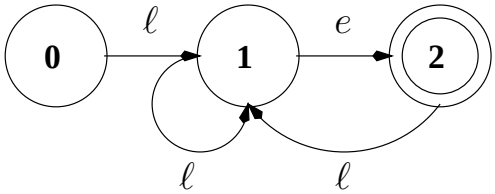
Dans le cadre de la recherche d'un motif dans un texte, l'automate permettant de réaliser ce travail ne comporte pas d'état poubelle. En effet il ne s'agit pas ici de soumettre une chaîne et de regarder si après le dernier caractère lu on se trouve dans un état terminal. On soumet un texte à un automate reconnaissant un certain motif et à chaque fois que l'on passe dans un état terminal, on sait que le motif a été détecté.

Principe de construction d'un automate lié à un motif : nous ne traiterons que le cas simple où les lettres du motif sont toutes distinctes. Dans ce cas là, la réalisation de l'automate se fait en 3 étapes. Pour illustrer ces 3 étapes, nous allons rechercher le motif `le` dans un texte dont les mots ont été construits sur l'alphabet $\Sigma = \{a,\ b, \ldots, z,\ ' '\}$.

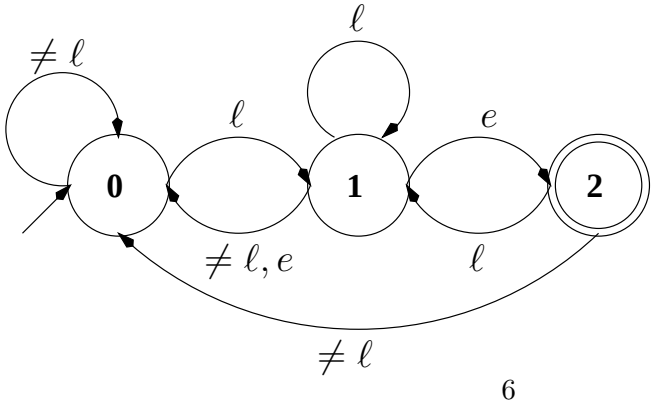
Etape 1 : Réalisation du squelette de l'automate. Pour un motif comportant n lettres, le squelette possède $n+1$ états. La transition de l'état $i-1$ à l'état i est étiquetée par la $i^{\text{ème}}$ lettre du motif ($1 \leqslant i \leqslant n$).



Etape 2 : Pour chaque état i on rajoute une transition vers l'état 1 étiquetée par la $1^{\text{ère}}$ lettre du motif ($1 \leqslant i \leqslant n$).



Etape 3 : Toute transition manquante retourne vers l'état 0.



7. Algorithme associé à un automate

Il existe une méthode générique permettant de définir l'algorithme associé à un automate.
Convention : On supposera que le ruban de caractères lu par l'automate se termine par un #.

Principe de construction : Soit M un automate à n états et soit $\Sigma = \{c_1, \ldots, c_p\}$. On numérote les états de 0 à $n-1$. Une variable `etat` est initialisée à 0. On parcourt le ruban tant qu'on ne rencontre pas le caractère #. Pour chaque état j de l'automate on compare le caractère lu avec le sous-ensemble S_j des caractères qui engendrent une transition de l'état j vers un autre état, on remet alors à jour la variable `etat` à l'aide de la fonction de transition.

```
1  Algo générique
2  données
3    carlu : caractère
4    etat : entier
5    début
6      etat ← 0
7      lire(carlu)
8      tant que carlu ≠'#' faire
9              selon que etat soit
10                          0 : si carlu ∈ S_0 alors etat ← δ(0, carlu) finsi
11                          1 : si carlu ∈ S_1 alors etat ← δ(1, carlu) finsi
12                          ⋮
13                  n − 1 : si carlu ∈ S_{n−1} alors etat ← δ(n − 1, carlu) finsi
14             finselon
15             lire(carlu)
16     fintq
17   fin
```

Exemple : Voici l'algorithme correspondant à la recherche du motif `le` :

```
1  Algo motif
2  données
3    carlu : caractère
4    etat : entier
5    début
6      etat ← 0
7      lire(carlu)
8      tant que carlu ≠'#' faire
9              selon que etat soit
10                  0 : si carlu ='ℓ' alors etat ← 1 finsi
12                  1 : si carlu ='e' alors
13                                        etat ← 2
14                    sinon si carlu ≠'ℓ' alors
15                                              etat ← 0
16                    finsi
18                  2 : si carlu ≠'ℓ' alors etat ← 0
19                                    sinon etat ← 1
20                    finsi
22             finselon
23             lire(carlu)
24     fintq
25   fin
```

Remarques :

. Cet algorithme ne fait rien, il ne fait que simuler le déroulement de l'automate. A vous de le compléter pour, par exemple, compter le nombre d'occurences du motif dans le texte (il suffit de gérer un compteur que l'on incrémentera à chaque fois que l'on a reconnu le motif).

. Dans le cas de la vérification de l'appartenance d'un mot à un langage, l'automate est constitué d'un état poubelle. Ainsi dès que la variable `état` prend la valeur de l'état poubelle, on sait qu'il est inutile de continuer à lire le mot. Sinon une fois le caractère # atteint, il suffit de vérifier si la variable `etat` correspond à un état terminal.