

<b>Automates finis</b>
------------------------

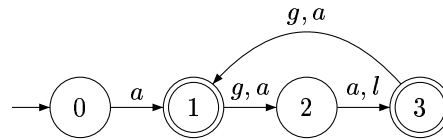
**Table des matières**

<b>1</b>	<b>Un premier exemple</b>	<b>2</b>
<b>2</b>	<b>Les automates et leurs langages</b>	<b>3</b>
2.1	Automates (déterministes) finis . . . . .	3
2.2	Automates complets . . . . .	3
2.3	Langage associé à un A.F. . . . .	4
2.4	Cas des automates non déterministes . . . . .	5
2.5	Automates émondés . . . . .	6
<b>3</b>	<b>Déterminisation</b>	<b>6</b>
3.1	Idée du théorème . . . . .	6
3.2	Théorème de déterminisation . . . . .	7
3.3	En pratique . . . . .	7
3.4	En Maple . . . . .	7
3.5	L'exemple du totomate . . . . .	7
3.6	Un dernier exemple . . . . .	8
3.7	Complexité . . . . .	9
<b>4</b>	<b>Le théorème de Kleene</b>	<b>9</b>
4.1	Théorème d'équivalence . . . . .	9
4.2	Quelques exemples . . . . .	9
4.3	Des conséquences fondamentales . . . . .	10
4.4	Quelques conséquences annexes . . . . .	10
<b>5</b>	<b><math>L</math> est-il rationnel ?</b>	<b>10</b>
5.1	Lemme de l'étoile . . . . .	10
5.2	Quelques conséquences du lemme de l'étoile . . . . .	10
5.3	Réciproque (?) du lemme de l'étoile . . . . .	11
5.4	Méthodes de réduction . . . . .	11
5.5	Méthode de séparation . . . . .	11
<b>6</b>	<b>Résiduels, minimisation</b>	<b>12</b>
6.1	Résiduels . . . . .	12
6.2	Caractérisation des langages rationnels . . . . .	12
6.3	Un exemple . . . . .	13
6.4	Théorème de minimisation . . . . .	13
6.5	Algorithme de minimisation . . . . .	13
<b>7</b>	<b>Epilogue</b>	<b>13</b>

# 1 Un premier exemple

On commence par présenter de façon informelle un exemple : le graphe suivant représente un automate  $\mathcal{A}$  sur l'alphabet  $A = \{a, b\}$ .

## EXEMPLE 1



Un premier automate

Il s'agit d'un *automate déterministe* (de chaque *état*, il part au plus une *transition* étiquetée par chaque lettre), *non complet* (depuis l'état 0, il ne part pas de transition étiquetée par  $g$ ). Cet automate est caractérisé par :

1. un alphabet de travail  $A = \{a, b, \dots, z\}$ ;
2. un ensemble d'état  $Q = \{0, 1, 2, 3\}$  (les *nœuds* du graphe, qui sont cerclés);
3. un ensemble de transitions représentées par des *arêtes* :

$$\Delta = \{(0, a, 1); (1, g, 2); (1, a, 2); (2, a, 3); (2, l, 3); (3, g, 1); (3, a, 1)\}$$

(la transition  $(0, a, 1)$  est représentée par l'arête reliant l'état 0 à l'état 1, étiquetée par  $a$ );

4. un état initial  $q_I = 0$  (particularisé sur la graphe par une arête entrante ne venant pas d'un autre état);
5. un ensemble d'état finaux (**ou finals**, acceptants, terminaux, mais pas terminaux!) qui sont représentés par un double cercle (ou une flèche sortante) : ici  $F = \{1, 3\}$ .

*Qu'est-ce qui se passe quand on lit un mot ?*

- Si on lit *agaga*, "on part de l'état initial, puis on passe à l'état 1 en lisant  $a$ , puis 2 en lisant  $g$ , puis 3, puis 1, puis 2". L'état 2 n'est pas terminal, donc le mot est rejeté. Même résultat si on lit *ag* ou *agagaaaa*.
- Si on lit *aglagla*, "on va de l'état initial à l'état 1, puis 2, puis...et enfin 1". Ce dernier état est terminal, donc on dit que *aglagla* est reconnu par l'automate, ou encore fait parti du langage associé à l'automate. Même chose pour les mots *agagaa* ou *aga*.
- Si on lit *agagaga*, on part de 1 pour arriver à l'état 2 (après avoir lu *agaga*). Mais il n'y a pas de transition partant de l'état 2 étiquetée par  $g$  : on dit que l'automate est *bloqué*, et le mot sera rejeté. Idem pour *agapouet* ou *agagatsointsoin*.

**REMARQUE 1** Les transitions peuvent également être formalisées par une *application partielle*  $\delta$  de  $Q \times A$  dans  $Q$ . Ici, on aurait  $\delta(0, a) = 1$ ,  $\delta(1, g) = 2$ , et  $\delta(3, z)$  n'est pas définie.

**REMARQUE 2** Lorsque  $(q_1, \alpha, q_2)$  et  $(q_1, \beta, q_2)$  sont deux transitions de  $\mathcal{A}$ , on les représente avec une seule arête entre  $q_1$  et  $q_2$ , étiquetée par  $a, b$ .

## 2 Les automates et leurs langages

### 2.1 Automates (déterministes) finis

#### DÉFINITION 1

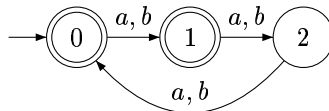
Un automate fini (déterministe) est un quintuplet  $\mathcal{A} = (A, Q, q_I, F, \delta)$ , où :

- $A$  est un alphabet ;
- $Q$  est un ensemble fini d'états ;
- $\delta$  est une application partielle de  $Q \times A$  dans  $Q$  (qu'on pourra choisir de représenter par l'ensemble des triplets  $(q, \alpha, \delta(q, \alpha))$  lorsque  $\delta(q, \alpha)$  est défini ;
- $q_i$  est l'état initial (celui à partir duquel on va "lire un mot" plus tard) ;
- $F \subset Q$  est l'ensemble des états terminaux ou acceptants ou ...

#### EXEMPLE 2 L'automate

$$\left( \{a, b\}, \{0, 1, 2\}, 0, \{0, 1\}, \{(0, a, 1), (0, b, 1), (1, a, 2), (1, b, 2), (2, a, 1), (2, b, 1)\} \right)$$

sera avantageusement représenté par :

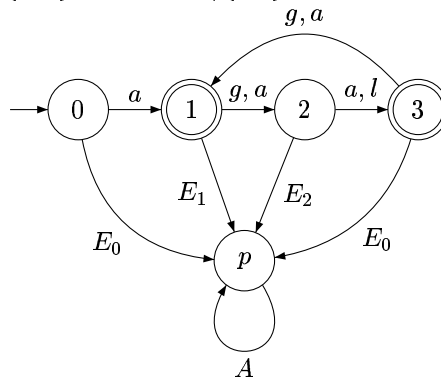


Un second exemple

### 2.2 Automates complets

L'automate est dit *complet* lorsque la fonction de transition est totale, c'est-à-dire si son ensemble de définition est  $Q \times A$ . Si  $\mathcal{A}$  n'est pas complet, on peut le *compléter* en un automate équivalent  $\mathcal{A}'$  : il suffit de rajouter un état *puits*  $q_p$ , et d'étendre la fonction de transition  $\delta$  à  $Q' \times A$  (avec  $Q' = Q \cup \{q_p\}$ ) en posant  $\delta'(q_p, \alpha) = q_p$  pour tout  $\alpha \in A$  et  $\delta'(q, \alpha) = q_p$  pour tout  $(q, \alpha) \in Q \times A$  qui n'est pas dans l'ensemble de définition de  $\delta$ . L'automate obtenu est le complété de  $\mathcal{A}$ . On verra que cet automate reconnaît le même langage que  $\mathcal{A}$ , en évitant les *blocages*.

EXEMPLE 3 L'automate du premier exemple se complète, en posant  $E_0 = A \setminus \{a\}$ ,  $E_1 = A \setminus \{a, g\}$  et  $E_2 = A \setminus \{a, l\}$  :



### 2.3 Langage associé à un A.F.

Lorsque  $(q, \alpha, q')$  est une transition de  $\mathcal{A}$ , on note  $q \xrightarrow[\mathcal{A}]{\alpha} q'$ , ou encore  $q \xrightarrow{\alpha} q'$  s'il n'y a pas d'ambiguïté sur  $\mathcal{A}$ . Ainsi, lorsqu'on écrira

$$q_0 \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} q_2 \xrightarrow{\alpha_3} \dots \xrightarrow{\alpha_n} q_n,$$

cela signifie que pour tout  $i \in \llbracket 1, n \rrbracket$ , on a  $(q_{i-1}, \alpha_i, q_i)$  qui est une transition de l'automate avec lequel on travaille.

#### DÉFINITION 2

- On dit qu'il existe un *chemin* étiqueté par  $w = \alpha_1 \dots \alpha_n \in A^*$  entre  $q$  et  $q'$  lorsqu'on a la suite de transitions :

$$q = q_0 \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} q_2 \xrightarrow{\alpha_3} \dots \xrightarrow{\alpha_n} q_n = q'.$$

- Un chemin *victorieux* est un chemin partant de l'état initial et arrivant à un état final.
- Un mot  $w$  est *accepté* par  $\mathcal{A}$  s'il existe un chemin victorieux étiqueté par  $w$ . L'ensemble de ces mots sera le langage *associé* à  $\mathcal{A}$ , ou encore *reconnu* par  $\mathcal{A}$ , et sera noté  $\mathcal{L}(\mathcal{A})$ .

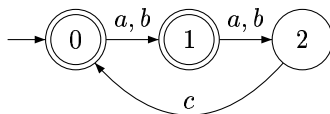
**REMARQUE 3** Dans le cas des automates déterministes, il existe au plus un chemin étiqueté par un mot donné et partant de l'état initial : s'il y a blocage (cf *agatsointsoin*), le mot n'est pas reconnu. Par contre, s'il n'y a pas blocage, le mot est reconnu si et seulement si l'état dans lequel on arrive en lisant  $w$  depuis  $q_i$ , noté  $\delta(q_i, w)$ , est final. On dispose donc d'un moyen simple (linéaire en  $|w|$ ) pour vérifier si un mot est ou non reconnu par un automate. Cela se compliquera dans les cas des automates non déterministes, puisque plusieurs transitions seront possibles en lisant une même lettre depuis un même état.

**EXEMPLE 4** On peut décrire les langages reconnus par les deux premiers exemples de ce chapitre :

- *Commençons par le second exemple : il n'y a jamais de blocage, et on a clairement  $\delta(0, w) = |w| \pmod{3}$  (reste dans la division euclidienne par 3). Le langage accepté est donc l'ensemble des mots dont la longueur est congrue à 0 ou 1 modulo 3, ce qui correspond à l'expression rationnelle  $((a+b)^3)^*(\varepsilon + a + b)$ .*
- *Dans le cas du premier exemple, il y a les chemins aboutissant à l'état 1, et ceux à l'état 3. Dans les deux cas, il faut commencer par un  $a$ , puis on peut faire un certain nombre de boucles  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ , de sorte qu'une expression rationnelle décrivant le langage reconnu par l'automate est :*

$$a.(a+g).(a+l).(a+g)^*.(a+g).(a+l).$$

**EXERCICE 1** Déterminer une expression rationnelle simple décrivant le langage associé à l'automate suivant :



(l'alphabet est  $\{a, b, c\}$ ).

## 2.4 Cas des automates non déterministes

On va décrire une classe d'automate incluant les automates déterministes, mais pour lesquels on s'autorise :

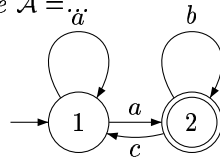
- un *ensemble* d'états initiaux ;
- plusieurs transitions étiquetées par la même lettre issues d'un même état ;
- un nouveau type de transition : les  $\varepsilon$ -*transitions*.

### DÉFINITION 3

Un automate fini (non déterministe) est un sextuplet  $(A, Q, I, F, \delta, T_\varepsilon)$ , avec :

- $A$  un alphabet (ensemble fini) ;
- $Q$  est un ensemble (fini) d'états ;
- $I \subset Q$  est un ensemble d'états initiaux ;
- $F \subset Q$  est un ensemble d'états acceptants ;
- $\delta$  est une application de  $Q \times A$  dans  $\mathcal{P}(Q)$ . Il s'agit de la "fonction de transition" : chaque élément  $q$  de  $\delta(p, \alpha)$  donne lieu à une arête étiquetée par  $\alpha$  entre les sommets  $p$  et  $q$  du graphe représentant  $\mathcal{A}$  ;
- $T_\varepsilon \subset Q \times Q$  est l'ensemble des " $\varepsilon$ -transitions", représentées par des arête étiquetées par  $\varepsilon$ .

EXEMPLE 5 L'automate  $\mathcal{A} =$



### REMARQUES 4

- Par défaut, quand on parle d'un automate fini, il est non déterministe. Cela dit, les  $\varepsilon$ -transitions sont "à la frontière extérieure" du programme. Il est donc déconseillé de les utiliser dans une épreuve écrite, sauf mention explicite sur l'énoncé.
- $\delta$  peut être représentée par  $\Delta \subset Q \times A \times Q$  constitué de l'ensemble des  $(p, \alpha, q)$ , où  $q \in \delta(p, \alpha)$ . De toute façon, on privilégie en général la représentation sous forme de graphe...

### DÉFINITION 4

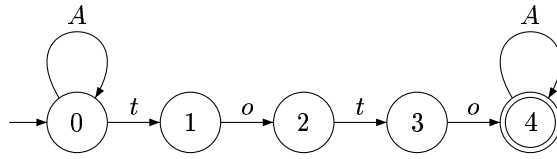
- Un chemin
- Un chemin sera dit victorieux
- Un mot est accepté...

### REMARQUES 5

- 
- 

EXEMPLE 6 L'automate vu à l'exemple 5 reconnaît  $a^+b^+(ca^+b^+)^*$ .

EXEMPLE 7 L'automate suivant reconnaît  $A^*totoA^*$ , c'est-à-dire l'ensemble des mots contenant toto comme facteur.



*LE totomate*

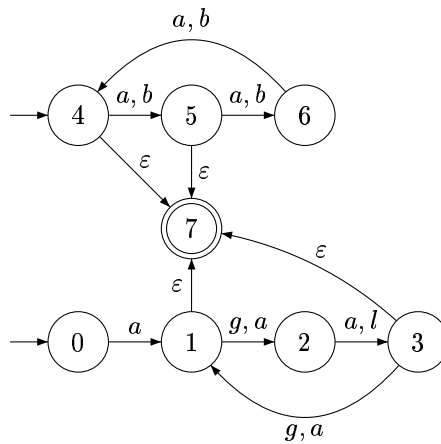
**EXERCICE 2** Construire un automate reconnaissant le langage constitué des mots ayant agaga pour suffixe.

**EXERCICE 3** Construire un automate reconnaissant le langage constitué des mots ayant agaga ou tsointsoin pour sous-mot.

**DÉFINITION 5**

Un langage sera dit *reconnaisable* s'il est le langage associé à un automate fini. L'ensemble des langages reconnaissables sur l'alphabet  $A$  est noté  $\text{Rec}(A)$ .

**EXERCICE 4** Décrire avec une expression rationnelle simple le langage reconnu par l'automate suivant sur l'alphabet  $\{a, b, g, l\}$  :



**EXERCICE 5** Représenter un automate associé au même langage, mais avec un seul état initial (les  $\epsilon$ -transitions sont autorisées...).

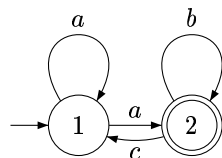
**2.5 Automates émondés**

...

**3 Déterminisation**

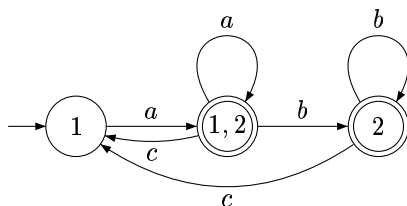
**3.1 Idée du théorème**

On va chercher à déterminer les mots reconnus par l'automate de l'exemple 5. L'alphabet de travail est  $\{a, b, c\}$ .



L'automate à déterminer

gnagnagna



Et voilà le travail...

### 3.2 Théorème de déterminisation

THÉORÈME 1 (Déterminisation)

Pour tout automate fini  $\mathcal{A}$ , il existe un automate fini déterministe complet  $\mathcal{A}'$  tel que  $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ .

PREUVE : Blabla...

■

### 3.3 En pratique

Il est inutile de considérer les  $N = 2^{|\mathcal{Q}|}$  états : il suffit de considérer ceux qui sont accessibles depuis l'état initial du déterminisé. Il n'est pas non plus utile de représenter l'état  $\{\}$  du déterminisé, qui correspond à un puit.

La méthode la plus pratique pour récupérer tous ces états est de faire un parcours en largeur du graphe, en tenant à jour une liste  $L$  d'états du déterminisé qu'il reste à traiter. Au départ,  $L$  est réduit à l'état initial du déterminisé, puis à chaque étape,

EXERCICE 6 Refaire la déterminisation de l'automate de l'exemple 5, en supposant cette fois que l'alphabet de travail est  $\{t, o, a\}$ .

### 3.4 En Maple

On donne en annexe un programme Maple de lecture limpide qui calcule le déterminisé du totomate de l'exemple 7.

### 3.5 L'exemple du totomate

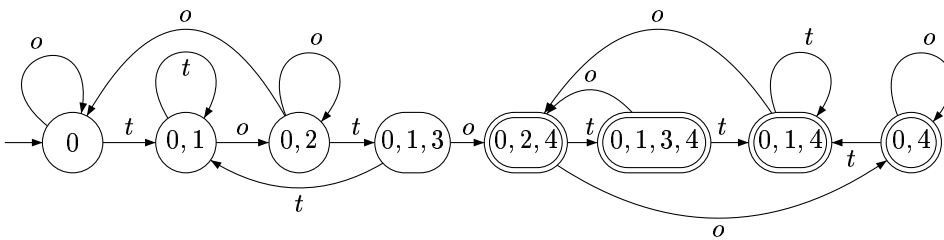
Il peut être pratique, pour ne pas oublier de transition ou d'état, de calculer les transitions à l'aide d'un tableau où on consigne les  $\delta_D(P, \alpha)$  pour tous les macros-états  $P$  : dès qu'un nouveau macro-état apparaît, on crée une ligne supplémentaire.

$q \backslash \alpha$	$t$	$o$
0	01	0
1	-	2
2	3	-
3	-	4
4	4	4

$P \backslash \alpha$	$t$	$o$
0	01	0
01	01	02
02	013	0
013	01	024
024	0134	04
0134	014	024
04	014	04
014	014	024

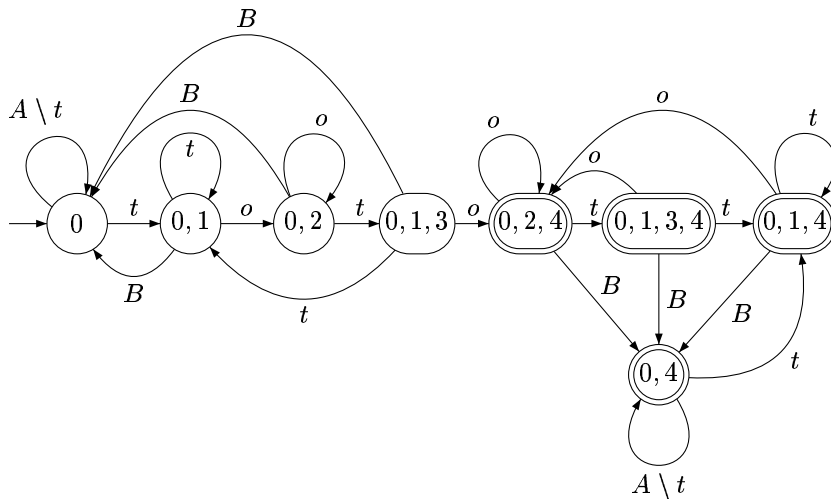
$\rightarrow det$

Et il n'y a plus qu'à dessiner :



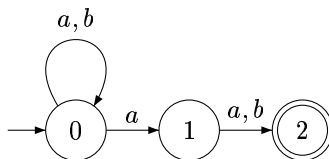
**REMARQUES 6**

- (états acceptants...)
- Si l'alphabet de travail est...

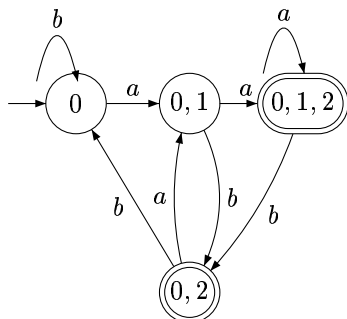


**3.6 Un dernier exemple**

L'automate



se déterminisera en :



### 3.7 Complexité

## 4 Le théorème de Kleene

### 4.1 Théorème d'équivalence

Le résultat qui suit est le pivot de ce chapitre : il relie deux types de descriptions d'un langage : l'une est lexicale, et l'autre...

THÉORÈME 2 (de Kleene)

Un langage est rationnel si et seulement si il est reconnu par un automate fini. Soit encore :  $\text{Rat}(A) = \text{Rec}(A)$

PREUVE : Une seule inclusion est au programme ; on commence par celle-ci :

-  
-

■

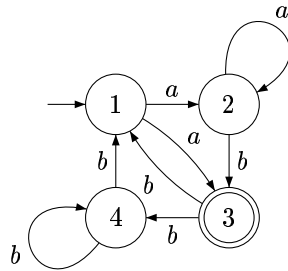
### 4.2 Quelques exemples

EXEMPLE 8

EXERCICE 7

EXEMPLE 9

EXERCICE 8 Donner une expression rationnelle définissant le langage reconnu par



### 4.3 Des conséquences fondamentales

### 4.4 Quelques conséquences annexes

## 5 $L$ est-il rationnel ?

### 5.1 Lemme de l'étoile

Le résultat suivant est un outil anecdotique qui permet en général d'établir que tel ou tel langage n'est pas rationnel. Les anglo-saxons l'appellent le "*pumping lemma*", et on le trouve parfois dans la littérature française sous le nom de "*lemme de gonflement*" (bof...) ou "*lemme d'itération*".

**THÉORÈME 3** (Lemme de l'étoile) Soit  $L$  un langage rationnel infini. Alors il existe un entier  $N$  tel que tout mot  $w$  de  $L$  de longueur  $\geq n$  peut se décomposer sous la forme  $w_1 w_2 w_3$ , où  $w_2 e q \varepsilon$ , et  $w_1 w_2^n w_3 \in L$  pour tout entier  $n \in \mathbb{N}$ .

PREUVE : ■

**REMARQUE 7** Dans les hypothèses, on oublie parfois... De même, dans la conclusion, oublier la condition  $w_2 e q \varepsilon$  enlève une grande partie de l'intérêt du lemme de l'étoile.

### 5.2 Quelques conséquences du lemme de l'étoile

Le lemme de l'étoile n'a pas un rôle central (loin de là) dans la théorie des automates finis, mais il fait partie du programme... donnons en donc quelques applications simples et de bon goût.

**PROPOSITION 1** Le langage  $\{a^n b^n \mid n \in \mathbb{N}\}$  n'est pas rationnel.

Du même tonneau :

**PROPOSITION 2**

On termine par un exercice bien difficile :

EXERCICE 9 *Le langage constitué des  $a^p$  pour  $p$  entier premier n'est pas rationnel (l'alphabet est  $\{a\}$ ).*

### 5.3 Réciproque (?) du lemme de l'étoile

Comme cela n'aura échapper à personne, l'énoncé du lemme de l'étoile n'est pas une équivalence mais une implication. De fait, la réciproque est fautive : on peut trouver des langages qui ne sont pas rationnels mais qui vérifient la conclusion du lemme de l'étoile.

EXERCICE 10 *Montrer que les langages suivants sont dans ce cas :*

- 
- 
- 

On peut énoncer un résultat de nature semblable qui établit une équivalence. . .

#### PROPOSITION 3

La preuve est laissée en exercice. On pourra appliquer ce résultat dans ce qui suit :

EXERCICE 11

### 5.4 Méthodes de réduction

De façon informelle, les méthodes de *réduction* consistent à . . .

EXEMPLE 10

En utilisant le résultat et la méthode de l'exemple précédent, on fera sans mal l'exercice suivant :

EXERCICE 12 *Montrer que l'ensemble des*

REMARQUE 8 Attention, il est très facile (si si. . .) de faire des erreurs de raisonnement du type "si blabla".

### 5.5 Méthode de séparation

Il s'agit d'une méthode qui permet, dans bien des cas, de minorer le nombre d'états d'un automate reconnaissant un langage donné, voire de montrer qu'un langage n'est pas rationnel, en exhibant "un nombre infini d'états" dans un automate censé reconnaître le langage en question.

## 6 Résiduels, minimisation

### 6.1 Résiduels

#### DÉFINITION 6

Soit  $L$  un langage (quelconque) sur un alphabet. On définit :

-  
-  
-

Les *résiduels* de  $L$  sont tous les langages de la forme  $w^{-1}L$ , pour  $w \in A^*$ .

REMARQUE 9 On vient de définir les résiduels gauches. Les résiduels droits se définissent de la façon à laquelle on pense, et vérifient les mêmes propriétés que les résiduels gauches. Mais ceux-là ne nous intéresseront pas.

Les résultats qui suivent ont des preuves simples qui sont laissées au lecteur. Il convient de savoir faire ces preuves quasi-instantanément, ce qui permet de se corriger quand on vient d'utiliser une relation bien pratique mais douteuse.

#### PROPOSITION 4

-  
-  
-  
-  
-  
-

Le résultat qui suit est essentiel pour la suite :

PROPOSITION 5 *Soit  $\mathcal{A}$  un automate fini déterministe complet*

PREUVE :

■

### 6.2 Caractérisation des langages rationnels

On va voir une caractérisation très simple des langages rationnels en terme de résiduels :

THÉORÈME 4 *Un langage  $L$  est rationnel si et seulement si il admet un nombre fini de résiduels.*

PREUVE : Le sens direct est simple. Pour la réciproque, on va construire l'automate des résiduels associé à  $L$ , qui aura une grande importance par la suite.

-  
-

■

### 6.3 Un exemple

### 6.4 Théorème de minimisation

Notons dès maintenant qu'une conséquence immédiate de la proposition et du théorème 4 est la :

**PROPOSITION 6** *Soit  $L$  un langage rationnel admettant  $N$  résiduels*

**PREUVE :** L'application  $q \mapsto L_q$  réalise une surjection (proposition) de  $Q$  sur l'ensemble des résiduels de  $L$  ■

Dans la suite, on va montrer l'existence d'un automate "canonique" (en un sens qui sera précisé) associé à chaque langage rationnel, ayant un nombre d'état minimal. Le premier point de vue n'est pas constructif, mais on verra ensuite un moyen effectif pour construire cet automate.

**THÉORÈME 5** (de minimisation)

Soit  $L$  un langage rationnel admettant  $N$  résiduels. Alors :

1. Il existe un automate fini déterministe complet à  $N$  états reconnaissant  $L$  ("automate minimal") ;
2. tout autre automate fini du même type reconnaissant  $L$  possède au moins  $N$  états ;
3. deux automates minimaux pour un même langage sont isomorphes.

**PREUVE :** ■

On a l'habitude de parler, par abus de langage de "*l'automate minimal*" d'un langage, à la place de "*l'un des automates déterministes complets de plus petit nombre d'état*".

### 6.5 Algorithme de minimisation

On va voir maintenant un moyen constructif d'obtenir un automate minimal. Un autre algorithme sera donné en exercice.

**EXEMPLE 11** *Reprenons l'exemple ? ? ?*

On verra dans les exercices d'autres exemples et quelques résultats complémentaires.

## 7 Epilogue

Après avoir défini de façon syntaxique les langages rationnels, on a vu l'aspect calculatoire de ces langages. Ce double point de vue se retrouve souvent en calculabilité :

- On définit indépendamment des langages dits *algébriques* et une extension des automates finis : les *automates à piles*, et on établit que ceux-ci reconnaissent ceux-là<sup>1</sup>.

---

<sup>1</sup>Plus précisément,...

- Même chose entre les...et les *réseaux de Pétri* (grosso modo, des automates finis dont chaque état possède une pile).

–

Notons pour terminer que l'on s'est attaché dans ce chapitre à bien montrer les aspects constructifs des divers résultats :