

Table des matières

Préface	5
Introduction	7
Remerciements	9
I Langages formels	11
1 Langages rationnels	13
1.1 Premières définitions	13
1.2 Opérations rationnelles	15
1.3 Combinatoire des mots	16
1.3.1 Périodicités	16
1.3.2 Mots infinis	20
1.3.3 Motifs inévitables	21
1.3.4 Codes	24
1.4 Un peu d'ordre	26
1.4.1 Quasi-ordres sur les mots	29
1.4.2 Ordres sur les mots	30
1.4.3 Quasi-ordres sur les arbres	31
1.5 Langages rationnels	33
1.5.1 Expressions rationnelles	33
1.5.2 Automates	35
1.6 Automates déterministes	41
1.7 Automate minimal	44
1.7.1 Quotients	45
1.7.2 Congruence de Nerode	46
1.7.3 Calcul de l'automate minimal	48
1.8 Propriétés de clôture	51
1.8.1 Opérations booléennes	51
1.8.2 Morphisme et morphisme inverse	51
1.9 Lemme de l'étoile et ses variantes	53
1.10 Hauteur d'étoile	57
1.11 Reconnaissance par morphisme	58
1.12 Langages sans étoile	65
1.13 Compléments	71
1.13.1 Conjecture de Černý	71
1.13.2 Rationnels d'un monoïde quelconque	71

Langages rationnels

LES LANGAGES RATIONNELS sont le premier niveau de la hiérarchie de Chomsky. Cette famille est constituée des langages acceptés par les automates finis qui sont le modèle le plus simple de machines. Cette famille de langages jouit de propriétés remarquables. Elle est en particulier close par de très nombreuses opérations.

La théorie des automates est une théorie relativement riche et certains de ses résultats sont de véritables perles. Elle entretient aussi des liens avec beaucoup d'autres domaines comme la dynamique symbolique, la combinatoire, l'algèbre, la topologie, la théorie des jeux, l'arithmétique, la logique et l'algorithmique. Certains de ces liens avec l'algèbre et la logique sont abordés dans ce support de cours (cf. sections 1.11 et 3.8 par exemple).

Les automates se sont aussi imposés comme un outil incontournable d'un point de vue pratique. Tout éditeur de texte un peu évolué comprend une fonction de recherche à partir d'une expression rationnelle. Cette recherche commence toujours par la construction d'un automate équivalent à l'expression. Les automates sont aussi utilisés dans le domaine de la vérification. Il s'agit dans ce domaine de vérifier qu'un objet tel un composant logiciel ou un protocole est bien conforme à une spécification.

Ce chapitre introduit les notions de mots et de langages. Après quelques éléments de combinatoire des mots, il développe les langages rationnels et les automates finis qui les acceptent. Une dernière partie est consacrée à la reconnaissance par morphismes des langages rationnels et à la caractérisation des langages sans étoile. Pour l'essentiel du chapitre, une bonne référence est [Per90].

1.1 Premières définitions

Nous commençons par quelques définitions élémentaires. La terminologie est empruntée à la linguistique et provient du fait qu'un des objectifs initiaux de ce domaine était la modélisation des langues naturelles.

Définition 1.1. – Un *alphabet* est un ensemble fini (souvent noté A ou Σ) dont les éléments sont appelés *lettres* ou *symboles*.

- Un *mot* w sur l'alphabet A est une suite finie $w_1 w_2 \cdots w_n$ de lettres de A . L'entier n est appelé la *longueur* de w qui est notée $|w|$.
- Le *mot vide* noté ε ou parfois 1 correspond à l'unique mot de longueur 0.
- Un *langage* sur l'alphabet A est un ensemble de mots sur A .
- L'*ensemble de tous les mots* sur l'alphabet A est noté A^* et il est appelé le *monoïde libre* sur A .

Un mot est écrit en juxtaposant ses éléments sans séparateur. Par exemple, le mot w de longueur 3 dont les éléments sont a , b et a est simplement écrit aba . Un mot de longueur 1 est écrit comme son unique lettre. Cette confusion entre une lettre a et le mot de longueur 1 dont l'unique élément est a est sans conséquence et même souvent très pratique.

La *concaténation* des deux mots $u = u_1 \cdots u_m$ et $v = v_1 \cdots v_n$ est le mot noté $u \cdot v$ ou uv et égal à $u_1 \cdots u_m v_1 \cdots v_n$ obtenu par simple juxtaposition. C'est une opération associative dont le mot vide ε est l'élément neutre. C'est pour cette raison que le mot vide est parfois noté 1.

Exemple 1.2. Si $u = abaa$ et $v = bab$, on a $uv = abaabab$ et $vu = bababaa$. La concaténation n'est pas commutative.

Comme la concaténation est une loi de composition interne associative avec un élément neutre, elle munit l'ensemble A^* de tous les mots d'une structure de monoïde. Pour cette raison, la concaténation est aussi appelée *produit*. Ce monoïde est dit libre parce que les seules équations qu'il satisfait sont celles qui sont conséquences de la définition de monoïde. Ce fait est énoncé de manière rigoureuse à la proposition 1.116 (p. 59). La notion de *structure libre* peut être définie de manière très générale dans le cadre de l'algèbre universelle [Alm94].

Exercice 1.3. Soit l'alphabet $A = \{0, 1\}$ et soit la suite de mots $(f_n)_{n \geq 0}$ définie par récurrence par $f_0 = 1$, $f_1 = 0$ et $f_{n+2} = f_{n+1}f_n$ pour $n \geq 0$.

1. Montrer que si $n \geq 2$, alors f_n se termine par 01 si n est pair et par 10 sinon.
2. Montrer que le mot g_n obtenu en supprimant les deux dernières lettres de f_n , c'est-à-dire $f_n = g_n 01$ si n pair et $f_n = g_n 10$ sinon est un palindrome. On rappelle qu'un *palindrome* est un mot qui reste identique lorsqu'il est retourné.

Solution.

1. La première propriété se montre facilement par récurrence sur n . On vérifie en effet que $f_2 = 01$ se termine par 01 et que $f_3 = 010$ se termine par 10. La relation $f_{n+2} = f_{n+1}f_n$ montre ensuite que f_{n+2} se termine comme f_n .
2. On remarque que $g_2 = \varepsilon$, $g_3 = 0$ et $g_4 = 010$ sont bien des palindromes et on montre le résultat par récurrence sur n . On suppose le résultat acquis pour f_n et f_{n+1} et on montre le résultat pour f_{n+3} . On suppose d'abord que n est pair. Les mots f_n et f_{n+1} s'écrivent alors $f_n = g_n 01$ et $f_{n+1} = g_{n+1} 10$ où g_n et g_{n+1} sont des palindromes. Les mots f_{n+2} et f_{n+3} sont alors égaux à $f_{n+1}f_n = g_{n+1}10g_n 01$ et $g_{n+1}10g_n 01g_{n+1}10$ d'où on tire $g_{n+3} = g_{n+1}10g_n 01g_{n+1}$ qui est un palindrome. Le cas n impair se montre de la même façon.

Un mot u est un *préfixe* (resp. *suffixe*) d'un mot w s'il existe un mot v tel que $w = uv$ (resp. $w = vu$). Un mot u est un *facteur* d'un mot w s'il existe deux mots v et v' tels que $w = vuv'$.

Exercice 1.4. Soit $w = w_1 \cdots w_n$ un mot de longueur n . Par un léger abus, on utilise la notation w_i même si l'entier i n'est pas dans l'intervalle $[1; n]$. On note ainsi la lettre $w_{i'}$ où i' est l'unique entier tel que $1 \leq i' \leq n$ et $i' \equiv i \pmod n$. On appelle *occurrence circulaire* d'un mot $u = u_1 \cdots u_p$ un entier k dans l'intervalle $[1; n]$ tel que $w_{k+i-1} = u_i$ pour chaque $1 \leq i \leq p$.

Montrer que pour tout alphabet A et tout entier k , il existe un mot w_k tel que tous les mots de longueur k sur A ont exactement une occurrence circulaire dans w_k . Un tel mot est appelé un mot de *de Bruijn* d'ordre k . Les mots 1100 et 11101000 sont des mots de de Bruijn d'ordre 2 et 3 sur l'alphabet $\{0, 1\}$.

On pourra considérer l'automate dont l'ensemble des états est A^{k-1} et dont l'ensemble des transitions est $\{au \xrightarrow{a} ub \mid a, b \in A \text{ et } u \in A^{k-2}\}$. Montrer qu'il y a bijection entre les cycles eulériens de cet automate et les mots de de Bruijn d'ordre k .

Soit A et B deux alphabets. Un morphisme de monoïdes μ de A^* dans B^* est une application de A^* dans B^* qui est compatible avec la structure de monoïde. Ceci signifie que l'image du mot vide est le mot vide ($\mu(\varepsilon) = \varepsilon$) et que $\mu(uv) = \mu(u)\mu(v)$ pour tout mot u et v de A^* . L'image d'un mot $w = w_1 \cdots w_n$ est donc égale à $\mu(w) = \mu(w_1) \cdots \mu(w_n)$. Le morphisme est complètement déterminé par les images des lettres. Un morphisme μ est dit *effaçant* s'il existe une lettre a tel que $\mu(a) = \varepsilon$.

Exemple 1.5. Soit A l'alphabet $\{a, b\}$ et soit $\mu : A^* \rightarrow A^*$ le morphisme de monoïdes déterminé par $\mu(a) = ab$ et $\mu(b) = a$. Ce morphisme est non-effaçant. L'image du mot aba par μ est le mot $abaab = ab \cdot a \cdot ab$.

1.2 Opérations rationnelles

On définit maintenant les opérations rationnelles qui permettent de construire de nouveaux langages.

Définition 1.6 (Union, Produit). – L'*union* est l'opération qui à deux langages L et L' associe le langage $L + L' = L \cup L'$.
– Le *produit* est l'opération qui à deux langages L et L' associe le langage $L \cdot L' = LL' = \{uv \mid u \in L \text{ et } v \in L'\}$.

Exemple 1.7. Soient les deux langages $L = \{u \in A^* \mid |u| \text{ pair}\}$ et $L' = \{u \in A^* \mid |u| \text{ impair}\}$. On a alors les égalités suivantes.

- $L + L' = A^*$ – $LL' = L' = L'L$
- $L'L' = L \setminus \{\varepsilon\}$ – $LL = L$ (L est un sous-monoïde de A^*)

Définition 1.8 (Étoile). Soit $L \subseteq A^*$, on définit :

$$L^0 = \{\varepsilon\}, \quad L^{i+1} = LL^i, \quad L^* = \bigcup_{i \geq 0} L^i$$

Cette définition justifie donc *a posteriori* la notation A^* puisque l'ensemble de tous les mots est bien l'étoile de l'alphabet. Il faut remarquer qu'en général L^i est différent de $\{u^i \mid u \in L \text{ et } i \geq 0\}$.

Un *sous-monoïde* de A^* est un langage de A^* qui contient le mot vide et qui est clos pour la concaténation. On vérifie que pour tout langage L , le langage L^* est le plus petit sous-monoïde de A^* qui contient L . Inversement tout sous-monoïde K de A^* est de la forme L^* puisque $K = K^*$.

Exemple 1.9. – Si $L = \{a, ba\}$, alors L^* est constitué de tous les mots dans lesquels chaque b est suivi d'un a .

- Si $L = \{a^n b \mid n \geq 0\}$, alors L^* est constitué du mot vide ε et de tous les mots qui se terminent par un b .

– Si $L = \emptyset$, alors L^* est le langage $\{\varepsilon\}$ uniquement constitué du mot vide.

Pour un langage L , on note L^+ le langage $LL^* = L^*L = \bigcup_{i \geq 1} L^i$. Si L contient le mot vide, on a $L^+ = L^*$ et sinon on a $L^+ = L^* \setminus \{\varepsilon\}$. Le langage A^+ est par exemple l'ensemble des mots non vides.

1.3 Combinatoire des mots

La combinatoire des mots est l'étude des propriétés structurelles des mots. Dans cette partie, on s'intéresse d'abord à des questions de périodicité puis à des questions de motifs inévitables.

1.3.1 Périodicités

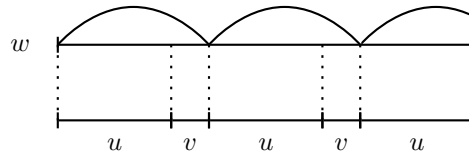


FIGURE 1.1 – Période d'un mot w

Soit $w = w_1 \cdots w_n$ un mot sur un alphabet A . Une *période* de w est un entier p entre 1 et $|w|$ tel que $w_i = w_{i+p}$ pour tout $1 \leq i \leq |w| - p$. Par convention, la longueur de w est une période de w . On note $P(w)$ l'ensemble des périodes de w . Il faut remarquer qu'une période ne divise pas nécessairement la longueur du mot. Si p est une période de w , le mot w se factorise $w = (uv)^k u$ où l'entier k vérifie $k \geq 1$, l'entier p est égal à $|uv|$ et le mot v est non vide (cf. figure 1.1).

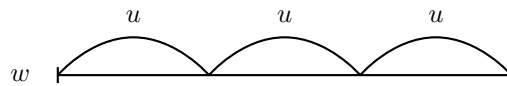


FIGURE 1.2 – Mot non primitif $w = u^3$

Un mot w est *primitif* s'il ne s'écrit pas $w = u^k$ avec $k \geq 2$. Autrement dit, un mot est primitif si sa longueur est l'unique période qui divise sa longueur (cf. figure 1.2). Tout mot w s'écrit de manière unique $w = u^k$ où l'entier k vérifie $k \geq 1$ et le mot u est primitif. L'entier k est égal à 1 quand w est primitif.

Exemple 1.10. Le mot $w = 010101$ a $P(w) = \{2, 4, 6\}$ pour ensemble de périodes. Il n'est pas primitif car il s'écrit $w = (01)^3$. Le mot $w' = 01001010010$ a $P(w') = \{5, 8, 10, 11\}$ pour ensemble de périodes. Il est primitif car aucune de ses périodes autre que sa longueur ne divise sa longueur.

Le plus grand commun diviseur de deux entiers p et q est noté $p \wedge q$. Le théorème de Fine et Wilf est le résultat de base de la combinatoire des mots. Les preuves de beaucoup d'autres résultats font appel à ce théorème.

Théorème 1.11 (Fine et Wilf 1965). *Si p et q sont deux périodes d'un mot w de longueur supérieure ou égale à $p + q - p \wedge q$, alors $p \wedge q$ est aussi une période de w .*

Avant de donner la preuve du théorème, on montre que la borne du théorème est optimale. Soit $(f_n)_{n \geq 0}$ la suite de mots définie par récurrence par $f_0 = 1$, $f_1 = 0$ et $f_{n+2} = f_{n+1}f_n$. Les premiers mots de cette suite sont $f_2 = 01$, $f_3 = 010$, $f_4 = 01001$ et $f_5 = 01001010$. Les mots de cette suite sont appelés *mots de Fibonacci*. En effet, la longueur de chaque mot f_n est le nombre de Fibonacci F_n où la suite $(F_n)_{n \geq 0}$ est définie par $F_0 = F_1 = 1$ et la relation de récurrence $F_{n+2} = F_{n+1} + F_n$ pour $n \geq 0$.

Les mots de Fibonacci possèdent beaucoup de propriétés remarquables. Ils constituent en particulier des cas extrémaux pour le théorème de Fine et Wilf. Soit, pour $n \geq 2$, le mot g_n obtenu en supprimant de f_n ses deux dernières lettres qui sont 01 si n est pair et 10 sinon. Les premiers mots de cette suite sont donc $g_2 = \varepsilon$, $g_3 = 0$, $g_4 = 010$ et $g_5 = 010010$. On va montrer que le mot g_n admet F_{n-1} et F_{n-2} pour périodes mais pas $F_{n-1} \wedge F_{n-2} = 1$ alors que sa longueur $|g_n|$ est égale à $F_n - 2 = F_{n-1} + F_{n-2} - (F_{n-1} \wedge F_{n-2}) - 1$. On vérifie facilement que deux nombres de Fibonacci consécutifs F_n et F_{n+1} sont premiers entre eux : $F_n \wedge F_{n+1} = 1$.

On montre par récurrence que $g_{n+2} = f_{n+1}g_n = f_n g_{n+1}$ pour tout $n \geq 2$. Les premières valeurs $g_2 = \varepsilon$, $g_3 = 0$ et $g_4 = 010$ permettent de vérifier ces égalités pour $n = 2$. La formule $f_{n+2} = f_{n+1}f_n$ et la définition de g_n donnent immédiatement l'égalité $g_{n+2} = f_{n+1}g_n$. On a ensuite $g_{n+2} = f_n f_{n-1} g_n = f_n g_{n+1}$ par hypothèse de récurrence. Les relations $g_{n+2} = f_n f_{n-1} g_n = f_n f_n g_{n-1}$ montrent que g_{n+2} admet $F_n = |f_n|$ et $F_{n+1} = |f_n f_{n-1}|$ pour périodes. Par contre, g_{n+2} n'admet pas $F_n \wedge F_{n+1} = 1$ pour période si $n \geq 2$ car il commence par le préfixe 01.

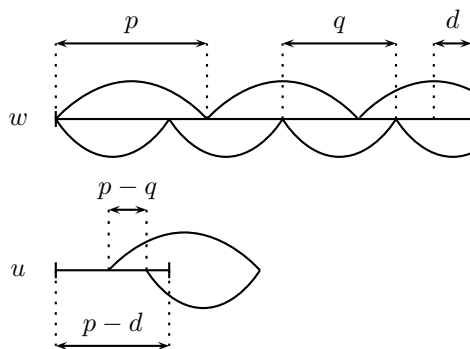


FIGURE 1.3 – Preuve du théorème de Fine et Wilf

Une preuve un peu différente est donnée en [Lot83, p. 9].

Preuve. Soit $d = p \wedge q$ le plus grand commun diviseur de p et q . On fixe d et on prouve le résultat par récurrence sur la valeur de $p + q$. Si $p + q \leq d$, c'est-à-dire $p = d$ et $q = 0$ ou $p = 0$ et $q = d$, le résultat est trivial et on suppose qu'il est vrai pour les entiers inférieurs à $p + q$. Soit w un mot tel que $|w| \geq p + q - d$ et ayant p et q comme périodes. Par symétrie, on peut supposer que $p \geq q$. Le mot w est alors factorisé $w = uv$ où u est de longueur $p - d$ (cf. figure 1.3). Pour tout $1 \leq i \leq q - d$, on a $u_i = w_i = w_{i+p} = w_{i+p-q} = u_{i+p-q}$ et u a donc $p - q$ comme période. Comme u a aussi q comme période, l'hypothèse de récurrence implique que u a $d = (p - q) \wedge q$ comme période. Comme $|u| \geq q$, le préfixe

de w de longueur q a aussi d comme période. Comme w a q comme période et que d divise q , alors w a aussi d comme période. \square

On va utiliser le théorème de Fine et Wilf pour prouver le théorème de Guibas et Odlyzko qui est un résultat assez surprenant. Il montre en effet que tout mot a les mêmes périodes qu'un mot sur un alphabet binaire. Ce théorème est contraire à l'intuition que plus de lettres dans l'alphabet donne plus de liberté pour choisir les périodes. Il est la première étape dans la caractérisation des ensembles d'entiers qui peuvent apparaître comme l'ensemble des périodes d'un mot.

Théorème 1.12 (Guibas et Odlyzko 1981). *Pour tout mot w , il existe un mot w' sur un alphabet binaire tel que $P(w) = P(w')$.*

Puisque l'ensemble des périodes d'un mot comprend sa longueur, le mot w' fourni par le théorème précédent est toujours de même longueur que le mot w . Si w est le mot 012, le mot $w' = 011$ convient puisque $P(012) = P(011) = \{3\}$.

La preuve du théorème est décomposée en plusieurs lemmes. Le premier est un corollaire immédiat du théorème de Fine et Wilf. Il donne une première idée de la structure de l'ensemble des périodes d'un mot. Il dit essentiellement que seules les grandes périodes sont irrégulières puisque les autres sont les multiples de la plus petite période. La plus petite période d'un mot w est toujours définie puisque l'ensemble $P(w)$ des périodes contient au moins la longueur de w .

Lemme 1.13. *Soit w un mot et p sa plus petite période. Si q est une période de w telle que $q \leq |w| - p$, alors q est un multiple de p .*

Preuve. Puisque $p + q \leq |w|$, l'entier $p \wedge q$ est aussi une période de w par le théorème de Fine et Wilf (théorème 1.11). Par définition de p , on a $d = p$ et p divise q . \square

Le second lemme permet de prolonger n'importe quel mot en un mot primitif. Sa preuve utilise encore le théorème de Fine et Wilf.

Lemme 1.14. *Pour tout mot w sur $\{0, 1\}$, $w0$ ou $w1$ est primitif.*

Preuve. Si $w = \varepsilon$, alors $w0$ et $w1$ sont primitifs. Supposons par l'absurde que w est non vide et que les mots $w0$ et $w1$ s'écrivent $w0 = u^k$ et $w1 = v^l$ pour des entiers $k, l \geq 2$ et des mots primitifs u et v . Les longueurs $|u|$ et $|v|$ sont des périodes de w et comme $|w| = k|u| - 1 = l|v| - 1 \geq 2 \max(|u|, |v|) - 1 \geq |u| + |v| - 1$, l'entier $d = |u| \wedge |v|$ est aussi une période de w par le théorème de Fine et Wilf (théorème 1.11). Comme d divise $|u|$ et $|v|$ et que les mots u et v sont primitifs, on a $|u| = d = |v|$. Comme u et v sont deux préfixes de w , on doit avoir $u = v$ et ceci contredit le fait que les mots u et v se terminent respectivement par les lettres 0 et 1. \square

On a vu que si p est une période du mot w , celui-ci se factorise $w = (uv)^k u$ où $|uv| = p$ et v est non vide. On applique ce résultat à la plus petite période du mot w . Dans les trois lemmes qui suivent, le mot w est factorisé de la manière suivante.

$$w = (uv)^k u \quad \text{où } p = |uv| = \min P(w), v \neq \varepsilon \text{ et } k \geq 1. \quad (1.1)$$

La preuve du théorème de Guibas et Odlyzko se fait par récurrence sur la longueur du mot w . Elle utilise la factorisation ci-dessus pour se ramener au mot uv qui est plus

petit sauf si $|w|$ est l'unique période de w (ce dernier cas est trivial). Les deux premiers lemmes traitent le cas $k \geq 2$ alors que le dernier traite le cas $k = 1$.

Lemme 1.15. *Soit w décomposé comme en (1.1) avec $k \geq 2$ et soit q tel que $|w| - p < q < |w|$. Soit $q = (k - 1)p + r$ où $|u| < r < |u| + p$. Alors q est une période de w si et seulement si r est une période de uvu .*

Preuve. Pour tout $0 < i < |w| - q = p + |u| - r$, on a $w_i = (uvu)_i$ et $w_{i+q} = (uvu)_{i+r}$. On a donc $w_i = w_{i+q}$ si et seulement si $(uvu)_i = (uvu)_{i+r}$ ce qui signifie exactement que $q \in P(w)$ si et seulement si $r \in P(uvu)$. \square

Lemme 1.16. *Soit w décomposé comme en (1.1) avec $k \geq 1$. Si $|uv| = |u'v'|$ et $P(u'v'u') = P(uvu)$, alors $P(w) = P(w')$ où $w' = (u'v')^k u'$.*

Preuve. Le cas $k = 1$ est trivial et on suppose dorénavant que $k \geq 2$. L'égalité $P(u'v'u') = P(uvu)$ implique $|uvu| = |u'v'u'|$ et donc $|w| = |w'|$. On remarque ensuite que $p = |uv|$ est une période de w' . On fixe un entier $1 \leq q \leq |w|$ et on montre que $q \in P(w)$ si et seulement si $q \in P(w')$. On considère deux cas suivant que q vérifie $q \leq |w| - p$ ou $q > |w| - p$.

On suppose d'abord que $q \leq |w| - p$. Si $q \in P(w')$, on a $|w'| = |w| \geq p + q$. D'après le théorème de Fine et Wilf (théorème 1.11), l'entier $d = p \wedge q$ est aussi une période de w' . Comme $d \leq p = |u'v'|$, d est aussi une période de $u'v'u'$ et donc de uvu . Comme d divise p , d est finalement une période de w . La minimalité de p implique $d = p$. Comme p divise q , q est une période de w . Si à l'inverse $q \in P(w)$, d'après le lemme 1.13, p divise q et q est une période de w' .

On suppose maintenant que $|w| - p < q < |w|$ et on pose $r = q - (k - 1)p$ qui vérifie $|u| < r < p + |u|$. D'après le lemme précédent, $q \in P(w)$ si et seulement si $r \in P(uvu) = P(u'v'u')$ qui est, encore d'après le lemme précédent, équivalent à $q \in P(w')$. \square

Lemme 1.17. *Soit $w = uvu$ comme en (1.1) avec $k = 1$. Soit $u' \in 0\{0, 1\}^*$ tel que $P(u') = P(u)$. Si $b \in \{0, 1\}$ est tel que $u'1^{|v|-1}b$ soit primitif alors on a $P(w) = P(w')$ où $w' = u'1^{|v|-1}bu'$.*

Preuve. Soit q une période de w qui vérifie donc $q \geq p = |u'1^{|v|-1}b|$. Elle est donc aussi période de w' puisque $P(u) = P(u')$. Ceci montre l'inclusion $P(w) \subseteq P(w')$. Supposons par l'absurde que cette inclusion soit stricte et soit q le plus petit entier de $P(w') \setminus P(w)$. Comme u' commence par un 0, q vérifie $q < |u'|$ ou $p - 1 \leq q < |w|$.

Si q vérifie $q < |u'|$, il est, par définition, la plus petite période de w' . Le lemme 1.13 implique que q divise p et que le mot $u'1^{|v|-1}b$ n'est pas primitif, ce qui est une contradiction.

Si q est égal à $p - 1$, on a alors $b = 0$ et l'égalité $u'1 = 0u'$ est à nouveau impossible. On a donc $p < q < |w|$ et on pose $r = q - p$. L'entier r est une période de u' et donc de u . Ceci implique que q est une période de w qui est en contradiction avec la définition de q . \square

On est maintenant en mesure de prouver le théorème de Guibas et Odlyzko en combinant les différents lemmes.

Preuve du théorème 1.12. On raisonne par récurrence sur la longueur de w . Le résultat est trivial si $|w| \leq 2$. On suppose le résultat vrai pour tout mot de longueur inférieure à n et soit w un mot de longueur n qui s'écrit comme en 1.1.

Si $k \geq 2$, on a $|uvu| < n$ et par hypothèse de récurrence, il existe u' et v' tels que $|wv| = |u'v'|$ et $P(uvu) = P(u'v'u')$. Le lemme 1.16 permet alors de conclure.

Si $k = 1$, on a $|u| < n$ puisque v n'est pas le mot vide. Par hypothèse de récurrence, il existe u' tel que $P(u) = P(u')$. Si $u = \varepsilon$, on a alors $P(w) = \{|w|\}$ et $w' = 01^{|w|-1}$ vérifie $P(w) = P(w')$. Sinon, on peut supposer par symétrie que u' commence par 0. D'après le lemme 1.14, il existe $b \in \{0, 1\}$ tel que $u'1^{|v|-1}b$ est primitif. Le lemme 1.17 permet alors de conclure. \square

La preuve du théorème est effective. Il est même possible d'en déduire un algorithme qui calcule en temps linéaire en la longueur de w le mot w' tel que $P(w) = P(w')$.

Exercice 1.18. Soient six mots x, y, z, x', y' et z' tels que $xyz \neq x'y'z'$. Montrer qu'il existe un entier k_0 tel que pour tout entier $k \geq k_0$, on a $xy^kz \neq x'y'^kz'$.

Solution. Si y et y' ne sont pas de même longueur, on peut supposer par symétrie que $|y| > |y'|$. Il est alors clair que $|xy^kz| > |x'y'^kz'|$ pour k assez grand et le résultat est acquis.

On suppose maintenant que y et y' sont de même longueur. Si xz et $x'z'$ ne sont pas de même longueur, l'égalité $|xy^kz| - |x'y'^kz'| = |xz| - |x'z'|$ montre que xy^kz et $x'y'^kz'$ ne sont pas de même longueur pour tout $k \geq 0$. Le résultat est alors évident.

On suppose alors que y et y' sont de même longueur et que xz et $x'z'$ sont aussi de même longueur. On peut supposer par symétrie que $|x| > |x'|$. Si $x'y'^2$ n'est pas un préfixe de xy^2 , alors xy^kz et $x'y'^kz'$ sont différents pour $k \geq 2$. Si $x'y'^2$ est un préfixe de xy^2 , il existe un mot u tel que $x = x'u$ et $uy = yu$. D'après le résultat de l'exercice 1.33, il existe un mot w et des entiers m et n tels que $v = w^m$ et $y = w^n$. Puisque $xyz \neq x'y'z'$, les mots w^mz et z' sont différents et on a le résultat.

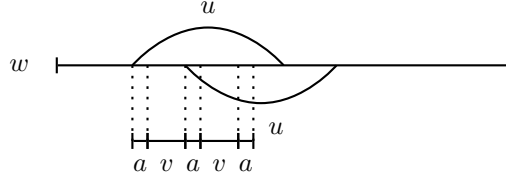
1.3.2 Mots infinis

On introduit ici les mots infinis parce qu'ils sont très utiles pour l'étude des motifs inévitables.

Un *mot infini* x sur un alphabet A est une suite infinie $x = x_0x_1x_2 \dots$ de lettres de A . L'ensemble des mots infinis sur A est noté A^ω . Un mot $x = x_0x_1x_2 \dots$ est *périodique* s'il existe un entier p appelé *période* tel que $x_n = x_{n+p}$ pour tout $n \geq 0$. Il est *ultimement périodique* s'il existe deux entiers l et p tels que $x_n = x_{n+p}$ pour tout $n \geq l$. Pour un mot fini u de longueur p , le mot infini de période p et dont les p premières lettres coïncident avec celles de u est noté u^ω . Le lemme 3.81 (p. 175) fournit une caractérisation en terme de facteurs des mots ultimement périodiques.

On décrit maintenant une façon très classique de construire des mots infinis. Un morphisme μ de A^* dans B^* est naturellement étendu aux mots infinis en posant $\mu(x) = \mu(x_0)\mu(x_1)\mu(x_2) \dots$ pour tout mot infini $x = x_0x_1x_2 \dots$ sur A . Quelques précautions sont nécessaires si le morphisme μ est effaçant car $\mu(x)$ n'est pas nécessairement infini.

Soit μ un morphisme de A^* dans A^* tel que $\mu(a)$ commence par la lettre a , c'est-à-dire $\mu(a) = au$ pour $u \in A^*$. On définit par récurrence la suite de mots finis $(w_n)_{n \geq 0}$ par $w_0 = a$ et $w_{n+1} = \mu(w_n)$. On montre par récurrence que chaque mot w_n est préfixe du

FIGURE 1.4 – Chevauchement de deux occurrences d'un facteur u

vide et v un mot quelconque. Un mot est dit *sans chevauchement* si aucun de ses facteurs n'est un chevauchement. La terminologie est justifiée par la remarque suivante. Un mot w contient un chevauchement si et seulement si un de ses facteurs a deux occurrences qui se chevauchent (cf. figure 1.4). Si w a un chevauchement $uvuvu$, le mot uvu a deux occurrences qui se chevauchent dans w . Réciproquement, soit w un mot ayant deux occurrences d'un facteur u qui se chevauchent. Soit a la première lettre de u et soit v le reste de u jusqu'au début de l'occurrence suivante (cf. figure 1.4). Le mot $avava$ est un chevauchement de w . Ceci montre en particulier qu'un mot ayant un chevauchement a toujours un chevauchement de la forme $avava$ où a est une lettre.

On commence par montrer qu'il existe des mots arbitrairement longs sans chevauchement sur l'alphabet $A = \{0, 1\}$. Soit le morphisme de Thue-Morse τ de A^* dans A^* défini de la manière suivante.

$$\tau : \begin{cases} 0 \mapsto 01 \\ 1 \mapsto 10 \end{cases}$$

Le mot infini $\tau^\omega(0)$ est appelé *mot de Thue-Morse*. On prouve sans difficulté que la n -ième lettre du mot de Thue-Morse est 0 si le nombre de 1 dans l'écriture binaire de n est pair et 1 sinon. Le début du mot de Thue-Morse est le suivant.

01101001100101101001011001101001...

Proposition 1.22. *Le mot de Thue-Morse est sans chevauchement.*

Soit X l'ensemble $(01 + 10)^*$. Par définition même du morphisme τ , tout mot $\tau^n(0)$ appartient à X pour $n \geq 1$. On commence par établir une propriété élémentaire de l'ensemble X .

Lemme 1.23. *Si x appartient à X , alors $0x0$ et $1x1$ n'appartiennent pas à X .*

Preuve. Il est clair que tout mot x de X vérifie l'égalité $|x|_0 = |x|_1$ où $|x|_a$ dénote le nombre d'occurrences de la lettre a dans x . Il s'ensuit que les mots $0x0$ et $1x1$ ne vérifient pas cette égalité et n'appartiennent pas à X . \square

Preuve de la proposition. Il suffit de montrer que les mots $\tau^n(0)$ sont sans chevauchement. On raisonne par récurrence sur n . Pour $n = 1$, le mot $\tau(0) = 01$ ne contient pas de chevauchement. Supposons par l'absurde que le mot $w = \tau^{n+1}(0)$ contienne un chevauchement. Comme cela a été vu, on peut supposer que ce chevauchement est de la forme $avava$ où $a \in \{0, 1\}$ et $v \in \{0, 1\}^*$. Le mot w se factorise donc $w = xavavay$. On peut supposer que x est de longueur paire. L'autre cas s'obtient en passant au mot miroir puisque x et y ont des longueurs de parités différentes. Si v est de longueur paire, la seconde occurrence de v commence à une position paire. On en déduit que v et ava

sont des mots de X^* et ceci est une contradiction d'après le lemme précédent. Si v est de longueur impaire, il se factorise $v = \bar{a}v'$. Comme le premier et le dernier a de $avava$ sont à des positions de même parité, la première lettre de y est aussi \bar{a} et y se factorise $y = \bar{a}y'$. Le mot w se factorise finalement $w = x\bar{a}\bar{a}v'a\bar{a}v'a\bar{a}y'$. Comme x est de longueur paire, le mot $a\bar{a}v'a\bar{a}v'a\bar{a}$ est l'image d'un facteur $auaua$ de $\tau^n(0)$. Ceci est en contradiction avec l'hypothèse de récurrence. \square

On considère le morphisme σ de $\{0, 1, 2\}$ dans $\{0, 1\}$ défini de la manière suivante.

$$\sigma : \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 01 \\ 2 \mapsto 011 \end{cases}$$

Chaque mot $\tau^n(0)$ ne contient pas de facteur 111 et commence par 0. Il appartient donc à l'ensemble $(0 + 01 + 011)^*$. On en déduit que pour tout entier n , il existe un mot w_n sur l'alphabet $\{0, 1, 2\}$ tel que $\sigma(w_n) = \tau^n(0)$. Le mot w_n est en outre unique. On note w'_n le mot obtenu en supprimant la dernière lettre de w_n . Chaque mot w'_n est alors préfixe du mot w'_{n+1} . La suite $(w_n)_{n \geq 0}$ converge vers un mot infini que l'on note $\sigma^{-1}(\tau^\omega(0))$. Le début de ce mot est le suivant.

21020121012021020120210121020120...

Théorème 1.24 (Thue 1906). *Le mot $\sigma^{-1}(\tau^\omega(0))$ est sans carré.*

Preuve. Il suffit de prouver que chacun des mots w'_n est sans carré. Supposons par l'absurde que w'_n contienne un carré uu . On peut supposer que uu a une occurrence qui n'est pas à la fin de w'_n . Sinon, on remplace w'_n par w'_{n+1} . Comme uua est un facteur de w'_n , $\sigma(u)\sigma(u)\sigma(a)$ est un facteur de $\tau^n(0)$. Les mots $\sigma(u)$ et $\sigma(a)$ commencent par 0. Le mot $\tau^n(0)$ a alors un chevauchement $avava$ contrairement au résultat de la proposition précédente. \square

Le mot infini $\sigma^{-1}(\tau^\omega(0))$ peut être directement obtenu en itérant un morphisme. Soit le morphisme μ défini de la manière suivante.

$$\mu : \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 20 \\ 2 \mapsto 210 \end{cases}$$

La suite de mots $(\mu^n(2))_{n \geq 0}$ converge vers le mot $\sigma^{-1}(\tau^\omega(0))$. En effet, on vérifie l'égalité $\tau \circ \sigma = \sigma \circ \mu$. En utilisant en outre l'égalité $\sigma(2)0 = \tau^2(0)$, on montre par récurrence que $\sigma(\mu^n(2))\tau^n(0) = \tau^{n+2}(0)$.

Soit le morphisme ν défini de la manière suivante.

$$\nu : \begin{cases} 0 \mapsto 010 \\ 1 \mapsto 011 \end{cases}$$

Soit $\nu^\omega(0)$ le point fixe de ce morphisme. Ce mot infini est sans cube. En observant les façons dont les mots de longueur 2 se découpent sur $\{010, 011\}$, on prouve facilement que si $\nu^\omega(0)$ contenait un cube, il contiendrait un cube de longueur strictement plus petite.

Les carrés et les chevauchements sont des cas particuliers des répétitions qui sont des puissances non entières de mots. Soit u un mot de longueur p et soit r un nombre rationnel de la forme $r = q/p$. On note u^r , l'unique mot fini de longueur q qui est préfixe du mot u^ω . Ainsi un carré est mot de la forme u^r avec $r = 2$ alors qu'un chevauchement est un mot de la forme u^r avec $r > 2$.

Plus généralement, un *motif* est un mot p sur un alphabet X de variables. Un mot sur un alphabet A *contient* le motif p , s'il possède un facteur de la forme $\mu(p)$ pour un morphisme $\mu : X^* \rightarrow A^*$ non effaçant. Dans le cas contraire, on dit qu'il *évite* le motif p . Les carrés correspondent au motif xx et les chevauchements $uvuvu$ correspondent au motif $xyxyx$ si v est non vide et au motif xxx si v est vide. Un motif p est dit *k-inévitable* si l'ensemble des mots sur un alphabet à k lettres qui évitent p est fini. Il est dit *inévitabile* s'il est inévitable pour tout entier k . La référence sur les motifs inévitables est [Cas02]. Le lemme suivant permet de construire des motifs inévitables.

Lemme 1.25. *Si p est un motif inévitable et si la lettre x n'apparaît pas dans p , alors le motif pxp est encore inévitable.*

Preuve. Soit A un alphabet. L'ensemble des mots sur A qui évitent p est fini. Il existe un entier l tel que tout mot de longueur l contient le motif p . Soit $n = |A|^l$ le nombre de mots sur A de longueur l . Un mot w de longueur $nl + l + n$ se factorise $w = u_0 a_1 u_1 a_2 \cdots a_n u_n$ où a_1, \dots, a_n sont des lettres et u_0, \dots, u_n sont des mots de longueur l . Par définition de n , deux mots u_i et u_j pour $i < j$ sont égaux. Le mot w se factorise donc $v_0 u_i v_1 u_i v_2$. Comme u_i est de longueur l , il se factorise $u_i = v_3 \mu(p) v_4$ où μ est une morphisme de $(X \setminus \{x\})^*$ dans A^* . En prolongeant μ à X^* par $\mu(x) = v_4 v_1 v_3$, le mot w contient le facteur $\mu(pxp)$. \square

Soit X l'alphabet infini $\{x_0, x_1, x_2, \dots\}$. On définit la suite $(z_n)_{n \geq 0}$ des mots de Zimin par $z_0 = \varepsilon$ et $z_{n+1} = z_n x_n z_n$ pour tout $n \geq 0$. Grâce au lemme précédent, chacun des mots z_n est un motif inévitable. La suite des mots de Zimin converge vers un mot infini z . Pour tout $n \geq 0$, la n -ième lettre de z est x_k si 2^k est la plus grande puissance de 2 qui divise n . Le début du mot z est le suivant.

$$z = x_0 x_1 x_0 x_2 x_0 x_1 x_0 x_3 x_0 x_1 x_0 x_2 x_0 x_1 x_0 x_4 x_0 x_1 x_0 x_2 x_0 x_1 x_0 x_3 \cdots$$

Il existe un algorithme dû à Zimin qui calcule si un motif donné est inévitable ou non. Cet algorithme est détaillé en [Cas02]. Par contre, aucun algorithme n'est connu pour déterminer si un mot est k -inévitabile pour un entier k donné.

1.3.4 Codes

Pour cette partie, on pourra consulter [BPR08]. De manière intuitive, un ensemble X de mots est un code si tout mot a au plus une décomposition en produit de mots de X .

Définition 1.26. Un ensemble $X \subset A^*$ est un *code* sur A si pour tous entiers $m, n \geq 0$ et toute suite $x_1, \dots, x_m, x'_1, \dots, x'_n$ de mots de X , l'égalité $x_1 \cdots x_m = x'_1 \cdots x'_n$ implique les égalités $m = n$ et $x_i = x'_i$ pour $1 \leq i \leq m$.

Exemple 1.27. L'ensemble $\{aa, baa, ba\}$ est un code sur l'alphabet $A = \{a, b\}$. Par contre, l'ensemble $\{a, ab, ba\}$ n'est pas un code car le mot $w = aba$ a les deux décompositions $(ab)a = a(ba)$. L'ensemble $X = a^*b$ est un code infini.

Soit X un code sur A et soit un alphabet B en bijection avec X par une fonction $\mu : B \rightarrow X$. La fonction μ se prolonge en un morphisme de B^* dans X^* en posant $\mu(b_1 \cdots b_n) = \mu(b_1) \cdots \mu(b_n)$ pour tout mot $b_1 \cdots b_n$ de B^* . La fonction ainsi prolongée est encore bijective si et seulement si X est un code. Lorsque X est un code, le monoïde X^* est alors isomorphe au monoïde libre B^* . Pour cette raison, il est qualifié de sous-monoïde libre de A^* .

Soit A un alphabet. On appelle *sous-monoïde* de A^* un ensemble M de mots tels que $\varepsilon \in M$ et $MM \subseteq M$. La dernière inclusion est en fait une égalité puisque $\varepsilon \in M$ implique l'inclusion inverse $M \subseteq MM$. Pour des définitions plus générales de monoïde et de sous-monoïde, on pourra se reporter à la section 1.11.

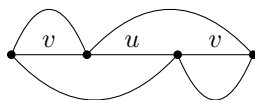


FIGURE 1.5 – Diagramme des mots uv , vu et v

Définition 1.28. Un sous-monoïde M de A^* est *libre* si pour tous mots u et v sur A , les appartenances $uv, vu, v \in M$ impliquent $u \in M$.

Le diagramme des mots uv , vu , et v est représenté à la figure 1.5. Cette définition est justifiée par la proposition suivante.

Proposition 1.29. Un sous-monoïde M de A^* est libre si et seulement si $M = X^*$ pour un code X sur A .

Lemme 1.30. Pour tout sous-monoïde M de A^* , l'ensemble $X = (M - \varepsilon) - (M - \varepsilon)^2$ est un ensemble minimal de générateurs.

Preuve. On montre facilement par récurrence sur la longueur que tout mot w de M se décompose $w = x_1 \cdots x_n$ où $x_i \in X$. Si w se décompose $w = w_1 w_2$ où $w_1, w_2 \in M - \varepsilon$, on applique l'hypothèse de récurrence à w_1 et w_2 pour obtenir une décomposition de w . Si w n'a pas décomposition $w = w_1 w_2$, il appartient à X .

Réciproquement, soit un ensemble Y tel que $M = Y^*$. Il est clair que tout mot x de X appartient à Y . Sinon x s'écrit $x = y_1 \cdots y_n$ avec $n \geq 2$ ce qui contredit le fait que x est indécomposable dans M . \square

Preuve de la proposition. Supposons d'abord que $M = X^*$ où X est un code. Soient u et v des mots tels que $uv, vu, v \in M$. Les mots uv , vu et v se décomposent en produits de mots de X . On considère alors le mot $w = vuv$ de M . Pour que ce mot ait une seule décomposition, il faut que la décomposition de uv soit faite d'une décomposition de u suivie de celle de v .

Réciproquement soit M un sous-monoïde libre de A^* . Soit X l'ensemble $X = (M - \varepsilon) - (M - \varepsilon)^2$. D'après le lemme précédent, on a l'égalité $M = X^*$. Supposons par l'absurde que X ne soit pas un code et soit z un mot de longueur minimale ayant une double factorisation $z = x_1 \cdots x_m = x'_1 \cdots x'_n$. Par définition de z , on a $x_1 \neq x'_1$ et par symétrie on peut supposer que $x'_1 = x_1 u$ pour un mot $u \neq \varepsilon$. On pose alors $v = x'_2 \cdots x'_n x_1$ et on vérifie que $vu = x'_2 \cdots x'_n x'_1$, $uv = x_2 \cdots x_n x_1$ et que $uv, vu, v \in M$. Puisque M est libre, le mot u appartient à M et ceci contredit la minimalité de z . \square

La définition d'un sous-monoïde libre entraîne qu'une intersection quelconque de sous-monoïdes libres est encore un sous-monoïde libre. Il s'ensuit que pour tout langage L de A^* , il existe un plus petit sous-monoïde libre contenant L , appelé *enveloppe libre*.

Proposition 1.31. *Soit X un ensemble fini de mots. Si X n'est pas un code, l'enveloppe libre de X est engendrée par un ensemble Y tel que $|Y| \leq |X| - 1$.*

Preuve. Soit M l'enveloppe libre de X et soit Y le code tel que $M = Y^*$. Tout mot x de X se décompose $x = y_1 \cdots y_n$ avec $y_1, \dots, y_n \in Y$. On définit la fonction $f : X \rightarrow Y$ en posant $f(x) = y_1$ pour tout mot $x \in X$. On montre que la fonction f est surjective mais pas injective. Supposons par l'absurde que f n'est pas surjective. Si y n'appartient pas à $f(X)$, alors on a l'inclusion $X \subseteq \varepsilon + (Y - y)Y^*$. On pose alors $Z = (Y - y)y^*$ et on montre facilement que $X \subseteq Z^* = \varepsilon + (Y - y)Y^*$ ce qui contredit le fait que Y^* est l'enveloppe libre de X . Si X n'est pas un code, il existe une relation $z = x_1 \cdots x_m = x'_1 \cdots x'_n$ avec $x_1 \neq x'_1$. On vérifie que $f(x_1) = f(x'_1)$ car le mot z a une unique factorisation. \square

Le corollaire suivant décrit les codes à deux mots. L'équivalence entre la deuxième et la troisième proposition peut aussi se montrer directement par récurrence sur la longueur de uv .

Corollaire 1.32. *Pour deux mots u et v , les propositions suivantes sont équivalentes.*

- L'ensemble $\{u, v\}$ n'est pas un code.
- Il existe un mot w tel que $u, v \in w^*$.
- Les mots u et v vérifient $uv = vu$.

Exercice 1.33. Montrer directement que deux mots u et v vérifient l'égalité $uv = vu$ si et seulement ils sont des puissances d'un même mot w , *i.e.* $u, v \in w^*$.

Solution. Il est clair que si $u = w^m$ et $v = w^n$, alors $uv = vu = w^{m+n}$. On montre la réciproque par récurrence sur $|u| + |v|$. Si $|u| + |v| = 0$, alors les deux mots u et v sont vides et le résultat est trivial. On suppose donc que $|u| + |v| > 0$. Si $|u| = |v|$, l'égalité $uv = vu$ implique immédiatement que $u = v$ et le résultat est acquis en prenant $w = u = v$. Sinon, on peut supposer par symétrie que $|u| < |v|$. L'égalité $uv = vu$ implique alors que u est un préfixe de v et que $v = uu'$ pour un certain mot u' . L'égalité $uv = vu$ s'écrit alors $uuu' = uu'u$ d'où on tire $uu' = u'u$ en simplifiant à gauche par u . Par hypothèse de récurrence, il existe un mot w tel que $u = w^m$ et $u' = w^n$ et donc $v = uu' = w^{m+n}$.

1.4 Un peu d'ordre

Dans cette partie, on s'intéresse aux bons quasi-ordres et aux théorèmes de Higman et de Kruskal. Ces ordres sont souvent utilisés pour prouver la terminaison de systèmes de réécriture ou de procédures. On rappelle aussi quelques ordres classiques sur l'ensemble des mots.

Une relation binaire \preceq sur un ensemble E est un *quasi-ordre* (on dit aussi *préordre*) si elle est réflexive et transitive. Contrairement à un ordre, un quasi-ordre n'est pas nécessairement antisymétrique. On peut très bien avoir $x \preceq y$ et $y \preceq x$ pour deux éléments distincts x et y . On écrit $y \succeq x$ pour $x \preceq y$. On écrit $x \prec y$ si $x \preceq y$ et $y \not\preceq x$. À un quasi-ordre \preceq sur E est associée une relation d'équivalence \approx définie par $x \approx y$

si $x \preceq y$ et $y \preceq x$. Le quasi-ordre \preceq induit alors un ordre sur l'ensemble quotient E/\approx . Deux éléments x et y sont dits *incomparables* (pour \preceq) si $x \not\preceq y$ et $y \not\preceq x$.

Exemple 1.34. Soit $E = \mathbb{N}$ l'ensemble des entiers naturels. La relation de divisibilité est un ordre sur \mathbb{N} . Soit la relation \preceq définie sur \mathbb{N} par $m \preceq n$ s'il existe un entier k tel que m divise n^k , c'est-à-dire si tout diviseur premier de m est aussi diviseur premier de n . C'est un quasi-ordre qui n'est pas un ordre. On a en effet $6 \preceq 12$ et $12 \preceq 6$ car 6 et 12 ont les mêmes diviseurs premiers.

Une relation d'équivalence \sim sur un ensemble E est un quasi-ordre sur E dont la relation d'équivalence associée \approx est justement la relation \sim .

Soit \preceq un quasi-ordre sur E . Une *chaîne décroissante* est une suite $(x_i)_{i \geq 0}$ finie ou infinie telle que $x_i \succ x_{i+1}$ pour tout $i \geq 0$. Un quasi-ordre est dit *bien fondé* s'il ne possède pas de chaîne infinie décroissante. Autrement dit, le quasi-ordre \preceq est bien fondé si la relation \succ induite sur E/\approx est noethérienne (cf. section 2.7.1). Une *antichaîne* est un ensemble d'éléments incomparables deux à deux. Un *idéal* (d'ordre) est un ensemble I tel que si x appartient à I et $x \preceq y$, alors y appartient aussi à I . Une *base* d'un idéal I est un sous-ensemble B de I tel que $I = \{x \mid \exists b \in B \ b \preceq x\}$. On dit alors que l'idéal I est *engendré* par la base B . Le théorème suivant donne plusieurs propriétés équivalentes qui définissent la notion fondamentale de bon quasi-ordre.

Théorème 1.35 (Higman). *Les propriétés suivantes sont équivalentes pour un quasi-ordre sur E .*

- i) *tout idéal possède une base finie,*
- ii) *toute chaîne croissante $I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots$ d'idéaux est constante à partir d'un certain rang,*
- iii) *toute suite infinie d'éléments de E contient une sous-suite infinie croissante,*
- iv) *toute suite infinie d'éléments de E contient une sous-suite croissante de longueur 2,*
- v) *toute chaîne décroissante est finie et toute antichaîne est finie,*
- vi) *tout quasi-ordre \preceq' qui prolonge \preceq est bien fondé.*

Un quasi-ordre qui satisfait l'une de ces propriétés est appelé bon quasi-ordre.

Il y a une différence de terminologie entre le français et l'anglais. En français, un ordre est implicitement total et le qualificatif *partiel* est ajouté s'il ne l'est pas. En anglais, un ordre est supposé partiel et on utilise l'adjectif *total* ou *linear* pour préciser qu'il est total. En anglais, un bon quasi-ordre est appelé *well quasi-order* souvent abrégé en *wqo*.

On a vu qu'une relation d'équivalence est un quasi-ordre. Elle est un bon quasi-ordre si et seulement si elle a un nombre fini de classes. En effet, deux éléments non équivalents sont incomparables. Pour qu'il n'existe pas d'antichaîne infinie, le nombre de classes doit être fini. La réciproque est évidente.

Exemple 1.36. L'ordre naturel des entiers naturels est un bon quasi-ordre sur \mathbb{N} . L'ordre naturel des entiers relatifs n'est pas bien fondé sur \mathbb{Z} et il n'est donc pas un bon quasi-ordre. La divisibilité est un ordre bien fondé sur les entiers mais elle n'est pas un bon quasi-ordre car les nombres premiers forment une antichaîne infinie.

Preuve. On commence par montrer l'équivalence entre i) et ii). Soit $I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots$ une chaîne croissante d'idéaux. L'ensemble $I = \bigcup_{n \geq 0} I_n$ est encore un idéal. Si le quasi-ordre a la propriété de base finie, cet idéal est engendré par une base finie B . Il existe

alors un entier k tel que $B \subseteq I_k$. On a alors $I = I_k$ et $I_{k+n} = I_k$ pour tout $n \geq 0$. Réciproquement soit I un idéal. Si I ne possède pas de base finie, on construit par récurrence une suite $(x_n)_{n \geq 0}$ d'éléments de I de la manière suivante. L'élément x_0 est choisi de façon arbitraire dans I . Les éléments x_0, \dots, x_k étant déjà choisis, l'élément x_{k+1} est choisi en dehors de l'idéal engendré par $B_k = \{x_0, \dots, x_k\}$. Soit I_k l'idéal engendré par B_k . Par construction, la chaîne d'idéaux $I_0 \subsetneq I_1 \subsetneq I_2 \subsetneq \dots$ est strictement croissante.

On montre maintenant l'équivalence entre iii), iv) et v). Il est clair que iii) implique iv) et que iv) implique à nouveau v). Nous montrons d'abord que v) implique iv) puis que iv) implique iii). Soit $(x_n)_{n \geq 0}$ une suite d'éléments et soit $X = \{x_i \mid \forall j > i \ x_i \not\leq x_j \text{ et } x_i \not\geq x_j\}$. Puisque les antichaînes sont finies, l'ensemble X est fini. Pour tout x_i n'appartenant pas à X , il existe x_j avec $j > i$ tel que $x_i \leq x_j$ ou $x_i \geq x_j$. Si la première relation est vraie pour au moins une paire (i, j) , on a trouvé une sous-suite croissante de longueur 2. Dans le cas contraire, on peut facilement extraire une sous-suite infinie décroissante, ce qui contredirait l'hypothèse. Nous montrons maintenant que iv) implique iii). Soit $(x_n)_{n \geq 0}$ une suite d'éléments et soit $X = \{x_i \mid \forall j > i \ x_i \not\leq x_j\}$. D'après l'hypothèse, l'ensemble X est fini. Pour tout x_i n'appartenant pas à X , il existe x_j avec $j > i$ tel que $x_i \leq x_j$. Il est alors facile d'extraire une sous-suite infinie croissante.

On montre ensuite l'équivalence entre i)-ii) et iii)-iv)-v). Soit $(x_n)_{n \geq 0}$ une suite d'éléments. Soit I_k l'idéal engendré par la base $B_k = \{x_0, \dots, x_k\}$. Puisque la chaîne d'idéaux $I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots$ est constante à partir d'un certain rang, il existe deux indices $k < l$ tel que $x_k \leq x_l$, ce qui montre iv). Réciproquement, soit $I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots$ une chaîne d'idéaux. Si cette chaîne n'est pas constante à partir d'un certain rang, on peut supposer, quitte à supprimer quelques idéaux de la chaîne, que chaque inclusion $I_n \subsetneq I_{n+1}$ est stricte. Soit x_n un élément de $I_{n+1} \setminus I_n$. La suite $(x_n)_{n \geq 0}$ ne contient aucune sous-suite croissante de longueur 2. Ceci contredit l'hypothèse que la chaîne d'idéaux n'est pas constante à partir d'un certain rang.

On montre finalement l'équivalence entre vi) et les autres propriétés. Soit \preceq' un quasi-ordre qui prolonge \preceq . Soit $(x_n)_{n \geq 0}$ une suite d'éléments. D'après iv), il existe deux indices $k < l$ tels que $x_k \leq x_l$ et donc $x_k \preceq' x_l$. Ceci montre que le quasi-ordre \preceq' n'a pas de chaîne infinie décroissante. Réciproquement, le quasi-ordre \preceq est son propre prolongement. Ceci montre qu'il est bien fondé. Il reste à montrer qu'il n'a pas d'antichaîne infinie. On montre que si A est une antichaîne de \preceq et si \preceq_0 est quasi-ordre quelconque sur A , il est possible de prolonger \preceq en un quasi-ordre \preceq' tel que la restriction de \preceq' à A est \preceq_0 . Soit \preceq' la clôture transitive de la relation $\preceq \cup \preceq_0$. C'est bien sûr un quasi-ordre. Il reste à prouver que pour tous éléments a et b de A , $a \preceq' b$ implique $a \preceq_0 b$ puisque la réciproque est évidente. Si $a \preceq' b$, il existe une suite finie x_0, \dots, x_n telle que $x_0 = a$, $x_n = b$ et $x_i \leq x_{i+1}$ ou $x_i \preceq_0 x_{i+1}$ pour tout $0 \leq i \leq n-1$. Si $n = 1$, le résultat est acquis puisque a et b sont incomparables pour \preceq . On suppose maintenant que $n \geq 2$. En utilisant la transitivité de \preceq_0 , on peut supposer que les éléments x_1, \dots, x_{n-1} n'appartiennent pas à A . On a alors $x_i \leq x_{i+1}$ pour tout $0 \leq i \leq n-1$ et donc $a \leq b$ contrairement à l'hypothèse que a et b sont incomparables pour \preceq . Ceci montre que \preceq' coïncide avec \preceq_0 sur A . Si le quasi-ordre \preceq a une antichaîne infinie A , on peut le prolonger en utilisant un quasi-ordre \preceq_0 sur A qui contient une chaîne infinie décroissante. Le quasi-ordre obtenu contient encore une chaîne infinie décroissante et n'est pas bien fondé. \square

Soient \preceq_1 et \preceq_2 deux quasi-ordres sur des ensembles E_1 et E_2 . Le produit cartésien

$E_1 \times E_2$ est naturellement muni d'un quasi-ordre \preceq défini par $(x_1, x_2) \preceq (y_1, y_2)$ si $x_1 \preceq_1 y_1$ et $x_2 \preceq_2 y_2$. Le résultat suivant est parfois attribué à Nash-Williams.

Lemme 1.37. *Si \preceq_1 et \preceq_2 sont des bons quasi-ordres sur E_1 et E_2 , alors \preceq est un bon quasi-ordre sur $E_1 \times E_2$.*

La preuve découle directement de la caractérisation iii) du théorème précédent. L'ordre naturel sur les entiers induit un ordre naturel sur les k -uplets d'entiers défini par $(m_1, \dots, m_k) \leq (n_1, \dots, n_k)$ si $m_i \leq n_i$ pour tout $1 \leq i \leq k$. Le lemme précédent a pour conséquence le résultat suivant appelé lemme de Dickson.

Lemme 1.38 (Dickson 1913). *Tout sous-ensemble de \mathbb{N}^k a un nombre fini d'éléments minimaux.*

1.4.1 Quasi-ordres sur les mots

Soit \preceq un quasi-ordre sur un ensemble A . On définit un quasi-ordre \preceq^* sur l'ensemble A^* des mots finis sur A de la façon suivante. Deux mots $w = a_1 \cdots a_m$ et $w' = a'_1 \cdots a'_n$ vérifient $w \preceq^* w'$ s'il existe une suite croissante $1 \leq j_1 < j_2 < \cdots < j_m \leq n$ d'indices tels que $a_i \preceq a'_{j_i}$ pour tout $1 \leq i \leq m$. Pour une relation R , on note parfois R^* la clôture réflexive et transitive de la relation R (cf. section 2.7.1). Ici le quasi-ordre \preceq^* n'est pas du tout la clôture réflexive et transitive de \preceq (qui est justement égale à \preceq). L'exposant $*$ rappelle seulement que \preceq^* est l'extension à A^* du quasi-ordre \preceq . On vérifie facilement que \preceq^* est le plus petit (au sens de l'inclusion) quasi-ordre sur A^* tel que :

1. $uv \preceq^* uav$ pour tous mots $u, v \in A^*$ et toute lettre $a \in A$,
2. $uav \preceq^* ubv$ si $a \preceq b$ pour tous mots $u, v \in A^*$ et toutes lettres $a, b \in A$.

Soit $w = w_1 \cdots w_n$ un mot de longueur n . Un *sous-mot* de w est un mot u tel qu'il existe une sous-suite croissante $1 \leq i_1 < \cdots < i_k \leq n$ d'indices vérifiant $u = w_{i_1} \cdots w_{i_k}$. Autrement dit, le mot u est obtenu en supprimant certaines lettres du mot w . La relation *être sous-mot* est bien sûr un ordre partiel sur l'ensemble des mots. La notion de sous-mot est à distinguer de celle de facteur où les lettres choisies sont consécutives.

Exemple 1.39. Si le quasi-ordre \preceq sur A est l'égalité, le quasi-ordre \preceq^* est l'ordre des sous-mots.

Théorème 1.40 (Higman 1952). *Si \preceq est un bon quasi-ordre sur A , alors \preceq^* est un bon quasi-ordre sur A^* .*

Preuve. On dit qu'une suite $(u_n)_{n \geq 0}$ est *mauvaise* s'il n'existe pas deux indices $k < l$ tels que $u_k \preceq^* u_l$ et on utilise la caractérisation iv) du théorème 1.35. On raisonne par l'absurde et on suppose qu'il existe au moins une mauvaise suite. On construit alors par récurrence une mauvaise suite minimale $(u_n)_{n \geq 0}$. Soit u_0 un mot le plus court possible tel que u_0 soit le premier élément d'une mauvaise suite. Les mots u_0, \dots, u_n étant choisis, u_{n+1} est un mot le plus court tel que u_0, \dots, u_{n+1} sont les premiers éléments d'une mauvaise suite. La suite ainsi construite est mauvaise. Puisque le mot vide vérifie $\varepsilon \preceq^* w$ pour tout mot w , aucun des mots u_n n'est vide. Chaque mot u_n s'écrit donc $u_n = a_n v_n$ où a_n est une lettre et v_n un mot. Comme \preceq est un bon quasi-ordre, il existe une suite extraite $(a_{i_n})_{n \geq 0}$ qui est croissante. On considère alors la suite suivante

$$u_0, u_1, u_2, \dots, u_{i_0-2}, u_{i_0-1}, v_{i_0}, v_{i_1}, v_{i_2}, \dots$$

obtenue en prenant les i_0 premiers éléments $u_0, u_1, \dots, u_{i_0-1}$ de la suite $(u_n)_{n \geq 0}$ puis tous les éléments de la suite $(v_{i_n})_{n \geq 0}$. Cette nouvelle suite est encore mauvaise. On ne peut pas avoir $k < l < i_0$ et $u_k \preceq^* u_l$, car $(u_n)_{n \geq 0}$ est mauvaise. La relation $u_k \preceq^* v_{i_l}$ implique la relation $u_k \preceq^* u_{i_l}$ et la relation $v_{i_k} \preceq^* v_{i_l}$ implique $u_{i_k} \preceq^* u_{i_l}$ car $a_{i_k} \preceq a_{i_l}$. On obtient alors une contradiction car v_{i_0} est plus court que u_{i_0} . \square

Exemple 1.41. D'après le théorème de Higman, l'ordre des sous-mots sur un alphabet fini est un bon quasi-ordre. En effet, l'égalité sur A est bien sûr un bon quasi-ordre sur A .

Le quasi-ordre \preceq^* défini sur les suites finies peut être étendu en un quasi-ordre \preceq^ω sur A^ω en posant $(a_n)_{n \geq 0} \preceq^\omega (a'_n)_{n \geq 0}$ s'il existe une suite strictement croissante d'indices $(i_n)_{n \geq 0}$ telle que $a_n \preceq a'_{i_n}$ pour tout $n \geq 0$. L'exemple suivant montre que le théorème de Higman ne se généralise pas à \preceq^ω .

Exemple 1.42. Soit $A = \mathbb{N} \times \mathbb{N}$ l'ensemble des paires d'entiers et soit l'ordre \preceq défini sur A par $(m, n) \preceq (m', n')$ si $m = m'$ et $n \leq n'$ ou $m < m'$ et $n < m'$. On vérifie que c'est un bon ordre partiel. Pour chaque entier i , soit $x_i \in A^\omega$ donné par

$$x_i = (i, 0), (i, 1), (i, 2), (i, 3), \dots$$

La suite $(x_i)_{i \geq 0}$ ne contient pas de sous-suite croissante de longueur 2 pour \preceq^ω .

Pour obtenir une généralisation du théorème de Higman aux suites infinies, des meilleurs quasi-ordres (better quasi-order en anglais souvent abrégé en *bqo*) ont été introduits par Nash-Williams. Pour une définition des meilleurs quasi-ordre et quelques résultats, on pourra consulter [Alm94, p. 33].

Exemple 1.43. L'ordre des sous-mots est défini sur A^ω par $(a_n)_{n \geq 0} \preceq^\omega (a'_n)_{n \geq 0}$ s'il existe une suite strictement croissante d'indices $(i_n)_{n \geq 0}$ telle que $a_n = a'_{i_n}$ pour tout $n \geq 0$. Si l'alphabet A est fini, c'est un bon quasi-ordre sur A^ω . Ceci provient du fait que l'égalité est un meilleur quasi-ordre sur A .

1.4.2 Ordres sur les mots

L'ordre des sous-mots n'est pas l'ordre le plus naturel sur les mots. Le plus familier est sans aucun doute l'*ordre lexicographique* utilisé dans le dictionnaire. On suppose que A est muni d'un ordre noté $<$. Deux mots w et w' vérifient $w <_{\text{lex}} w'$ soit si w est préfixe de w' soit s'il existe deux lettres a et b vérifiant $a < b$ et trois mots u, v et v' tels que $w = uav$ et $w' = ubv'$. Le mot u est en fait le plus long préfixe commun à w et w' qui est noté $w \wedge w'$. Dans le cas où w est préfixe de w' , on a $w \wedge w' = w$.

On pourra remarquer que la fonction d définie par $d(w, w') = |w| + |w'| - 2|w \wedge w'|$ est une distance sur l'ensemble des mots.

L'ordre lexicographique est total mais il n'est pas bien fondé dès que l'alphabet a au moins deux lettres. Sur l'alphabet $A = \{0, 1\}$, la suite de mots $w_n = 0^n 1$ est infinie et décroissante. Pour cette raison, on considère aussi l'*ordre hiérarchique* où les mots sont d'abord classés par longueur puis par ordre lexicographique. Deux mots w et w' vérifient $w <_{\text{hié}} w'$ si $|w| < |w'|$ ou $|w| = |w'|$ et $w <_{\text{lex}} w'$. Cet ordre a l'avantage d'être total et bien fondé.

1.4.3 Quasi-ordres sur les arbres

Le théorème de Higman admet une généralisation aux arbres finis due à Kruskal. La construction de bons quasi-ordres sur les arbres est essentielle pour la terminaison de systèmes de réécriture de termes, souvent utilisés pour donner une sémantique aux langages de programmation fonctionnels.

On commence par rappeler la notion de domaine d'arbre. On note \mathbb{N}^* l'ensemble des suites finies d'entiers. Un *domaine d'arbre* est sous-ensemble D de \mathbb{N}^* qui vérifie les deux conditions suivantes.

1. D est clos par préfixe, c'est-à-dire si uv appartient à D , alors u appartient aussi à D pour toutes suites $u, v \in \mathbb{N}^*$.
2. D est clos par valeurs inférieures, c'est-à-dire si ui appartient à D , alors uj appartient aussi à D pour tout $u \in \mathbb{N}^*$ et tous entiers $0 \leq j \leq i$.

Soit A un alphabet non nécessairement fini. On appelle *arbre sur A* une fonction $t : D \rightarrow A$ où D est un domaine d'arbre. Un élément du domaine D de l'arbre est appelé un *nœud* de l'arbre. L'arbre est dit *vide* (resp. *fini*) si son domaine D est vide (resp. fini). On appelle *taille* de t et on note $|t|$ le cardinal du domaine de t . Le domaine d'un arbre t est noté $\text{dom}(t)$. Un arbre est représenté comme un graphe ayant les nœuds pour sommets et les paires de nœuds de la forme (u, ui) pour arêtes.

Un mot fini $w = a_0 \cdots a_n$ s'identifie avec un arbre t de domaine $D = \{\varepsilon, 0, 0^2, \dots, 0^n\}$, c'est-à-dire $D = (\varepsilon + 0)^n$, et défini par $t(0^k) = a_k$ pour tout $0 \leq k \leq n$.

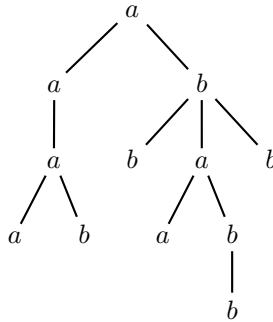


FIGURE 1.6 – L'arbre de l'exemple 1.44

Exemple 1.44. Soit le domaine d'arbre D égal à l'ensemble

$$D = \{\varepsilon, 0, 1, 00, 10, 11, 12, 000, 001, 110, 111, 1110\}$$

et soit l'arbre t défini sur ce domaine par $t(w) = a$ si w a un nombre pair de 1 et $t(w) = b$ sinon. Cet arbre t est représenté à la figure 1.6.

Soit t un arbre de domaine D et soit w un mot sur \mathbb{N} . L'arbre $w^{-1}t$ est l'arbre de domaine $w^{-1}D$ défini par $(w^{-1}t)(u) = t(wu)$ pour tout $u \in w^{-1}D$. Cet arbre est appelé le *sous-arbre de t enraciné en w* . Si w n'appartient pas au domaine de t , l'arbre $w^{-1}D$ est vide.

Soit \preceq un quasi-ordre sur un ensemble A . On définit un quasi-ordre \preceq^Δ sur l'ensemble de tous les arbres finis sur A de la façon suivante. Deux arbres t et t' de domaines

respectifs D et D' vérifient $t \preceq^\Delta t'$ s'il existe une fonction injective f de D dans D' telle que pour tous w et w' dans D , on ait les propriétés suivantes.

1. $t(w) \preceq t'(f(w))$,
2. $f(w \wedge w') = f(w) \wedge f(w')$ et
3. Si $w <_{\text{lex}} w'$ alors $f(w) <_{\text{lex}} f(w')$

où $w \wedge w'$ dénote le plus long préfixe commun de w et w' .

La fonction f généralise la suite croissante d'indices j_1, \dots, j_m utilisée dans le cas des mots pour définir l'ordre \preceq^* . La première condition vérifiée par f est l'analogie de la condition $a_i \preceq a'_{j_i}$ pour les mots. Les deux autres conditions assurent que f est bien un plongement de l'arbre t dans l'arbre t' . La seconde condition exprime que la relation de parenté est préservée et la dernière que l'ordre des fils est également respecté.

Ce quasi-ordre peut être aussi défini de manière récursive sur les arbres.

Théorème 1.45 (Kruskal 1960). *Si \preceq est un bon quasi-ordre sur A , alors \preceq^Δ est un bon quasi-ordre sur l'ensemble des arbres finis sur A .*

Dans les arbres que nous avons définis, les fils de chaque nœud sont ordonnés et ils forment une suite. Il est possible de définir des arbres où l'ordre des fils est sans importance. Le théorème de Kruskal reste vrai pour cette variante des arbres.

Preuve. La preuve reprend les grandes lignes de la preuve du théorème de Higman. On dit qu'une suite $(t_n)_{n \geq 0}$ est *mauvaise* s'il n'existe pas deux indices $k < l$ tels que $t_k \preceq^\Delta t_l$ et on utilise la caractérisation iv) du théorème 1.35. On raisonne par l'absurde et on suppose qu'il existe au moins une mauvaise suite. On construit alors par récurrence une mauvaise suite minimale $(t_n)_{n \geq 0}$. Soit t_0 un arbre de taille minimale tel que t_0 soit le premier élément d'une mauvaise suite. Les arbres t_0, \dots, t_n étant choisis, t_{n+1} est un arbre de taille minimale tel que t_0, \dots, t_{n+1} sont les premiers éléments d'une mauvaise suite. La suite ainsi construite est mauvaise.

Il existe un nombre fini d'indices n tels que $|t_n| \leq 2$. Sinon il existe une sous-suite infinie d'arbres de taille 1 et on peut trouver deux indices $k < l$ tels que $t_k(\varepsilon) \preceq t_l(\varepsilon)$ qui est équivalent à $t_k \preceq^\Delta t_l$ si t_k et t_l sont de taille 1. Quitte à supprimer un nombre fini d'éléments de la suite, on peut supposer que $|t_n| \geq 2$ pour tout $n \geq 0$.

Puisque \preceq est un bon quasi-ordre sur A , il existe une suite extraite $(t_{i_n})_{n \geq 0}$ telle que $t_{i_0}(\varepsilon) \preceq t_{i_1}(\varepsilon) \preceq t_{i_2}(\varepsilon) \preceq \dots$. Quitte à remplacer la suite $(t_n)_{n \geq 0}$ par la suite extraite $(t_{i_n})_{n \geq 0}$, on peut supposer que la suite $(t_n)_{n \geq 0}$ vérifie $t_0(\varepsilon) \preceq t_1(\varepsilon) \preceq t_2(\varepsilon) \preceq \dots$.

Soit \mathcal{D} l'ensemble des arbres qui apparaissent juste sous la racine d'un des arbres t_n . Plus formellement, on définit l'ensemble \mathcal{D} par

$$\mathcal{D} = \{i^{-1}t_n \mid i \in \mathbb{N} \text{ et } i \in \text{dom}(t_n)\}.$$

On prétend que \preceq^Δ est un bon quasi-ordre sur \mathcal{D} . Sinon, il existe une mauvaise suite $(r_n)_{n \geq 0}$ d'arbres de \mathcal{D} . On peut extraire de cette suite une nouvelle suite $(r'_n)_{n \geq 0}$ telle que, pour tous $k < l$, les arbres r'_k et r'_l sont des sous-arbres de t_m et t_n avec $m < n$. On peut donc supposer que la suite $(r_n)_{n \geq 0}$ satisfait cette propriété. Soit alors n_0 le plus petit indice tel que r_0 est un sous-arbre de t_{n_0} . On considère la suite

$$t_0, t_1, \dots, t_{n_0-1}, r_0, r_1, r_2, \dots$$

et on constate que cette suite est encore mauvaise et qu'elle contredit la minimalité de la suite $(t_n)_{n \geq 0}$ construite. Cette contradiction prouve que \preceq^Δ est bien un bon quasi-ordre sur \mathcal{D} . Par le théorème de Higman, le quasi-ordre \preceq^{Δ^*} est un bon quasi-ordre sur \mathcal{D}^* .

Pour chaque arbre t_n , on note k_n l'entier $\max\{i \mid i \in \text{dom}(t_n)\}$. Les sous-arbres sous la racine de t_n sont donc les arbres $0^{-1}t_n, 1^{-1}t_n, \dots, k_n^{-1}t_n$. Puisque \preceq^{Δ^*} est un bon quasi-ordre sur \mathcal{D}^* , il existe deux indices $m < n$ tels que

$$(0^{-1}t_m, 1^{-1}t_m, \dots, k_m^{-1}t_m) \preceq^{\Delta^*} (0^{-1}t_n, 1^{-1}t_n, \dots, k_n^{-1}t_n)$$

Puisqu'on a en outre $t_m(\varepsilon) \preceq t_n(\varepsilon)$, on a finalement $t_m \preceq^\Delta t_n$ qui prouve que \preceq^Δ est un bon quasi-ordre sur l'ensemble de tous les arbres finis. \square

Les théorèmes de Higman et de Kruskal peuvent encore être généralisés aux graphes. Le théorème de Robertson et Seymour établit que l'ordre par mineur des graphes est un bon quasi-ordre. On rappelle qu'un *mineur* d'un graphe G est un graphe obtenu par contraction d'arêtes d'un sous-graphe de G . Un *sous-graphe* est le graphe induit par un sous-ensemble de sommets. La contraction d'une arête consiste à identifier les deux sommets qu'elle relie. Beaucoup de classes de graphes comme celle des graphes planaires ou celle des graphes triangulés sont caractérisées par des mineurs interdits. Elles sont égales à l'ensemble des graphes qui ne contiennent pas certains graphes fixés comme mineur. La preuve du résultat de Robertson et Seymour est excessivement difficile et s'étend sur plusieurs centaines de pages.

L'exercice suivant introduit une extension du théorème de Ramsey (théorème 1.107) aux arbres.

Exercice 1.46. Pour $n \geq 0$, on note B_n l'ensemble $\{0,1\}^{\leq n}$ des mots de longueur au plus n sur l'alphabet $\{0,1\}$. Un arbre t est dit *monochrome* si pour tous $w, w' \in \text{dom}(t)$, on a $t(w) = t(w')$. Montrer que pour tout arbre t de domaine B_{2n} sur l'alphabet $A = \{a,b\}$, il existe un arbre t' monochrome de domaine B_n tel que $t' \preceq^\Delta t$ où l'ordre \preceq sur A est l'égalité.

1.5 Langages rationnels

La classe des langages rationnels est le premier niveau de la hiérarchie de Chomsky. Ces langages sont très simples mais ils possèdent de très nombreuses propriétés remarquables. Ils peuvent être introduits par des définitions de natures très différentes. Ceci est une preuve de leur rôle central en théorie des langages formels.

1.5.1 Expressions rationnelles

On commence par la définition de ces langages qui utilise les opérations rationnelles et justifie la terminologie.

Définition 1.47 (Langages rationnels). La classe \mathcal{R} des *langages rationnels* (sur A) est la plus petite famille de langages telle que :

- i) $\emptyset \in \mathcal{R}$ et $\{a\} \in \mathcal{R}$ pour toute lettre a ;
- ii) \mathcal{R} est close pour les opérations rationnelles (l'union, le produit et l'étoile).

Exemple 1.48. Quelques exemples de langages rationnels.

- Le langage $\{\varepsilon\}$ est rationnel car il s'écrit \varnothing^* .
- Le langage A est rationnel puisqu'il s'écrit $A = \bigcup_{a \in A} \{a\}$.
- Le langage L des mots de longueur paire est rationnel puisqu'il s'écrit $L = (AA)^* = (A^2)^*$.
- Le langage L' des mots de longueur impaire est rationnel puisqu'il s'écrit $L' = AL$.
- Le langage des mots qui contiennent un facteur aba est rationnel puisqu'il s'écrit A^*abaA^* .

La notion d'*expression* n'est pas formellement définie. Elle doit être entendue dans le sens usuel en mathématiques comme c'est le cas des expressions arithmétiques. Les opérateurs arithmétiques sont remplacés dans notre cadre par les opérateurs rationnels et les parenthèses sont encore utilisées pour lever les ambiguïtés.

Définition 1.49 (Expressions rationnelles). La classe \mathcal{E} des *expressions rationnelles* est la plus petite famille d'expressions telles que :

- $\varnothing \in \mathcal{E}$, et $a \in \mathcal{E}$ pour toute lettre a ;
- pour toutes expressions E et E' de \mathcal{E} , les expressions $E + E'$, $E \cdot E'$ et E^* sont encore dans \mathcal{E} .

Notons que $+$, \cdot et $*$ sont vus ici des symboles inertes et non des opérations.

Notation 1.50. Pour alléger les notations, Le point dénotant le produit et les accolades autour des singletons sont omis dans l'écriture des expressions rationnelles. Ainsi l'expression $(\{a\} + \{b\} \cdot \{a\})^*$ est écrite $(a + ba)^*$. De plus, Si $A = \{a_1, \dots, a_n\}$, on utilise A comme une abréviation pour $a_1 + \dots + a_n$.

Exemple 1.51. Quelques exemples d'expressions rationnelles.

A^*	tous les mots
aA^*	mots commençant par a
A^*a	mots finissant par a
$(b + ab)^*(a + \varepsilon)$	mots n'ayant pas deux a consécutifs
$a^* + b^*$	mots n'ayant que des a ou que des b
$(aa + b)^*$	mots avec des blocs de a de longueur paire
$(ab^*a + b)^*$	mots ayant un nombre pair de a

Exercice 1.52. Donner une description en français des langages donnés par les expressions rationnelles suivantes : AA , $(\varepsilon + A)(\varepsilon + A)$, $(AA)^*$, A^*aA^* , A^*abA^* , $A^*aA^*bA^*$ et $(ab)^*$.

Solution.

- AA est le langage des mots de longueur 2.
- $(\varepsilon + A)(\varepsilon + A)$ est le langage des mots de longueur au plus 2.
- $(AA)^*$ est le langage des mots de longueur paire.
- A^*aA^* est le langage des mots ayant au moins une occurrence de a .
- A^*abA^* est le langage des mots ayant au moins une occurrence du facteur ab .
- $A^*aA^*bA^*$ est le langage des mots ayant au moins une occurrence de a puis ensuite une occurrence d'un b .
- $(ab)^*$ est le langage des mots commençant par a , finissant par b et n'ayant pas deux a ou deux b consécutifs.

1.5.2 Automates

Une autre définition des langages rationnels peut être donnée en utilisant les automates. Il s'agit d'un type de machines très simples qui sont des cas particuliers des machines de Turing.

Définition 1.53 (Automate). Un *automate* \mathcal{A} sur l'alphabet A est un quintuplet (Q, A, E, I, F) où Q est fini, $I \subseteq Q$, $F \subseteq Q$ et $E \subseteq Q \times A \times Q$. On appelle les éléments de Q les *états*, ceux de I les *états initiaux*, ceux de F les *états finaux* et ceux de E les *transitions*.

Un automate est en fait un graphe enrichi d'étiquettes sur les arêtes et d'états initiaux et finaux. Une transition (p, a, q) est notée $p \xrightarrow{a} q$ à la manière d'une arête d'un graphe.

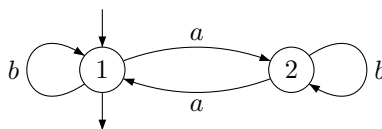


FIGURE 1.7 – Un automate avec quatre transitions

Exemple 1.54. Soit $\mathcal{A} = (\{1, 2\}, \{a, b\}, \{(1, b, 1), (1, a, 2), (2, b, 2), (2, a, 1)\}, \{1\}, \{1\})$ un automate représenté à la figure 1.7. Cet automate a les deux états 1 et 2. L'état 1 est à la fois initial (marqué d'une petite flèche entrante) et final (marqué d'une petite flèche sortante). Il possède quatre transitions représentées comme des arêtes d'un graphe.

Définition 1.55 (Chemin). Un *chemin* dans un automate (Q, A, E, I, F) est une suite finie de transitions consécutives

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$$

noté aussi de manière concise $q_0 \xrightarrow{a_1 \cdots a_n} q_n$. L'état q_0 est l'*état de départ* et q_n est l'*état d'arrivée* du chemin. Le mot $a_1 \cdots a_n$ est l'*étiquette* du chemin.

Exemple 1.56. La suite $2 \xrightarrow{a} 1 \xrightarrow{b} 1 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{b} 2$ est un chemin dans l'automate de l'exemple précédent. Son étiquette est le mot *abbab*.

Définition 1.57 (Acceptation). Un chemin est *acceptant* ou *réussi* lorsque l'état de départ est initial et l'état d'arrivée est final. Un mot est *accepté* par l'automate \mathcal{A} s'il est l'étiquette d'un chemin acceptant de \mathcal{A} . Le langage des mots acceptés par l'automate \mathcal{A} est noté $L(\mathcal{A})$.

Exemple 1.58. L'ensemble des mots acceptés par l'automate de la figure 1.7 est le langage $(ab^*a + b)^*$ des mots ayant un nombre pair d'occurrences de a .

Le théorème de Kleene établit l'équivalence entre les expressions rationnelles et les automates finis dans le sens où ces deux notions définissent les mêmes langages. Ce résultat est remarquable car il relie deux notions de natures différentes, combinatoire pour les expressions rationnelles et opérationnelle pour les automates. Le théorème de Kleene admet de nombreuses extensions à des structures telles les arbres et les mots infinis ou même transfinis.

Théorème 1.59 (Kleene 1956). *Un langage L est rationnel si et seulement si il existe un automate (fini) \mathcal{A} tel que $L = L(\mathcal{A})$.*

La preuve du théorème utilise les notions d'automate émondé et normalisé, que nous définissons successivement.

Définition 1.60 (Automate émondé). Un automate est *émondé* si par tout état passe au moins un chemin acceptant.

Il est clair que les états par lesquels ne passe aucun chemin acceptant peuvent être supprimés sans changer l'ensemble des mots acceptés par l'automate. Un état q apparaît sur un chemin acceptant s'il est accessible d'un état initial et si un état final est accessible à partir de q (on dit que q est co-accessible). L'ensemble des états accessibles et co-accessibles peut être calculé par deux parcours en largeur de l'automate considéré comme un graphe. On suppose souvent dans la suite que les automates sont émondés.

Définition 1.61 (Automate normalisé). Un automate est *normalisé* s'il possède un unique état initial qui n'est l'état d'arrivée d'aucune transition et un unique état final qui n'est l'état de départ d'aucune transition.

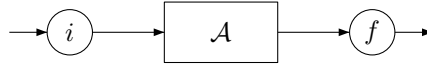


FIGURE 1.8 – Schéma d'un automate normalisé

Si l'état initial et l'état final d'un automate normalisé coïncident, aucune transition n'est adjacente à cet état et l'automate accepte uniquement le mot vide. Sinon, l'automate n'accepte pas le mot vide. La proposition suivante n'est pas nécessaire à la preuve du théorème mais la construction utilisée dans la preuve est intéressante en soi.

Proposition 1.62 (Normalisation). *Pour tout automate \mathcal{A} , il existe un automate normalisé \mathcal{A}' tel que $L(\mathcal{A}') = L(\mathcal{A}) \setminus \{\varepsilon\}$.*

Preuve. Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate. Soient i et f deux nouveaux états n'appartenant pas à Q . L'automate \mathcal{A}' est égal à $(Q \cup \{i, f\}, A, E', \{i\}, \{f\})$ où l'ensemble E' des transitions est donné par

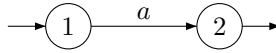
$$\begin{aligned} E' = & E \cup \{i \xrightarrow{a} q \mid \exists p \in I \ p \xrightarrow{a} q \in E\} \\ & \cup \{p \xrightarrow{a} f \mid \exists q \in F \ p \xrightarrow{a} q \in E\} \\ & \cup \{i \xrightarrow{a} f \mid \exists p \in I \ \exists q \in F \ p \xrightarrow{a} q \in E\}. \end{aligned}$$

C'est pure routine de vérifier que $L(\mathcal{A}') = L(\mathcal{A}) \setminus \{\varepsilon\}$. □

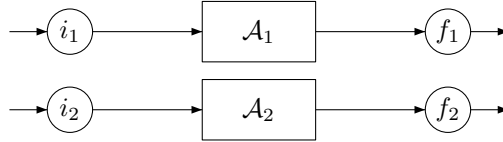
Preuve. Pour un langage L , on note respectivement $\varepsilon(L)$ et $\alpha(L)$ les langages $L \cap \{\varepsilon\}$ et $L \setminus \{\varepsilon\}$. Les formules suivantes montrent qu'on peut calculer récursivement $\alpha(L)$ et $\varepsilon(L)$ à partir d'une expression rationnelle de L .

$$\begin{aligned} \varepsilon(L + L') &= \varepsilon(L) + \varepsilon(L') & \alpha(L + L') &= \alpha(L) + \alpha(L') \\ \varepsilon(LL') &= \varepsilon(L)\varepsilon(L') & \alpha(LL') &= \alpha(L)\alpha(L') + \varepsilon(L)\alpha(L') + \alpha(L)\varepsilon(L') \\ \varepsilon(L^*) &= \varepsilon & \alpha(L^*) &= \alpha(L)^+ \end{aligned}$$

On montre par induction sur (la longueur de) l'expression rationnelle que si L est rationnel alors $\alpha(L)$ est accepté par un automate normalisé. Pour obtenir un automate acceptant L , il suffit d'ajouter éventuellement un nouvel état à la fois initial et final pour accepter le mot vide.

FIGURE 1.9 – Automate normalisé acceptant $L = a$

On considère d'abord les cas de base. Le langage $L = a$ est accepté par l'automate de la figure 1.9

FIGURE 1.10 – Automates normalisés \mathcal{A}_1 et \mathcal{A}_2

Soient L_1 et L_2 deux langages rationnels et soient \mathcal{A}_1 et \mathcal{A}_2 deux automates normalisés acceptant respectivement les langages $\alpha(L_1)$ et $\alpha(L_2)$. Ces deux automates sont schématisés à la figure 1.10

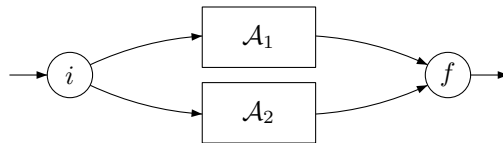


FIGURE 1.11 – Automate pour l'union

Si $L = L_1 + L_2$, on utilise la formule $\alpha(L) = \alpha(L_1) + \alpha(L_2)$. Un automate acceptant $\alpha(L)$ est construit en faisant l'union disjointe des deux automates \mathcal{A}_1 et \mathcal{A}_2 . Cet automate est ensuite normalisé en fusionnant i_1 avec i_2 et f_1 avec f_2 (cf. figure 1.11).

Si $L = L_1L_2$, on utilise la formule $\alpha(L) = \alpha(L_1)\alpha(L_2) + \varepsilon(L_1)\alpha(L_2) + \alpha(L_1)\varepsilon(L_2)$. D'après le cas précédent, on sait construire un automate normalisé pour l'union. Il suffit donc de savoir construire un automate pour $\alpha(L_1)\alpha(L_2)$. Un automate normalisé acceptant $\alpha(L_1)\alpha(L_2)$ est construit en faisant l'union disjointe des deux automates \mathcal{A}_1 et \mathcal{A}_2 puis en fusionnant f_1 avec i_2 (cf. figure 1.12).

Si $L = L_1^*$, on utilise la formule $\alpha(L) = (\alpha(L_1))^+$. Un automate normalisé acceptant $(\alpha(L_1))^+$ est construit en dupliquant chacun des états i_1 et f_1 en des états i'_1 et f'_1 puis en fusionnant i'_1 avec f'_1 (cf. figure 1.13). Cet automate peut aussi être obtenu en fusionnant d'abord i_1 avec f_1 puis en normalisant ensuite l'automate grâce à la proposition précédente.

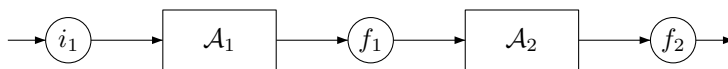


FIGURE 1.12 – Automate pour le produit

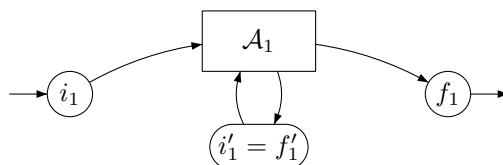


FIGURE 1.13 – Automate pour l'étoile

Pour l'autre sens de l'équivalence, on renvoie le lecteur aux algorithmes ci-dessous. \square

Il existe de nombreuses autres méthodes pour convertir une expression rationnelle en automate. Une variante de la méthode présentée ci-dessus consiste à utiliser des automates *semi-normalisés* où seules les conditions sur l'état initial sont satisfaites. Les constructions pour l'automate du produit et de l'étoile sont un peu plus délicates. Par contre, ces automates peuvent accepter le mot vide et les constructions limitent le nombre d'états engendrés. Cette méthode est pratique pour des constructions à la main. La méthode de Thompson utilise des automates avec ε -transitions qu'il faut ensuite supprimer. D'autres méthodes sont basées sur les quotients à gauche calculés directement sur l'expression. On peut obtenir un automate déterministe par la méthode Brzozowski ou un automate non déterministe par celle d'Antimirov (cf. [Sak03]).

Pour achever la preuve du théorème de Kleene, nous donnons plusieurs algorithmes pour calculer une expression rationnelle décrivant le langage accepté par un automate.

Algorithme de McNaughton et Yamada Ce premier algorithme est facile à programmer mais il impose plus de calculs et conduit de ce fait à de grosses expressions rationnelles. Il n'est pas très pratique pour des calculs à la main.

Quitte à renommer les états, on peut supposer que l'ensemble des états est $Q = \{1, \dots, n\}$. Pour un entier $0 \leq k \leq n$, et deux états $q, q' \in Q$, on définit l'ensemble $L_{q,q'}^{(k)}$ de mots qui étiquettent un chemin de q à q' ne passant que par des états intermédiaires dans l'ensemble $\{1, \dots, k\}$. Plus formellement, l'ensemble $L_{q,q'}^{(k)}$ est défini par

$$L_{q,q'}^{(k)} = \{a_1 \cdots a_n \mid q \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q' \text{ et } p_1, \dots, p_{n-1} \in \{1, \dots, k\}\}.$$

L'ensemble des mots acceptés par \mathcal{A} est égal à l'union finie $\bigcup_{i \in I, f \in F} L_{i,f}^{(n)}$. Il suffit donc de montrer que chacun de ces langages est rationnel. On procède par récurrence sur k . Pour $k = 0$, il ne peut pas y avoir d'état intermédiaire dans le chemin. Celui-ci est

donc de longueur 0 ou 1. Pour $k \geq 0$, on distingue les chemins passant par l'état $k + 1$ des chemins n'y passant pas. Si le chemin de q à q' passe par l'état $k + 1$, il est décomposé en un premier chemin de q à $k + 1$, en un certain nombre de cycles autour de $k + 1$, puis en un dernier chemin de $k + 1$ à q' . On obtient les formules suivantes.

$$L_{q,q'}^0 = \{a \mid q \xrightarrow{a} q' \in E\} \cup \{\varepsilon \mid \text{si } q = q'\}$$

$$L_{q,q'}^{(k+1)} = L_{q,q'}^{(k)} + L_{q,k+1}^{(k)} (L_{k+1,k+1}^{(k)})^* L_{k+1,q'}^{(k)}$$

Comme les formules ci-dessus ne font intervenir que les opérations rationnelles, on montre facilement par récurrence sur k que tous les langages $L_{q,q'}^{(k)}$ et donc aussi $L(\mathcal{A})$ sont rationnels.

Méthode par élimination Cet algorithme manipule des automates dont les transitions sont étiquetées par des expressions rationnelles. La première étape consiste à normaliser l'automate en utilisant la proposition 1.62. La seconde étape remplace pour chaque paire d'états (p, q) , toutes les transitions $p \xrightarrow{a} q$ par une seule transition étiquetée par l'expression rationnelle $\sum_{(p,a,q) \in E} a$. À chaque étape suivante, un des états autre que l'état initial et l'état final est supprimé. L'algorithme s'arrête lorsqu'il ne reste plus que l'état initial et l'état final ainsi qu'une seule transition entre les deux. Le résultat est alors l'expression rationnelle qui étiquette cette transition.

À chaque suppression d'un état s on ajoute, pour chaque paire (p, q) d'états restants, à l'expression portée par la transition de p à q l'expression xy^*z où x, y et z sont les expressions rationnelles portées respectivement par les transitions de p à s , de s à s et de s à q .

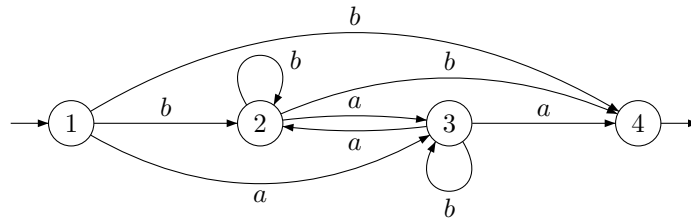


FIGURE 1.14 – Méthode par élimination : normalisation

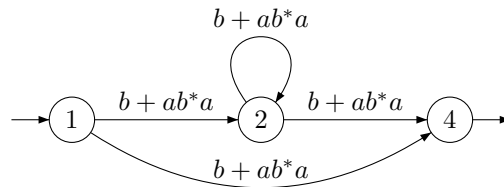


FIGURE 1.15 – Méthode par élimination : suppression de 3

Exemple 1.63. Si on applique la méthode par élimination à l'automate de la figure 1.7, on obtient successivement les automates des figures 1.14, 1.15 et 1.16.

Le lemme suivant permet de résoudre des équations linéaires entre des langages.

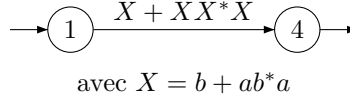


FIGURE 1.16 – Méthode par élimination : suppression de 2

Lemme 1.64 (Lemme d'Arden). *Soient K et L deux langages et soit l'équation $X = KX + L$ où l'inconnue X désigne un langage.*

1. *si $\varepsilon \notin K$, l'unique solution de l'équation est $X = K^*L$.*
2. *si $\varepsilon \in K$, les solutions sont de la forme $X = K^*(L + Y)$ où $Y \subseteq A^*$.*

Le lemme est surtout utilisé lorsque $\varepsilon \notin K$. Dans ce cas, la solution K^*L est à rapprocher de la solution $L/(1 - K)$ de la même équation résolue dans un corps. Le langage K^* est en fait un pseudo-inverse de $1 - K$ comme l'indique son expression $K^* = \sum_{n \geq 0} K^n$ qui ressemble au développement en série entière de la fonction $1/(1 - K)$.

Preuve. Soit X une solution de l'équation $X = KX + L$. Si $Y \subset X$, on montre par récurrence sur n que $K^n Y \subset X$ et donc $K^*Y \subset X$. En considérant $Y = L$, on constate que $K^*L \subset X$. Réciproquement, l'égalité $K = KK^* + \varepsilon$ implique que K^*L est une solution de l'équation. Ceci démontre que K^*L est la plus petite (pour l'inclusion) solution de l'équation. Ce résultat est en fait un cas particulier de la proposition 2.24 concernant les solutions d'un système polynomial.

1. Si $\varepsilon \notin K$, il reste à montrer que K^*L est en fait l'unique solution de l'équation. Montrons par l'absurde que $X = K^*L$ en supposant que $X \setminus K^*L$ est non vide. Soit w un mot de $X \setminus K^*L$ de longueur minimale. Comme $w \notin L$, ce mot s'écrit $w = kw$ avec $k \in K$ et $x \in X$. Comme $k \neq \varepsilon$, le mot x est de longueur strictement inférieure à w . La minimalité de $|w|$ implique que x appartient à K^*L . Ceci aboutit à la contradiction que $w \in K^*L$.
2. Si $\varepsilon \in K$ la solution X s'écrit $X = K^*L + Y$. Comme $Y \subset X$, on a aussi $K^*Y \subset X$ et donc l'égalité $X = K^*(L + Y)$. Comme $KK^* = K^*$, tout langage de la forme $K^*(L + Y)$ est solution de l'équation.

□

Élimination de Gauß Cette méthode consiste à résoudre un système d'équations entre langages par une utilisation systématique du lemme d'Arden. Elle s'apparente à la résolution d'un système d'équations linéaires par l'élimination de Gauß.

Pour chaque état $q \in Q$, soit X_q l'ensemble des mots qui étiquettent un chemin de q à un état final. Le langage $L(\mathcal{A})$ des mots acceptés est égal à l'union finie $\bigcup_{i \in I} X_i$. Pour trouver des expressions pour chacun des langages X_q , on résout grâce au lemme d'Arden le système suivant.

$$X_p = \begin{cases} \sum_{(p,a,q) \in E} aX_q + \varepsilon & \text{si } p \text{ est final} \\ \sum_{(p,a,q) \in E} aX_q & \text{sinon} \end{cases} \quad \text{pour } p \in Q.$$

Supposons pour simplifier que l'ensemble des états est $\{1, \dots, n\}$. La résolution du système est menée de la façon suivante. La première équation est résolue en X_1 en considérant les variables X_2, \dots, X_n comme des constantes. L'expression trouvée est substituée à X_1 dans les autres équations pour faire disparaître la variable X_1 . La seconde équation est alors résolue en X_2 . Le résultat est substituée à X_2 dans les équations restantes et dans l'expression de X_1 trouvée à la première étape. On procède ainsi jusqu'à la dernière équation qui donne une expression de X_n . Cette expression est finalement substituée dans les expressions de X_1, \dots, X_{n-1} trouvées aux étapes précédentes.

Exemple 1.65. Pour l'automate de la figure 1.7, on obtient le système

$$\begin{aligned} X_1 &= bX_1 + aX_2 + \varepsilon \\ X_2 &= aX_1 + bX_2 \end{aligned}$$

En utilisant le lemme d'Arden sur la seconde équation, on obtient $X_2 = b^*aX_1$. En substituant cette expression dans la première équation, on obtient $X_1 = (ab^*a + b)X_1 + \varepsilon$ qui donne finalement $L(\mathcal{A}) = X_1 = (ab^*a + b)^*$ grâce au lemme d'Arden.

Exercice 1.66. On appelle automate avec ε -transitions un automate dont les étiquettes des transitions sont soit une lettre soit le mot vide. Montrer que tout automate avec ε -transition est équivalent à un automate sans ε -transition.

Solution. Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate avec ε -transitions. On définit une relation sur Q notée $\xrightarrow{\varepsilon^*}$ de la façon suivante. Pour deux états p et q , on écrit $p \xrightarrow{\varepsilon^*} q$ s'il existe un chemin, éventuellement vide, de p à q uniquement constitué de transitions étiquetées par ε . La relation $\xrightarrow{\varepsilon^*}$ est donc la clôture réflexive et transitive de la relation $\xrightarrow{\varepsilon}$. On définit alors l'automate $\mathcal{A}' = (Q, A, E', I, F)$ où l'ensemble E' des transitions est donné par

$$E' = \{p \xrightarrow{a} q \mid \exists p', q' \in Q \ p \xrightarrow{\varepsilon^*} p', p' \xrightarrow{a} q' \text{ et } q' \xrightarrow{\varepsilon^*} q\}.$$

Il est immédiat de vérifier que l'automate \mathcal{A}' est sans ε -transition et qu'il est équivalent à \mathcal{A} .

1.6 Automates déterministes

La notion de machine ou de modèle de calcul déterministe est fondamentale et elle apparaît tout au long de cet ouvrage. De manière intuitive, une machine est déterministe si pour chaque entrée, un seul calcul est possible. Pour certains modèles comme les automates ou les machines de Turing, toute machine est équivalente à une machine déterministe. Par contre, le passage à une machine déterministe a un prix qui est le nombre d'états pour les automates (cf. exemples 1.72 et 1.73) ou le temps de calcul pour les machines de Turing. Pour d'autres modèles comme les automates à pile, les machines déterministes sont moins puissantes.

Une propriété essentielle des automates finis est que tout automate est équivalent à un automate déterministe qui accepte le même langage. Cette propriété a un intérêt aussi bien théorique que pratique. D'un point de vue théorique, elle permet de montrer la clôture par complémentation des langages rationnels. D'un point de vue pratique, les automates déterministes sont plus faciles à implémenter. La commande UNIX `grep` utilise par exemple un automate déterministe pour rechercher une occurrence d'une expression

rationnelle dans un texte. En fait, la commande `grep` calcule d'abord un automate non déterministe puis le détermine de manière paresseuse. Elle ne calcule que les états vraiment parcourus par la lecture du texte dans l'automate.

De manière intuitive, un automate est déterministe si une seule transition est possible à chaque instant.

Définition 1.67 (Automate déterministe). Un automate $\mathcal{A} = (Q, A, E, I, F)$ est *déterministe* si :

- il a un unique état initial : $|I| = 1$;
- si (p, a, q) et $(p, a, q') \in E$ alors $q = q'$.

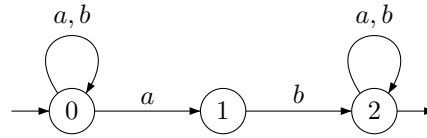


FIGURE 1.17 – Automate non déterministe

Exemple 1.68. L'automate de la figure 1.7 (p. 35) est déterministe. Par contre, l'automate de la figure 1.17 n'est pas déterministe. Il accepte le langage A^*abA^* des mots qui contiennent au moins un facteur ab .

Proposition 1.69. *Tout automate est équivalent à (accepte le même langage que) un automate déterministe.*

Preuve. Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate donné. On construit un automate déterministe équivalent $\hat{\mathcal{A}}$ dont les états sont les parties de Q .

$$\hat{\mathcal{A}} = (\mathfrak{P}(Q), A, \hat{E}, \{I\}, \{P \subseteq Q \mid P \cap F \neq \emptyset\})$$

avec $\hat{E} = \{P \xrightarrow{a} P' \mid P' = \{q \mid \exists p \in P \ p \xrightarrow{a} q\}\}.$

Ainsi, $\hat{\mathcal{A}}$ est déterministe et accepte le même langage que \mathcal{A} comme le montre le lemme suivant. \square

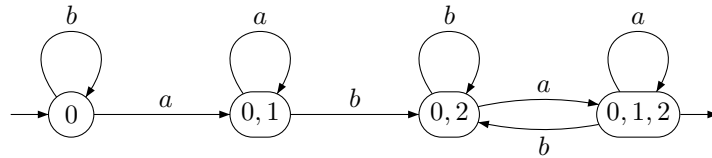


FIGURE 1.18 – Déterminisation de l'automate de la figure 1.17

La construction de l'automate déterministe donnée dans la preuve de la proposition précédente est généralement appelée *construction par sous-ensembles*. Lorsqu'on

applique cette construction sur un exemple précis, on évite de considérer systématiquement tous les sous-ensembles. On se contente de ceux qui donnent un état accessible à partir de l'état initial I . La construction est menée en pratique de la façon suivante. On part de l'état initial puis on ajoute au fur et à mesure les états qui apparaissent comme état d'arrivée des transitions partant des états déjà rencontrés. Il s'agit en fait d'un parcours en largeur de l'automate déterministe émondé.

Exemple 1.70. En appliquant la construction par sous-ensembles à l'automate de la figure 1.17 puis en ne gardant que les états accessibles de l'état initial, on obtient l'automate de la figure 1.18

Le fait que l'automate $\hat{\mathcal{A}}$ soit équivalent à \mathcal{A} découle directement du lemme suivant.

Lemme 1.71. *Pour tout $w \in A^*$, il existe un chemin de I à P dans $\hat{\mathcal{A}}$ étiqueté par w si et seulement si $P = \{q \mid \exists i \in I \ i \xrightarrow{w} q \text{ dans } \mathcal{A}\}$*

La preuve du lemme se fait par récurrence sur la longueur de w .

Un des problèmes de la construction par sous-ensembles utilisée dans la preuve de la proposition 1.69 est l'explosion du nombre d'états. Si l'automate \mathcal{A} a n états, l'automate déterministe $\hat{\mathcal{A}}$ équivalent peut avoir jusqu'à 2^n états comme le montrent les exemples suivants.

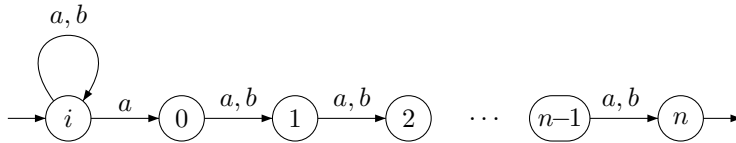


FIGURE 1.19 – Automate non déterministe pour A^*aA^n

Exemple 1.72. Soient $A = \{a, b\}$ et n un entier. Le langage A^*aA^n est accepté par un automate non déterministe ayant $n + 2$ états. Par contre tout automate déterministe acceptant ce langage a au moins 2^n états.

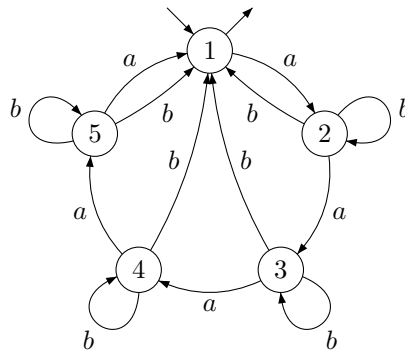


FIGURE 1.20 – Automate \mathcal{A}_5 de l'exemple 1.73

Exemple 1.73. Soient $A = \{a, b\}$, n un entier et Q_n l'ensemble $\{1, \dots, n\}$. On considère l'automate $\mathcal{A}_n = (Q_n, A, E_n, \{1\}, \{1\})$ où l'ensemble E_n des transitions est donné par

$$E_n = \{i \xrightarrow{a} i+1 \mid 1 \leq i \leq n-1\} \cup \{n \xrightarrow{a} 1\} \\ \cup \{i \xrightarrow{b} i \mid 2 \leq i \leq n\} \cup \{i \xrightarrow{b} 1 \mid 2 \leq i \leq n\}.$$

Tout automate déterministe et complet qui est équivalent à \mathcal{A}_n possède au moins 2^n états.

Définition 1.74 (Automate complet). Un automate est *complet* si pour toute paire (p, a) de $Q \times A$, il existe un état q tel que $p \xrightarrow{a} q$ soit une transition.

Notation 1.75. Dans un automate déterministe complet $\mathcal{A} = (Q, A, E, I, F)$, on notera pour tout état $q \in Q$ et toute lettre $a \in A$, l'état $q \cdot a$ l'unique état p tel que $q \xrightarrow{a} p$ soit une transition de \mathcal{A} .

Proposition 1.76. *Tout automate (déterministe) est équivalent à un automate (déterministe) complet.*

Preuve. On introduit un nouvel état $p \notin Q$ appelé *puits* et on pose $Q' = Q \cup \{p\}$. On note $R = \{(q, a) \mid q \in Q' \text{ et } \forall q' \ q \xrightarrow{a} q' \notin E\}$ l'ensemble des transitions manquantes. L'automate complété est l'automate \mathcal{A}' défini par $\mathcal{A}' = (Q', A, E', I, F)$, avec $E' = E \cup \{q \xrightarrow{a} p \mid (q, a) \in R\}$. Notons que si \mathcal{A} est déterministe, l'automate complété \mathcal{A}' l'est également. \square



FIGURE 1.21 – Complétion par ajout d'un état *puits*

Corollaire 1.77 (Clôture par complémentation). *Si $L \subseteq A^*$ est rationnel, alors $A^* \setminus L$ est rationnel.*

Preuve. Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate déterministe et complet acceptant le langage L . L'automate $\mathcal{A}' = (Q, A, E, I, Q \setminus F)$ accepte le langage $L(\mathcal{A}') = A^* \setminus L$. \square

1.7 Automate minimal

Dans cette partie, on montre que tout langage rationnel est accepté par un automate minimal qui est le quotient de tout automate déterministe acceptant ce langage. On étudie ensuite quelques algorithmes permettant de calculer cet automate minimal. Une très bonne référence pour cette partie est [BBC93].

1.7.1 Quotients

Les quotients à gauche constituent un outil indispensable à l'étude des automates déterministes acceptant un langage L puisque le nombre de quotients donne un minortant du nombre d'états. Ils fournissent également une caractérisation très utile des langages rationnels (Proposition 1.82). Celle-ci est, par exemple, utilisée de manière cruciale dans la preuve du théorème 1.106 (p. 54).

Définition 1.78 (Quotients à gauche). Soit $L \subseteq A^*$ un langage. Le *quotient à gauche* de L par un mot $u \in A^*$ est le langage $u^{-1}L = \{v \mid uv \in L\}$. Le *quotient à gauche* de L par un langage $K \subseteq A^*$ est $K^{-1}L = \bigcup_{k \in K} k^{-1}L$.

De manière symétrique, on peut aussi définir les *quotients à droite* d'un langage. L'intérêt des quotients à gauche réside dans leurs liens avec les automates déterministes.

Exemple 1.79. Soit $L = (ab^*a + b)^*$ le langage des mots ayant un nombre pair d'occurrences de a . On a les égalités $a^{-1}L = b^*aL$ et $b^{-1}L = L$ où b^*aL est le langage des mots ayant un nombre impair d'occurrences de a .

Les formules suivantes permettent de calculer explicitement un quotient $a^{-1}L$ puis un quotient $w^{-1}L$ en procédant par récurrence sur la longueur de w . On parvient ainsi à calculer tous les quotients à gauche d'un langage.

Proposition 1.80. Pour toute lettre a , tous mots u, v et w et tous langages K et L , on a les relations suivantes qui permettent de calculer les quotients à gauche.

- $w^{-1}(K + L) = w^{-1}K + w^{-1}L$
- $a^{-1}(KL) = (a^{-1}K)L + \varepsilon(K)a^{-1}L$
- $w^{-1}(KL) = (w^{-1}K)L + \sum_{uv=w} \varepsilon(u^{-1}K)v^{-1}L$ en notant $\varepsilon(L) = \varepsilon \cap L$.
- $a^{-1}(L^*) = (a^{-1}L)L^*$
- $w^{-1}L^* = \sum_{uv=w} \varepsilon(u^{-1}L^*)(v^{-1}L)L^*$
- $(uv)^{-1}L = v^{-1}(u^{-1}L)$.

Le lemme suivant établit le lien fondamental entre les quotients à gauche d'un langage et les automates déterministes acceptant ce même langage. Des quotients à droite auraient pu être définis de manière symétrique. Nous avons privilégié les quotients à gauche parce qu'ils sont justement adaptés aux automates déterministes. Les quotients à droite auraient été utiles pour les automates co-déterministes, c'est-à-dire qui deviennent déterministes lorsque les transitions sont renversées en échangeant état de départ et état d'arrivée.

Lemme 1.81. Soit $\mathcal{A} = (Q, A, E, \{i\}, F)$ un automate déterministe acceptant un langage L . Pour tout chemin $i \xrightarrow{u} q$, on a $u^{-1}L = \{v \mid q \xrightarrow{v} f \text{ avec } f \in F\}$.

La preuve du lemme est triviale. Le lemme a pour corollaire immédiat que le nombre de quotients à gauche d'un langage est inférieur au nombre d'états de tout automate déterministe et complet acceptant ce langage. En particulier, le nombre de quotients à gauche d'un langage rationnel est fini. Cette propriété est en fait caractéristique des langages rationnels.

Proposition 1.82. Un langage L est rationnel si et seulement si il a un nombre fini de quotients à gauche.

Pour la preuve de la proposition, on introduit la définition suivante.

Définition 1.83. Soit L un langage rationnel. L'*automate minimal* de L est automate $\mathcal{A}_L = (Q, A, E, I, F)$ où

$$Q = \{u^{-1}L \mid u \in A^*\}, \quad I = \{L\}, \quad F = \{u^{-1}L \mid u \in L\}$$

$$E = \{u^{-1}L \xrightarrow{a} (ua)^{-1}L \mid u \in A^* \text{ et } a \in A\}.$$

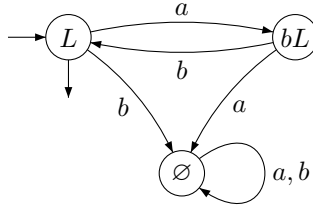


FIGURE 1.22 – Automate minimal de $L = (ab)^*$

Exemple 1.84. Les différents quotients à gauche du langage $L = (ab)^*$ sont les langages $a^{-1}L = bL$, $b^{-1}L = \emptyset$, $a^{-1}(bL) = \emptyset$ et $b^{-1}(bL) = L$. En appliquant la construction donnée dans la preuve de la proposition précédente, on obtient l'automate minimal de L de la figure 1.22.

Le nombre d'états de l'automate minimal est bien sûr minimal puisqu'il est égal au nombre de quotients à gauche. La proposition découle alors directement du lemme suivant.

Lemme 1.85. L'*automate minimal* \mathcal{A}_L accepte le langage L .

Preuve. On montre facilement par récurrence sur la longueur que pour tout mot u , on a un chemin $L \xrightarrow{u} u^{-1}L$ dans l'automate \mathcal{A}_L . La définition des états finaux donne immédiatement le résultat. \square

1.7.2 Congruence de Nerode

On montre dans cette partie que l'automate minimal a en outre la propriété d'être le quotient de tout autre automate déterministe acceptant le même langage. Il peut toujours être obtenu en fusionnant des états.

Pour toute paire $(p, a) \in Q \times A$ d'un automate déterministe, on note, s'il existe, $p \cdot a$ l'unique état q tel que $p \xrightarrow{a} q$ soit une transition. Si de plus l'automate est complet, $p \cdot a$ est toujours défini. Cette notation peut être étendue à tous les mots en définissant par récurrence $p \cdot (ua) = (p \cdot u) \cdot a$. L'état $p \cdot w$ est alors l'unique état q tel que $p \xrightarrow{w} q$ soit un chemin. On a alors défini une action à droite du monoïde A^* sur Q puisque $p \cdot uv = (p \cdot u) \cdot v$ pour tous mots u et v .

Définition 1.86 (Congruence). Soit $\mathcal{A} = (Q, A, E, \{i\}, F)$ un automate déterministe et complet. Une *congruence* sur \mathcal{A} est une relation d'équivalence \sim sur Q qui vérifie pour tous $q, q' \in Q$ et $a \in A$:

$$q \sim q' \implies (q \in F \iff q' \in F),$$

$$q \sim q' \implies q \cdot a \sim q' \cdot a.$$

La première propriété signifie que chaque classe d'une congruence ne contient que des états finaux ou que des états qui ne sont pas finaux. La seconde propriété est qu'une congruence est compatible avec les transitions de l'automate. Rien dans cette définition ne concerne l'état initial.

Si \sim est une congruence sur un automate $\mathcal{A} = (Q, A, E, \{i\}, F)$, l'automate quotient \mathcal{A}/\sim est l'automate $(Q/\sim, A, E', \{[i]\}, \{[f] \mid f \in F\})$ où l'ensemble E' des transitions est donné par

$$E' = \{[q] \xrightarrow{a} [q \cdot a] \mid q \in Q \text{ et } a \in A\}.$$

Cet automate est encore déterministe car la classe $[q \cdot a]$ ne dépend que de la classe de q . Le lemme suivant montre qu'un automate quotient accepte le même langage.

Lemme 1.87. *Soit \sim une congruence sur un automate \mathcal{A} , alors l'automate quotient \mathcal{A}/\sim accepte le langage $L(\mathcal{A})$.*

Preuve. Soit un chemin

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$$

d'étiquette $w = a_1 \dots a_n$ dans l'automate \mathcal{A} . On en déduit que

$$[q_0] \xrightarrow{a_1} [q_1] \xrightarrow{a_2} \cdots \xrightarrow{a_n} [q_n]$$

est un chemin étiqueté par w dans \mathcal{A}/\sim . Si de plus q_0 est initial et q_n est final dans \mathcal{A} , alors $[q_0]$ est initial et $[q_n]$ est final dans \mathcal{A}/\sim . Ceci prouve l'inclusion $L(\mathcal{A}) \subseteq L(\mathcal{A}/\sim)$.

Soit maintenant un chemin

$$[q_0] \xrightarrow{a_1} [q_1] \xrightarrow{a_2} \cdots \xrightarrow{a_n} [q_n]$$

d'étiquette $w = a_1 \dots a_n$ dans l'automate \mathcal{A}/\sim . On montre par récurrence sur n que pour tout état q'_0 de la classe $[q_0]$, il existe un chemin

$$q'_0 \xrightarrow{a_1} q'_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q'_n$$

d'étiquette w tel que $q'_i \sim q_i$ pour tout $0 \leq i \leq n$. Si le chemin dans \mathcal{A}/\sim est acceptant, on peut choisir $q'_0 = i$ et l'état q'_n est alors final. Ceci montre l'inclusion $L(\mathcal{A}/\sim) \subseteq L(\mathcal{A})$. \square

On introduit maintenant la congruence de Nerode qui permet le calcul de l'automate minimal d'un langage à partir de n'importe quel automate déterministe le reconnaissant.

Définition 1.88 (Congruence de Nerode). Soit $\mathcal{A} = (Q, A, E, \{i\}, F)$ un automate déterministe et complet. La congruence de Nerode est définie pour tous états q et q' par

$$q \sim q' \stackrel{\text{def}}{\iff} \forall w \in A^* (q \cdot w \in F \iff q' \cdot w \in F).$$

La relation ainsi définie est effectivement une congruence d'automates. Elle sature l'ensemble des états finaux. L'équivalence $q \in F \iff q' \in F$ obtenue en prenant $w = \varepsilon$ dans la définition montre que deux états équivalents sont tous les deux finaux ou tous les deux non finaux. De plus, l'équivalence $q \cdot aw \in F \iff q' \cdot aw \in F$ implique que si q et q' sont équivalents, alors $q \cdot a$ et $q' \cdot a$ sont encore équivalents.

La congruence de Nerode est la congruence la plus grossière qui sépare les états finaux des états non finaux. En effet, si $q \cdot w$ est final alors que $q' \cdot w$ ne l'est pas, les états q et q' ne peuvent pas être équivalents car sinon les états $q \cdot w$ et $q' \cdot w$ le sont aussi.

La proposition suivante donne comment obtenir l'automate minimal à partir de n'importe quel automate déterministe.

Proposition 1.89. *Soit \mathcal{A} un automate déterministe et complet acceptant un langage L . L'automate minimal \mathcal{A}_L est égal à \mathcal{A}/\sim où \sim est la congruence de Nerode de \mathcal{A} .*

Preuve. Pour un état q , on note L_q le langage $\{w \mid q \cdot w \in F\}$. Deux états q et q' vérifient $q \sim q'$ si et seulement si $L_q = L_{q'}$. D'après le lemme 1.81, chaque langage L_q est de la forme $u^{-1}L$ pour un mot u qui étiquette un chemin de l'état initial à q . On peut donc identifier les classes de la congruence de Nerode avec les quotients à gauche de L . \square

1.7.3 Calcul de l'automate minimal

D'après la proposition précédente, calculer l'automate minimal revient à calculer la congruence de Nerode d'un automate déterministe. Ce calcul peut être fait de manière très efficace. Une première méthode naïve donne un temps de calcul en $O(n^2)$ où n est le nombre d'états de l'automate. Une seconde méthode basée sur le principe *diviser pour régner* permet d'obtenir un temps de calcul en $O(n \log n)$. Le résultat de cette méthode est étonnant car le principe *diviser pour régner* n'est pas évident à appliquer à la minimisation d'automates.

Méthode itérative

Soit $\mathcal{A} = (Q, A, E, \{i\}, F)$ un automate déterministe. On définit par récurrence une suite $(\sim_i)_{i \geq 0}$ de relations d'équivalence sur Q par

$$\begin{aligned} q \sim_0 q' &\iff (q \in F \iff q' \in F) \\ q \sim_{i+1} q' &\iff q \sim_i q' \text{ et } \forall a \in A \quad q \cdot a \sim_i q' \cdot a \end{aligned}$$

Proposition 1.90. *Il existe $k \leq |Q|$ tel que $\sim_k = \sim_{k+1}$. De plus, $\sim_k = \sim_{k+n}$ pour tout $n \geq 0$ et \sim_k est la congruence de Nerode.*

Preuve. Les deux premières affirmations sont évidentes et on conclut par récurrence. Si $q \not\sim_k q'$ alors (comme $\sim_k = \sim_{k+1}$), $q \cdot a \not\sim_k q' \cdot a$, et donc $L_q \neq L_{q'}$ d'où $q \not\sim_N q'$. Cela montre que \sim_k est moins fine que la congruence de Nerode.

Mais, si $q \sim_k q'$, alors pour tout $u = a_1 \cdots a_n$ on a :

$$q \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_n \in F \iff q' \xrightarrow{a_1} \cdots \xrightarrow{a_n} q'_n \in F$$

Ce qui implique que $q \sim_N q'$. \square

Algorithme de Hopcroft

L'algorithme de Hopcroft calcule la congruence de Nerode d'un automate déterministe en temps $O(|A|n \log n)$ où n est le nombre d'états. Comme beaucoup d'algorithmes en temps $n \log n$, il utilise le principe de *diviser pour régner*. Son utilisation est ici particulièrement astucieuse. La mise en œuvre de ce principe pour le calcul de la congruence de Nerode n'est pas évidente. Ce tour de force conduit à un algorithme dont le fonctionnement est relativement subtil. L'analyse de la complexité est également difficile.

Dans cette partie, on identifie une relation d'équivalence sur un ensemble Q avec la partition de Q en classes qu'elle induit.

Définition 1.91 (Stabilité et coupure). Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate complet déterministe et émondé, et soient $a \in A$ et $B, C \subseteq Q$. La partie B est *stable* pour (C, a) si $B \cdot a \subseteq C$ ou si $B \cdot a \cap C = \emptyset$ avec $B \cdot a = \{q \cdot a \mid q \in B\}$. Sinon, la paire (C, a) *coupe* B en les deux parties B_1 et B_2 définies par

$$B_1 = \{q \in B \mid q \cdot a \in C\} \quad \text{et} \quad B_2 = \{q \in B \mid q \cdot a \notin C\}$$

Par extension, une partition de Q est dite *stable* pour (C, a) si chacune de ses parts est stable pour (C, a) .

La notion de stabilité qui vient d'être introduite permet de reformuler le fait d'être une congruence pour une partition. Une partition $\{P_1, \dots, P_k\}$ de Q compatible avec F est une congruence si et seulement si elle est stable pour chacune des paires (P_i, a) où P_i parcourt les parts et a les lettres de l'alphabet.

La preuve du lemme suivant est immédiate. Par contre, le résultat de celui-ci est le principe même sur lequel est basé l'algorithme de Hopcroft.

Lemme 1.92. Si $B \subseteq Q$ et $C = C_1 \uplus C_2$ où \uplus représente l'union disjointe, on a alors :

1. Si B est stable pour (C_1, a) et (C_2, a) alors B est stable pour (C, a) .
2. Si B est stable pour (C_1, a) et (C, a) alors B est stable pour (C_2, a) .

Cet algorithme fonctionne globalement de la façon suivante. Il maintient une partition de l'ensemble Q des états dans la variable \mathcal{P} et un ensemble \mathcal{S} de paires (C, a) où C est une part de la partition et a une lettre. Les paires (C, a) de \mathcal{S} sont celles pour lesquelles il reste à vérifier que les parts de la partition sont stables. L'algorithme s'arrête donc dès que cet ensemble \mathcal{S} est vide. Tant que cet ensemble n'est pas vide, une paire (C, a) de cet ensemble est traitée. Le traitement consiste à remplacer chaque part B coupée par (C, a) en B_1 et B_2 par les nouvelles parts B_1 et B_2 . Ce remplacement doit s'effectuer d'abord dans la partition \mathcal{P} puis dans l'ensemble \mathcal{S} . C'est ce second remplacement qui recèle toute l'ingéniosité de cet algorithme. Si une paire (B, b) est présente dans \mathcal{S} , celle-ci est remplacée par les deux paires (B_1, b) et (B_2, b) . On applique alors le point 1 du lemme précédent. Si la paire (B, b) est absente de l'ensemble \mathcal{S} , seule une des deux paires (B_1, b) ou (B_2, b) est ajoutée à \mathcal{S} . On applique alors le point 2 du lemme précédent. Le principe *diviser pour régner* est mis en œuvre en ajoutant à \mathcal{S} la paire dont la première composante est de plus petit cardinal. Pour deux parties B et B' de Q , on note $\min(B, B')$ celle des deux ayant le plus petit cardinal ou n'importe laquelle des deux si elles ont même cardinal.

Nous allons maintenant analyser cet algorithme. Nous allons en particulier prouver qu'il termine toujours et qu'il calcule bien la congruence de Nerode. L'implémentation de cet algorithme est particulièrement délicate. Pour obtenir la complexité annoncée de $n \log n$, il faut utiliser les structures de données appropriées. Une description complète de l'implémentation peut être trouvée en [BBC93].

Nous commençons par prouver la terminaison. À chaque itération de la boucle principale `while`, soit la partition \mathcal{P} est raffinée parce qu'au moins une part B est coupée, soit le cardinal de \mathcal{S} a diminué d'une unité. Comme ces deux événements ne peuvent survenir qu'un nombre fini de fois, l'algorithme s'arrête nécessairement car \mathcal{S} devient vide.

Nous montrons maintenant que la partition \mathcal{P} est égale à la congruence de Nerode lorsque l'algorithme se termine. Il est facile de voir que la congruence de Nerode raffine

Input Automate déterministe $\mathcal{A} = (Q, E, E, I, F)$

- 1: $\mathcal{P} \leftarrow (F, Q \setminus F)$
- 2: $\mathcal{S} \leftarrow \{(\min(F, Q \setminus F), a) \mid a \in A\}$
- 3: **while** $\mathcal{S} \neq \emptyset$ **do**
- 4: $(C, a) \leftarrow$ un élément de \mathcal{S}
- 4: // Boucle en complexité proportionnelle à $|C||a|$
- 5: $\mathcal{S} \leftarrow \mathcal{S} \setminus \{(C, a)\}$
- 6: **for** chaque B coupé par (C, a) en B_1, B_2 **do**
- 7: remplacer B par B_1, B_2 dans \mathcal{P}
- 8: **for all** $b \in A$ **do**
- 9: **if** $(B, b) \in \mathcal{S}$ **then**
- 10: remplacer (B, b) par (B_1, b) et (B_2, b) dans \mathcal{S}
- 11: **else**
- 12: ajouter $(\min(B_1, B_2), b)$ à \mathcal{S}

Algorithme 1: Algorithme de Hopcroft

toujours la partition \mathcal{P} . Chaque coupure réalisée par l'algorithme est en effet nécessaire. Il reste donc à montrer qu'aucune coupure n'a été oubliée, c'est-à-dire que la partition \mathcal{P} est stable pour chacune des paires (C, a) où C parcourt toutes les parts de \mathcal{P} . Dans ce but, nous allons montrer l'invariant suivant.

Lemme 1.93. *Pour toute lettre $a \in A$ et toute part $P \in \mathcal{P}$, P est combinaison booléenne de :*

- parties C telles que \mathcal{P} est stable pour (C, a) ;
- parties C telles que $(C, a) \in \mathcal{S}$.

La preuve se fait facilement par induction sur le nombre d'itérations de la boucle principale de l'algorithme. Lorsque l'ensemble \mathcal{S} est vide, la partie P est combinaison booléenne de parties C telles que \mathcal{P} est stable pour (C, a) . Il découle directement du lemme 1.92 que la partition \mathcal{P} est stable pour (P, a) .

Il reste finalement à analyser la complexité de l'algorithme. Une implémentation rigoureuse garantit que le déroulement du corps de la boucle principale prend un temps au plus proportionnel à la taille de la partie C . La complexité totale est donc bornée par la somme des tailles de ces parties C . Pour calculer cette somme, nous allons utiliser la relation suivante qui est vraie pour chaque lettre a fixée.

$$\sum_{(C,a) \text{ traitée}} |C| = \sum_{q \in Q} |\{(C, a) \mid (C, a) \text{ traitée et } q \in C\}|$$

Pour obtenir la complexité $O(|A||Q| \log |Q|)$, il suffit de montrer que le cardinal de chaque ensemble $\{(C, a) \mid (C, a) \text{ traitée et } q \in C\}$ est borné par $\log_2 |Q|$ pour chaque lettre a et chaque état q .

Soit un état q et une lettre a . Supposons qu'à une itération donnée de la boucle principale, une paire (C, a) soit traitée et que la part C contienne q . Soit (C', a) la paire suivante traitée telle que C' contienne q . Comme les parties C et C' sont des parts de la partition \mathcal{P} et que cette dernière ne peut être que raffinée, la partie C' est contenue dans C et C' provient d'une coupure de C . Comme l'algorithme ajoute à \mathcal{S} la paire

dont la première composante est de cardinal minimal, on a l'inégalité $|C'| \leq |C|/2$. Ceci prouve que le nombre de paires (C, a) traitées où C contient q est au plus $\log_2 |Q|$ et termine l'analyse de la complexité de l'algorithme.

Les méthodes pour minimiser s'appliquent lorsque l'automate de départ est déterministe. Si ce n'est pas le cas, il peut toujours être d'abord déterminisé grâce à la proposition 1.69 puis ensuite minimisé. L'exercice suivant donne une autre méthode pour calculer l'automate minimal en utilisant uniquement des déterminisations.

Exercice 1.94. Pour un automate $\mathcal{A} = (Q, A, E, I, T)$, on note $d(\mathcal{A})$ la partie accessible de l'automate $\hat{\mathcal{A}}$ construit dans la preuve de la proposition 1.69. On note aussi $t(\mathcal{A})$ l'automate (Q, A, E^t, F, I) où l'ensemble E^t des transitions est donné par

$$E^t = \{p \xrightarrow{a} q \mid q \xrightarrow{a} p \in E\}.$$

Soit \mathcal{A} acceptant un langage L . Montrer que l'automate $d(t(d(t(\mathcal{A}))))$ est l'automate minimal \mathcal{A}_L de L .

Solution. Soit \mathcal{A}' l'automate $d(t(d(t(\mathcal{A}))))$. Par construction, cet automate est déterministe. On vérifie en outre qu'il accepte le langage L . Il reste donc à vérifier que cet automate est minimal.

Soient P et P' deux états distincts de \mathcal{A}' . Par définition, P et P' sont deux ensembles états de l'automate $d(t(\mathcal{A}))$. Soit R un état de $d(t(\mathcal{A}))$ tel que $R \in P$ et $R \notin P'$. Par définition de $d(t(\mathcal{A}))$, R est un ensemble d'états de \mathcal{A} . Grâce au lemme 1.71, il existe un mot w tel que $R = \{q \mid q \xrightarrow{w} f \text{ avec } f \in F\}$. On en déduit que dans \mathcal{A}' , il existe un chemin de P à un état final étiqueté par w alors que ce n'est pas le cas pour P' . Ceci montre que P et P' ne sont pas équivalents pour la congruence de Nerode.

1.8 Propriétés de clôture

La classe des langages rationnels est très robuste. Elle est close pour d'innombrables opérations. Ceci est une propriété importante de ces langages et justifie l'intérêt qui leur est porté. Cette partie donne les principales propriétés de clôture mais la liste est loin d'être exhaustive.

1.8.1 Opérations booléennes

Il a déjà été vu que le complémentaire d'un langage rationnel est encore rationnel (cf. corollaire 1.77). Comme la classe des langages rationnels est par définition close par union, elle est aussi close par intersection.

1.8.2 Morphisme et morphisme inverse

La classe des langages rationnels est close par morphisme et par morphisme inverse.

Proposition 1.95. *Soit μ un morphisme de A^* dans B^* . Si $L \subseteq A^*$ est rationnel, alors $\mu(L)$ est rationnel. Si $K \subseteq B^*$ est rationnel, alors $\mu^{-1}(K)$ est aussi rationnel.*

Preuve. Il est possible d'obtenir une expression rationnelle pour $\mu(L)$ à partir d'une expression rationnelle pour L en remplaçant chaque occurrence d'une lettre a par $\mu(a)$.

Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate acceptant le langage K . On construit un automate \mathcal{A}' pour $\mu^{-1}(K)$ ayant même ensemble d'états. Il y a une transition $p \xrightarrow{a} q$ dans \mathcal{A}' s'il y a un chemin $p \xrightarrow{\mu(a)} q$ dans \mathcal{A} . En gardant les mêmes états initiaux et finaux, l'automate \mathcal{A} accepte le langage $\mu^{-1}(K)$. On peut remarquer que si \mathcal{A} est déterministe alors \mathcal{A}' est aussi déterministe. \square

Exercice 1.96. Soient K et L deux langages sur un alphabet A . Le mélange $K \text{ III } L$ des langages K et L est le langage sur A^* défini par la formule suivante.

$$K \text{ III } L = \{u_0 v_0 u_1 \cdots u_n v_n \mid u_i, v_i \in A^*, u_0 \cdots u_n \in K \text{ et } v_0 \cdots v_n \in L\}.$$

1. Montrer que Si les langages L et L' sont rationnels, alors le langage $L \text{ III } L'$ est encore rationnel.
2. Montrer en utilisant le théorème de Higman qu'un langage de la forme $L \text{ III } A^*$ est toujours rationnel même si L n'est pas rationnel.

Solution. Soient $\mathcal{A} = (Q, A, E, I, F)$ et $\mathcal{A}' = (Q', A, E', I', F')$ deux automates acceptant respectivement les langages K et L . On construit l'automate dont l'ensemble d'états est $Q \times Q'$, les ensembles d'états initiaux et finaux sont $I \times I'$ et $F \times F'$ et dont l'ensemble des transitions est

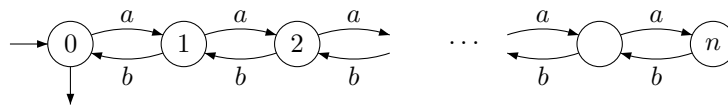
$$\{(p, p') \xrightarrow{a} (q, p') \mid p \xrightarrow{a} q \in E\} \cup \{(p, p') \xrightarrow{a} (p, q') \mid p' \xrightarrow{a} q' \in E'\}.$$

On vérifie sans difficulté que cet automate accepte $K \text{ III } L$.

Le langage $L \text{ III } A^*$ est un idéal pour l'ordre sous-mot. D'après le théorème de Higman, cet idéal est de base fini. Il existe donc un ensemble fini F de mots tel que $L \text{ III } A^* = \bigcup_{w \in F} w \text{ III } A^*$. D'après le résultat précédent, chaque langage $w \text{ III } A^*$ est rationnel et L est donc aussi rationnel.

- Exercice 1.97.*
1. Soient les langages L_n définis par récurrence par $L_0 = \varepsilon$ et $L_{n+1} = L_n \text{ III } (ab)^*$. Montrer que L_n est l'ensemble des mots w tels que $|w|_a = |w|_b$ et $|u|_b \leq |u|_a \leq |u|_b + n$ pour tout préfixe u de w .
 2. Soit une suite croissante n_1, \dots, n_k d'entiers telle que $n_1 = 1$ et pour tout $1 \leq i < n$, on a $n_i \leq n_{i+1} \leq n_0 + \dots + n_i + 1$. Montrer que le langage $(a^{n_1} b^{n_1})^* \text{ III } (a^{n_2} b^{n_2})^* \text{ III } \dots \text{ III } (a^{n_k} b^{n_k})^*$ est encore égal à l'ensemble des mots w tels que $|w|_a = |w|_b$ et $|u|_b \leq |u|_a \leq |u|_b + n_1 + \dots + n_k$ pour tout préfixe u de w . On peut remarquer qu'une suite n_1, \dots, n_k d'entiers vérifie les conditions ci-dessus si et seulement si pour tout entier s tel que $1 \leq s \leq n_1 + \dots + n_k$, il existe une suite d'indices $i_1 < \dots < i_r$ telle que $s = n_{i_1} + \dots + n_{i_r}$.

Solution. Soit un mot w appartenant à L_n . Il existe des mots w_1, \dots, w_n de $(ab)^*$ tels que $w \in w_1 \text{ III } \dots \text{ III } w_n$. Tout préfixe u de w appartient à un langage $u_1 \text{ III } \dots \text{ III } u_n$ où chaque u_i est un préfixe de w_i . Comme chaque u_i vérifie $|u_i|_b \leq |u_i|_a \leq |u_i|_b + 1$, le mot u vérifie bien sûr les inégalités requises. Inversement le langage des mots vérifiant les inégalités est accepté par l'automate déterministe ci-dessous.



Soit w un mot accepté par cet automate. Pour $1 \leq i \leq n$, soit w_i le mot formé des lettres w qui étiquettent les transitions entre les états $i - 1$ à i dans le chemin acceptant w . Il est clair que chaque mot w_i appartient à $(ab)^*$ et que le mot w appartient à $w_1 \text{ III } \cdots \text{ III } w_n$.

Exercice 1.98. Soit L un langage. Pour tout entier n tel que L contienne au moins un mot de longueur n , w_n est défini comme le mot le plus petit pour l'ordre lexicographique de $L \cap A^n$. Montrer que si L est rationnel, le langage $\{w_n \mid L \cap A^n \neq \emptyset\}$ est encore rationnel.

1.9 Lemme de l'étoile et ses variantes

L'intérêt principal de ce lemme, également appelé *lemme d'itération* ou *lemme de pompe* est de donner une condition nécessaire pour qu'un langage soit rationnel. Cette condition est souvent utilisée pour montrer qu'un langage donné n'est pas rationnel. Cette partie et en particulier la présentation du théorème de Ehrenfeucht, Parikh et Rozenberg doit beaucoup à [Sak03, p. 77].

Il existe une multitude de variantes pour ce lemme. Leurs démonstrations sont toutes basées sur un usage plus ou moins fin du principe des tiroirs, ou de sa version plus évoluée, le théorème de Ramsey. Nous donnons d'abord la version la plus simple puis une version forte. Nous terminons par le théorème de Ehrenfeucht, Parikh et Rozenberg qui établit une réciproque au lemme de l'étoile.

Proposition 1.99 (Lemme de l'étoile). *Pour tout langage rationnel L , il existe un entier n tel que pour tout mot f ,*

$$\left. \begin{array}{l} |f| \geq n \\ f \in L \end{array} \right\} \implies \left\{ \begin{array}{l} \exists u, v, w \in A^* \quad v \neq \varepsilon \\ f = uvw \text{ et } uv^*w \subseteq L \end{array} \right.$$

Preuve. Soit n le nombre d'états d'un automate acceptant le langage L , par exemple l'automate minimal de L . Si $|f| \geq n$, un chemin acceptant pour f doit repasser au moins deux fois par un même état q . Si u, v et w sont les étiquettes d'un chemin de l'état initial à q , du circuit de q à q et du chemin de q à un état final, on obtient une factorisation convenable de f . \square

Exercice 1.100. Montrer en utilisant le lemme de l'étoile que le langage $L = \{a^n b^n \mid n \geq 0\}$ n'est pas rationnel.

Solution. Supposons par l'absurde que le langage L soit rationnel. Soit n l'entier fourni par lemme de l'étoile et soit f le mot $f = a^n b^n$. Le mot f se factorise $f = uvw$ où v est non vide et où $uv^*w \subseteq L$. Si v contient à la fois des lettres a et des lettres b , alors uv^2w n'appartient pas à $a^*b^* \supseteq L$ contrairement à l'hypothèse. Si u ne contient que des a ou que des b , le mot uv^2w n'a pas le même nombre de a que de b , ce qui conduit à une contradiction.

Exercice 1.101. Soit A l'alphabet $\{a, b\}$. Montrer en utilisant le résultat de l'exercice précédent que le langage $L = \{w \in A^* \mid |w|_a = |w|_b\}$ n'est pas rationnel.

Solution. Si le langage L est rationnel, alors le langage $L' = L \cap a^*b^*$ est aussi rationnel puisque les langages rationnels sont clos par intersection. Or ce langage L' est justement le langage $\{a^n b^n \mid n \geq 0\}$ considéré à l'exercice précédent.

Le problème du lemme de l'étoile est qu'il donne une condition nécessaire mais pas suffisante. Certains langages peuvent vérifier la condition sans être rationnels.

Exercice 1.102. Soit L le langage $\{a^n b^n \mid n \geq 0\}$. Montrer que le langage $L' = L \cup A^* b a A^*$ vérifie la condition du lemme de l'étoile sans être rationnel.

Solution. Si $n \geq 1$, le mot $f = a^n b^n$ se factorise $f = uvw$ où $u = a^{n-1}$, $v = ab$ et $w = b^{n-1}$. On vérifie sans peine que uv^*w est inclus dans L' . Pour montrer que L' n'est pas rationnel, on procède comme à l'exercice précédent en considérant $L' \cap a^* b^*$.

Il est possible de renforcer la condition du lemme de l'étoile. On obtient alors un lemme qui permet de montrer plus facilement que certains langages ne sont pas rationnels.

Proposition 1.103 (Lemme de l'étoile fort). *Pour tout langage rationnel L , il existe un entier n tel que pour tout mot f ,*

$$\left. \begin{array}{l} f = uv_1 \cdots v_n w \\ f \in L \text{ et } v_i \in A^+ \end{array} \right\} \implies \left\{ \begin{array}{l} \exists i, j \ 0 \leq i < j \leq n \\ uv_1 \cdots v_i (v_{i+1} \cdots v_j)^* v_{j+1} \cdots v_n w \subseteq L \end{array} \right.$$

Malgré le renforcement de la condition, celle-ci n'est toujours pas suffisante, comme le montre l'exemple suivant.

Exemple 1.104. Soient les alphabets $A = \{a, b\}$ et $B = \{a, b, \#\}$. Soit le langage L sur B défini par $L = K' + A^* K A^* \# A^* + A^* \# A^* K A^*$ où les langages K et K' sont définis par $K = \{uu \mid u \in A^+\}$ et $K' = \{u\#v \mid u, v \in A^* \text{ et } u \neq v\}$. On vérifie que le langage L vérifie les conclusions du lemme précédent bien qu'il ne soit pas rationnel.

Nous allons maintenant énoncer le théorème de Ehrenfeucht, Parikh et Rozenberg qui fournit une condition nécessaire et suffisante pour qu'un langage soit rationnel. Ce théorème est basé sur des propriétés σ_k et σ'_k dont nous donnons maintenant les définitions.

Définition 1.105 (Propriétés σ_k et σ'_k). Un langage L vérifie la propriété σ_k (resp. σ'_k) pour $k \geq 0$ si pour toute factorisation $f = uv_1 v_2 \cdots v_k w$ où tous les mots v_i sont non vides, il existe deux indices $1 \leq i < j \leq k$ tels que

$$\begin{array}{ll} \forall n \geq 0 \quad f \in L \iff uv_1 \cdots v_i (v_{i+1} \cdots v_j)^n v_{j+1} \cdots v_k w \in L & \text{pour } \sigma_k \\ f \in L \iff uv_1 \cdots v_i v_{j+1} \cdots v_k w \in L & \text{pour } \sigma'_k \end{array}$$

Il faut remarquer une différence essentielle entre la condition du lemme de l'étoile et les propriétés σ_k et σ'_k . Ces dernières sont des équivalences alors que la condition du lemme est seulement une implication. Ceci signifie que la condition du lemme porte sur le langage lui-même alors que les propriétés σ_k et σ'_k portent simultanément sur le langage et son complémentaire.

Le théorème suivant établit que les propriétés σ_k et σ'_k qui généralisent les lemmes de l'étoile sont caractéristiques des langages rationnels. L'intérêt de ce théorème réside plus dans la technique de preuve et dans l'utilisation astucieuse du théorème de Ramsey que dans le résultat lui-même.

Théorème 1.106 (Ehrenfeucht, Parikh et Rozenberg 1981). *Pour tout langage L , les propositions suivantes sont équivalentes.*

1. L est rationnel.
2. Il existe $k \geq 0$ tel que L vérifie σ_k .
3. Il existe $k \geq 0$ tel que L vérifie σ'_k .

La preuve du théorème s'appuie sur le théorème suivant qui étend le principe des tiroirs. Ce principe très simple énonce que si $n + 1$ objets sont rangés dans n tiroirs, deux objets se trouvent nécessairement dans le même tiroir. Dans le théorème de Ramsey, ce sont les parties à k objets qui sont rangés dans les tiroirs plutôt que les objets eux-mêmes. L'affectation des tiroirs est donnée par la fonction f et les tiroirs sont les éléments de l'ensemble C . Si E est un ensemble, on notera $\mathfrak{P}_k(E)$ l'ensemble des parties à k éléments de E . Le théorème de Ramsey se démontre par récurrence sur k , le cas $k = 1$ étant le principe des tiroirs. Une démonstration détaillée peut être consultée en [vLW92].

Théorème 1.107 (Ramsey 1929). *Pour tout triplet d'entiers naturels (k, m, r) , il existe un entier $N(k, m, r)$ tel que pour tout :*

- ensemble E tel que $|E| \geq N(k, m, r)$,
 - ensemble C tel que $|C| = m$,
 - fonction $f : \mathfrak{P}_k(E) \rightarrow C$,
- il existe $F \subseteq E$ tel que :
- $|F| \geq r$;
 - $|f(\mathfrak{P}_k(F))| \leq 1$.

Dans la littérature, les éléments de l'ensemble C sont appelées les *couleurs* et la fonction f affecte une couleur à chaque partie à k éléments. La condition $|f(\mathfrak{P}_k(F))| \leq 1$ signifie que toutes les parties à k éléments de F se voient affecter la même couleur. On dit alors que F est *monochrome*.

Preuve. Nous montrons successivement que la condition 1 implique la condition 2, qui implique à son tour la condition 3 qui implique finalement la condition 1. Il est tout d'abord évident que la condition 1 implique la condition 2 et que la condition 2 implique la condition 3. Il reste donc à montrer que la condition 3 implique la condition 1. La suite de la preuve est consacrée à cette implication.

La preuve de cette implication se décompose en deux étapes. On montre dans un premier temps que si L satisfait σ'_k , alors le quotient $v^{-1}L$ satisfait σ'_k pour chaque mot v . Dans un second temps, on montre que le nombre de langages vérifiant σ'_k est fini. Finalement, avec ces deux résultats, on obtient qu'un langage vérifiant σ'_k possède un nombre fini de quotients à gauche. La proposition 1.82 (p. 45) permet de conclure qu'il est rationnel.

Première étape Soit L un langage qui vérifie σ'_k pour un entier k fixé et soit un mot $v \in A^*$. On montre que le langage $v^{-1}L$ vérifie aussi σ'_k . Soit alors f , un mot qui s'écrit $f = uv_1v_2 \cdots v_kv$ où les mots v_i ne sont pas vides pour $1 \leq i \leq k$. Les mots u et w sont quelconques. En appliquant la condition σ'_k au langage L et au mot $f' = vf$, on obtient une paire (i, j) et on a alors les équivalences suivantes.

$$\begin{aligned}
 f \in v^{-1}L &\iff vf \in L && \text{Par définition de } v^{-1}L \\
 &\iff vuv_1 \cdots v_iv_{j+1} \cdots v_kv \in L && \text{Par définition de } (i, j) \\
 &\iff uv_1 \cdots v_iv_{j+1} \cdots v_kv \in v^{-1}L && \text{Par définition de } v^{-1}L
 \end{aligned}$$

On a montré que $v^{-1}L$ vérifie σ'_k .

Seconde étape Nous montrons maintenant que le nombre de langages vérifiant σ'_k est fini pour chaque entier k . C'est la partie délicate de la preuve qui fait appel au théorème de Ramsey. En appliquant ce théorème pour $k = 2$, $m = 2$ et $r = k + 1$, on obtient un entier $N = N(2, 2, k + 1)$.

On note $A^{\leq N}$ l'ensemble des mots de longueur au plus N sur l'alphabet A . Soient L et K deux langages vérifiant σ'_k pour un entier k fixé. On montre que si K et L coïncident sur les mots de longueur au plus N , c'est-à-dire $L \cap A^{\leq N} = K \cap A^{\leq N}$, alors les deux langages K et L sont égaux. Ceci prouve que le nombre de langages vérifiant σ'_k est fini. Le nombre de tels langages est en effet borné par 2^m où $m = (|A|^{N+1} - 1)/(|A| - 1)$ est le nombre de mots de longueur au plus N sur l'alphabet A .

On suppose maintenant que $L \cap A^{\leq N} = K \cap A^{\leq N}$. Pour tout mot f , on montre par récurrence sur la longueur $|f|$ que f appartient à K si et seulement si f appartient à L . Le résultat est bien sûr immédiat si f est de longueur au plus N .

On suppose alors que $|f| > N$. Le mot f est factorisé $f = a_0 \cdots a_N g$ où a_0, \dots, a_N sont des lettres et g est un mot quelconque.

On définit maintenant un ensemble de paires d'entiers qui va permettre l'utilisation du théorème de Ramsey. Toute l'intelligence de cette preuve réside dans l'introduction de cet ensemble X défini par la formule suivante.

$$X = \{(i, j) \mid a_0 \cdots a_i a_{j+1} \cdots a_N g \in L\}$$

Soient E l'ensemble $\{1, \dots, N\}$ des entiers de 1 à N , C l'ensemble à deux couleurs $C = \{0, 1\}$ et la fonction χ de $\mathfrak{P}_2(E)$ dans C définie par

$$\chi(i, j) = \begin{cases} 1 & \text{si } (i, j) \in X \\ 0 & \text{sinon.} \end{cases}$$

La fonction χ est en fait la fonction caractéristique de l'ensemble X vu comme un sous-ensemble de $\mathfrak{P}_2(E)$. On a implicitement identifié une partie $\{i, j\}$ à deux éléments avec la paire (i, j) où $i < j$. Le théorème de Ramsey pour $k = 2$, $m = |C|$ et $r = k + 1$ donne un ensemble à $k + 1$ éléments $F = \{i_0, \dots, i_k\}$ tel que $\chi(i_m, i_n)$ reste constant quand les deux indices m et n parcourent $\{0, \dots, k\}$. Ceci signifie que pour toutes paires (m, n) et (m', n') telles que $0 \leq m < n \leq k$ et $0 \leq m' < n' \leq k$, on a l'équivalence

$$a_0 \cdots a_{i_m} a_{i_n+1} \cdots a_N g \in L \iff a_0 \cdots a_{i_{m'}} a_{i_{n'}+1} \cdots a_N g \in L.$$

Les $k + 1$ entiers i_0, \dots, i_k induisent une factorisation $f = uv_1 v_2 \cdots v_k w$ où les mots u, v_1, \dots, v_k, w sont respectivement donnés par

$$\begin{aligned} u &= a_0 a_1 \cdots a_{i_0-1} \\ v_j &= a_{i_{j-1}} a_{i_{j-1}+1} \cdots a_{i_j-1} \text{ pour chaque } 1 \leq j \leq k \\ w &= a_{i_k} \cdots a_N g \end{aligned}$$

Comme les deux langages K et L vérifient la propriété σ'_k , la factorisation $f = uv_1 v_2 \cdots v_k w$ donne deux paires (m, n) et (m', n') d'entiers *a priori* différentes. On a

alors la suite d'équivalences suivantes.

$$\begin{aligned}
f \in L &\iff a_0 \cdots a_{i_m} a_{i_m+1} \cdots a_N g \in L && \text{par définition de } (m, n) \\
&\iff a_0 \cdots a_{i_{m'}} a_{i_{m'}+1} \cdots a_N g \in L && \text{par définition de } F \\
&\iff a_0 \cdots a_{i_{m'}} a_{i_{m'}+1} \cdots a_N g \in K && \text{par hypothèse de récurrence} \\
&\iff f \in K && \text{par définition de } (m', n')
\end{aligned}$$

On a donc montré que f appartient à K si et seulement si f appartient à L et que K et L sont égaux. Ceci termine la preuve du théorème. \square

1.10 Hauteur d'étoile

On étudie ici une classification des langages rationnels basée sur le nombre d'étoiles imbriquées des expressions rationnelles. La raison pour laquelle on s'intéresse à cette opération est qu'elle est celle qui donne toute leur puissance aux expressions rationnelles. Les seuls langages à pouvoir être décrits sans étoile sont les langages finis.

La hauteur d'étoile $h(e)$ d'une expression rationnelle e est définie par récurrence sur la structure de l'expression par les formules suivantes.

$$\begin{aligned}
h(a) &= h(\varepsilon) = 0 \text{ pour toute lettre } a \in A \\
h(e + f) &= \max(h(e), h(f)) \\
h(e f) &= \max(h(e), h(f)) \\
h(e^*) &= 1 + h(e)
\end{aligned}$$

Exemple 1.108. Les expressions $ab + ba$, $(ab + a)^*$ et $(ab^*a + b)^*$ sont respectivement de hauteur d'étoile 0, 1 et 2.

Il faut remarquer qu'un même langage peut être décrit par plusieurs expressions rationnelles ayant des hauteurs d'étoile différentes. Par exemple, les deux expressions $(a + b)^*$ et $(a^*b)^*a^*$ sont de hauteurs d'étoile 1 et 2 mais décrivent le même langage de tous les mots sur l'alphabet $\{a, b\}$. La hauteur d'étoile d'un langage L est définie comme la hauteur minimale d'une expression rationnelle décrivant L . Les langages de hauteur d'étoile 0 sont les langages finis.

Exemple 1.109. Le langage $(a + b)^*$ est de hauteur d'étoile 1 puisqu'il est décrit par une expression rationnelle de hauteur 1 et qu'il ne peut être décrit par une expression de hauteur 0 parce qu'il est infini.

Un premier résultat sur la hauteur d'étoile est que pour tout entier n , il existe un langage rationnel de hauteur d'étoile n . On peut en effet montrer que pour tout entier $n \geq 1$, le langage $L_n = \{w \in A^* \mid |w|_a \equiv 0 \pmod{2^n}\}$ sur l'alphabet $A = \{a, b\}$ est de hauteur d'étoile n . Il existe un algorithme qui calcule la hauteur d'étoile d'un langage rationnel donné mais celui-ci est très complexe et dépasse le cadre de cet ouvrage.

Comme la classe des langages rationnels est close pour toutes les opérations booléennes, il est possible de considérer des expressions utilisant non seulement les opérations rationnelles mais aussi toutes les opérations booléennes. On peut d'ailleurs se limiter à la complémentation puisque l'union et la complémentation permettent d'exprimer les autres opérations booléennes. La hauteur d'étoile peut être étendue à ces

expressions généralisées en posant

$$\begin{aligned}h(\emptyset) &= 0, \\h(e^c) &= h(e).\end{aligned}$$

La *hauteur d'étoile généralisée* d'un langage L est la hauteur d'étoile minimale d'une expression utilisant les opérations rationnelles et booléennes et décrivant L .

Exemple 1.110. Le langage $(a + b)^* = \emptyset^c$ est de hauteur d'étoile généralisée 0.

Les langages de hauteur d'étoile généralisée 0 sont appelés langages sans étoile. Ils sont étudiés en détail et caractérisés de manière algébrique à la section 1.12 (p. 65). C'est encore un problème ouvert de savoir s'il existe des langages de hauteur d'étoile généralisée strictement plus grande que 1.

1.11 Reconnaissance par morphisme

Les langages rationnels peuvent être décrits par des expressions rationnelles ou par des automates. On donne maintenant une troisième façon plus algébrique de définir ces langages.

Définition 1.111 (Monoïde). On appelle *monoïde* tout ensemble M muni d'une loi de composition interne associative qui possède un élément neutre noté 1_M .

Exemple 1.112. Quelques exemples classiques de monoïdes :

- A^* muni de la concaténation (pour A , alphabet quelconque),
- l'ensemble $\mathfrak{P}(A^*)$ des parties de A^* muni du produit des langages,
- tout groupe G muni de sa loi naturelle,
- l'ensemble des matrices $n \times n$ sur un anneau et plus généralement sur un semi-anneau (pas d'inverse pour l'addition),
- $M = \{1, \alpha, \beta\}$ où 1 est l'élément neutre et où la loi est définie pour les autres éléments par $\alpha\beta = \alpha^2 = \alpha$ et $\beta^2 = \beta\alpha = \beta$ (loi $xy = x$),
- $M = \{1, \alpha, \beta\}$ où 1 est l'élément neutre et où la loi est définie pour les autres éléments par $\alpha^2 = \beta\alpha = \alpha$ et $\alpha\beta = \beta^2 = \beta$ (loi $xy = y$),
- $M = \{1\} \cup I \times J$ pour deux ensembles I et J où la loi est $(i, j)(i', j') = (i, j')$.

Exercice 1.113. Déterminer à isomorphisme près tous les monoïdes à 2 et 3 éléments.

Solution. Pour décrire un monoïde, il suffit de donner sa table de multiplication. La ligne et la colonne de l'élément neutre sont déjà fixées.

Si le monoïde a deux éléments 1 et x , il reste à fixer le carré de x pour déterminer la table de multiplication. Si on fixe $x^2 = 1$, on trouve le groupe $\mathbb{Z}/2\mathbb{Z}$ et si on fixe $x^2 = x$, on trouve le monoïde $\{1, 0\}$ avec le produit usuel.

Si le monoïde a trois éléments 1, x et y , il reste à fixer les produits x^2 , xy , yx et y^2 . Si $xy = 1$ ou $yx = 1$, tous les éléments sont inversibles et on trouve le groupe $\mathbb{Z}/3\mathbb{Z}$. Si $x^2 = 1$ (ou symétriquement $y^2 = 1$), alors $\{1, x\}$ est le groupe $\mathbb{Z}/2\mathbb{Z}$ et on a nécessairement $xy = yx = y^2 = y$. Il reste donc le cas où $\{x, y\}$ est sous-semigroupe du monoïde. Comme il existe 5 semigroupes à 2 éléments à isomorphisme près, on trouve 5 nouveaux monoïdes dont les tables de multiplication restreintes à $\{x, y\}$ sont données

ci-dessous.

$$\begin{array}{c|c|c} \times & x & y \\ \hline x & x & y \\ \hline y & y & x \end{array} \quad \begin{array}{c|c|c} \times & x & y \\ \hline x & x & y \\ \hline y & y & y \end{array} \quad \begin{array}{c|c|c} \times & x & y \\ \hline x & x & x \\ \hline y & y & y \end{array} \quad \begin{array}{c|c|c} \times & x & y \\ \hline x & x & y \\ \hline y & x & y \end{array} \quad \begin{array}{c|c|c} \times & x & y \\ \hline x & y & y \\ \hline y & y & y \end{array}$$

Les deux premiers sont les deux monoïdes à 2 éléments, les deux suivants ont déjà été donnés à l'exemple 1.112 et le dernier est celui où le produit est constant.

Définition 1.114 (Morphisme de monoïdes). Soient M et M' deux monoïdes. Un *morphisme* de M dans M' est une application $\mu : M \rightarrow M'$ qui vérifie :

- $\mu(1_M) = 1_{M'}$,
- $\mu(xy) = \mu(x)\mu(y)$, pour tous éléments x et y de M .

Exemple 1.115. L'ensemble \mathbb{N} muni de l'addition est un monoïde. L'application $w \mapsto |w|$ est un morphisme de A^* dans \mathbb{N} .

La proposition suivante justifie le fait que le monoïde A^* soit appelé monoïde libre. Cette propriété caractérise le monoïde libre engendré par A .

Proposition 1.116. *Toute fonction $\mu : A \rightarrow M$ de A dans un monoïde M se prolonge de façon unique en un morphisme de monoïde de A^* dans M .*

Preuve. Si le morphisme $\hat{\mu}$ prolonge μ , on a pour tout mot $w = a_1 \cdots a_n$ de A^* ,

$$\hat{\mu}(w) = \hat{\mu}(a_1) \cdots \hat{\mu}(a_n) = \mu(a_1) \cdots \mu(a_n).$$

Ceci définit $\hat{\mu}$ de manière unique. Inversement, il est facile de vérifier que la formule ci-dessus définit bien un morphisme. \square

Définition 1.117 (Reconnaissance par monoïde). Un langage $L \subseteq A^*$ est *reconnu* par un morphisme $\mu : A^* \rightarrow M$ si et seulement si il existe une partie P de M telle que $L = \mu^{-1}(P)$. Par extension, un monoïde M *reconnaît* le langage L s'il existe un morphisme $\mu : A^* \rightarrow M$ qui reconnaît L .

Quand un morphisme de monoïdes $\mu : A^* \rightarrow M$ reconnaît un langage L , on peut toujours prendre la partie P égale à $P = \mu(L)$. Autrement dit, un morphisme $\mu : A^* \rightarrow M$ reconnaît L si et seulement si $L = \mu^{-1}(\mu(L))$. Si le morphisme μ n'est pas surjectif, on peut remplacer le monoïde M par le sous-monoïde $M' = \mu(A^*)$.

Exemple 1.118. On considère le monoïde M égal au groupe $\mathbb{Z}/2\mathbb{Z}$ à deux éléments.

- $\mu : A^* \rightarrow \mathbb{Z}/2\mathbb{Z}$ (dans ce cas : $\mu^{-1}(0) = (A^2)^*$)
 $w \mapsto |w| \bmod 2$
- $\mu : A^* \rightarrow \mathbb{Z}/2\mathbb{Z}$ (dans ce cas : $\mu^{-1}(0) = (ab^*a + b)^*$)
 $w \mapsto |w|_a \bmod 2$

Exemple 1.119. On considère $A = \{a, b\}$ et le monoïde $M = \{1, \alpha, \beta\}$ avec $\alpha^2 = \alpha\beta = \alpha$ et $\beta\alpha = \beta^2 = \beta$ (cf. exemple 1.112). Le morphisme $\mu : A^* \rightarrow M$ défini par :

$$\mu(w) = \begin{cases} 1 & \text{si } w = \varepsilon \\ \alpha & \text{si } w \in aA^* \\ \beta & \text{si } w \in bA^* \end{cases}$$

Proposition 1.120 (Rationalité par morphisme). *Un langage $L \subseteq A^*$ est rationnel si et seulement s'il est reconnu par un monoïde fini.*

Preuve. Soit un langage L et un morphisme $\mu : A^* \rightarrow M$ tel que $L = \mu^{-1}(P)$ pour une partie P de M . On construit l'automate $\mathcal{A} = (M, A, E, \{1\}, P)$ où l'ensemble des transitions est $E = \{(m, a, m\mu(a)) \mid m \in M, a \in A\}$. Cet automate reconnaît L , ce qui prouve un sens de l'équivalence. \square

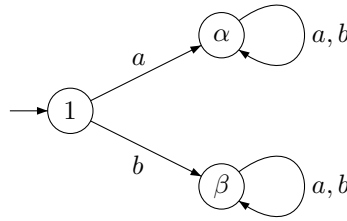


FIGURE 1.23 – Automate du monoïde $M = \{1, \alpha, \beta\}$ avec la loi $xy = x$

Exemple 1.121. Si on applique la construction précédente au monoïde $M = \{1, \alpha, \beta\}$ avec la loi $xy = x$ (cf. exemple 1.112), on obtient l'automate de la figure 1.23.

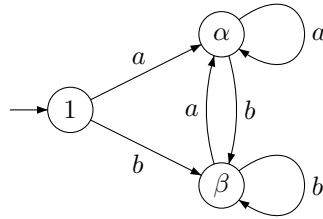


FIGURE 1.24 – Automate du monoïde $M = \{1, \alpha, \beta\}$ avec la loi $xy = y$

Exemple 1.122. Si on applique la construction précédente au monoïde $M = \{1, \alpha, \beta\}$ avec la loi $xy = y$ (cf. exemple 1.112), on obtient l'automate de la figure 1.24.

Pour montrer que tout langage rationnel est reconnu par un monoïde fini, on introduit quelques définitions. Soient r et r' deux relations binaires sur un même ensemble E . On définit la composition de ces deux relations, notée rr' par la formule suivante.

$$rr' = \{(x, z) \mid \exists y \in E (x, y) \in r \text{ et } (y, z) \in r'\}.$$

On vérifie sans peine que la loi de composition ainsi définie est associative et que l'identité est un élément neutre. On note \mathcal{R}_E le *monoïde des relations binaires* sur l'ensemble E muni de cette loi de composition interne.

Définition 1.123 (Monoïde des transitions). Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate. L'application $\mu : A^* \rightarrow \mathcal{R}_Q$ définie par :

$$\mu(w) = r_w = \{(q, q') \mid q \xrightarrow{w} q' \text{ dans } \mathcal{A}\}$$

est un morphisme de A^* dans \mathcal{R}_Q . Son image $\mu(A^*)$ dans \mathcal{R}_Q est appelée *monoïde des transitions* de l'automate \mathcal{A} .

Avec ce morphisme, il suffit de prendre :

$$P = \{r \mid r \cap (I \times F) \neq \emptyset\}$$

pour avoir le résultat recherché.

Définition 1.124 (Sous-monoïde). Un monoïde M' est appelé *sous-monoïde* de M lorsque $M' \subseteq M$ et si l'application identité $M' \xrightarrow{id} M$ est un morphisme de M' dans M .

Une façon équivalente d'exprimer que M' est un sous-monoïde de M est de dire que M' est inclus dans M , que l'unité de M' coïncide avec l'unité de M et que le produit de M' est la restriction à M' du produit de M .

Exemple 1.125. Quelques exemples de sous-monoïdes.

- Les sous-monoïdes de A^* sont les langages de la forme L^* pour $L \subseteq A^*$.
- L'ensemble des parties rationnelles de A^* est un sous-monoïde de $\mathfrak{P}(A^*)$.

Exercice 1.126. On dit qu'un langage est *préfixe* si $L \cap LA^+ = \emptyset$, c'est-à-dire si un préfixe stricte d'un mot de L n'est jamais un mot de L . Montrer que l'ensemble des langages préfixes est un sous-monoïde de $\mathfrak{P}(A^*)$ et que dans ce monoïde, l'égalité

$$K_1 \cdots K_m = L_1 \cdots L_n$$

n'est vérifiée que si $m = n$ et $K_i = L_i$ pour tout $1 \leq i \leq m$. Un tel monoïde est dit *libre* car il est isomorphe à un monoïde de la forme B^* .

Définition 1.127 (Monoïde quotient). On dit que M est *quotient* de M' s'il existe un morphisme surjectif $\mu : M' \rightarrow M$.

$$\begin{array}{ccc} M_1 & \xrightarrow{i} & M' \\ \downarrow \pi & & \\ M & & \end{array}$$

FIGURE 1.25 – Relation de division $M \triangleleft M'$

Définition 1.128 (Monoïde diviseur). On dit que M *divise* M' et on note $M \triangleleft M'$ si M est un quotient d'un sous-monoïde M_1 de M' (cf. figure 1.25).

la notion de *diviseur* pour des objets algébriques est l'équivalent pour les graphes de la notion de *mineur*.

Proposition 1.129 (Ordre de la division). *La relation de division est une relation d'ordre sur l'ensemble des monoïdes finis.*

On perd l'anti-symétrie lorsque les monoïdes sont infinis comme le montre l'exemple suivant.

Exemple 1.130. On peut illustrer ce fait avec $M = \{a, b\}^*$ et $M' = \{a, b, c, d\}^*$ (remarquer que $M \subsetneq M'$). Et en considérant les morphismes définis par :

$$\begin{aligned} \mu : M &\rightarrow M' & : w &\mapsto w \\ \mu' : M' &\rightarrow M & : \begin{cases} a \mapsto aa \\ b \mapsto bb \\ c \mapsto ab \\ d \mapsto ba \end{cases} \end{aligned}$$

Preuve. La réflexivité est évidente et l'anti-symétrie se vérifie facilement par équipotence.

Pour la transitivité, supposons que $M \triangleleft M' \triangleleft M''$, alors :

$$\begin{array}{ccccc} M_2 & \xrightarrow{id} & M'_1 & \xrightarrow{id} & M'' \\ \downarrow \pi' & & \downarrow \pi' & & \\ M_1 & \xrightarrow{id} & M' & & \\ \downarrow \pi & & & & \\ M & & & & \end{array}$$

On vérifie alors que le monoïde $M_2 = \pi'^{-1}(M_1)$ est un sous-monoïde de M'_1 et donc de M'' et que $\pi(\pi'(M_2)) = \pi(M_1) = M$. \square

Définition 1.131 (Congruence). Une relation d'équivalence \sim définie sur un monoïde M est appelée *congruence* (de monoïde) si pour tous x, x', y et y' dans M , on a :

$$\left. \begin{array}{l} x \sim x' \\ y \sim y' \end{array} \right\} \Rightarrow xy \sim x'y'$$

Pour qu'une relation d'équivalence \sim soit une congruence, il suffit que pour tous y et y' dans M , l'implication $y \sim y' \implies \forall x, z \, xyz \sim xy'z$ soit satisfaite.

Proposition 1.132. *Si \sim est une congruence sur M , le quotient M/\sim est un monoïde pour la loi définie par $[x][y] = [xy]$ où $[x]$ désigne la classe de x dans le quotient.*

Définition 1.133 (Congruence syntaxique). Pour tout langage $L \subseteq A^*$, on appelle *contexte* de $y \in A^*$ l'ensemble $C(y) = \{(x, z) \in (A^*)^2 : xyz \in L\}$. La relation \sim_L définie par :

$$y \sim_L y' \Leftrightarrow C(y) = C(y') \Leftrightarrow \forall x, z \in A^* (xyz \in L \Leftrightarrow xy'z \in L)$$

est appelée *congruence syntaxique* de L .

La proposition suivante justifie la terminologie.

Proposition 1.134 (Monoïde syntaxique). *Pour tout langage L , la congruence syntaxique \sim_L est effectivement une congruence et le monoïde quotient A^*/\sim_L est appelé monoïde syntaxique de L .*

Preuve. Si $y \sim_L y'$, on a $\forall x, z \in A^*$, $C(xyz) = C(xy'z)$ et donc $xyz \sim_L xy'z$. \square

la proposition suivante montre que le monoïde syntaxique d'un langage est en quelque sorte l'équivalent algébrique de l'automate minimal. Il est le plus petit monoïde reconnaissant le langage. Beaucoup de propriétés d'un langage peuvent être déduites de son monoïde syntaxique.

Proposition 1.135. *Un monoïde M reconnaît un langage $L \subseteq A^*$ si et seulement si le monoïde syntaxique $M(L)$ divise M .*

On commence par prouver un premier lemme qui montre que la relation de division est compatible avec la reconnaissabilité.

Lemme 1.136. *Pour tout langage $L \subseteq A^*$ et tout monoïde M , les propositions suivantes sont vérifiées.*

1. *Si M reconnaît L si M est sous-monoïde de M' , alors M' reconnaît L .*
2. *Si M reconnaît L si M est quotient de M' , alors M' reconnaît L .*
3. *Si M reconnaît L si M divise M' , alors M' reconnaît L .*

Preuve. La proposition 1. découle du fait qu'un morphisme de A^* dans M peut être vu comme un morphisme de A^* dans M' . La proposition 3. est conséquence de 1. et de 2.

Pour la proposition 2, soit μ un morphisme de A^* dans M et π un morphisme surjectif de M' dans M . Pour toute lettre a de A , on pose $\mu'(a) = m_a$ où m_a est un élément quelconque de M' vérifiant $\pi(m_a) = \mu(a)$. D'après la proposition 1.116, μ' se prolonge en un morphisme de A^* dans M' qui vérifie $\pi(\mu'(w)) = \mu(w)$. Il est alors facile de vérifier que la partie $P' = \pi'^{-1}(\mu(L))$ de M' vérifie $\mu'^{-1}(P') = L$ et que μ' reconnaît L . \square

Ce second lemme établit qu'un langage est bien reconnu par son monoïde syntaxique.

Lemme 1.137. *Pour tout langage L , le monoïde syntaxique $M(L)$ reconnaît L .*

Preuve. Il faut remarquer qu'un mot w appartient à L si et seulement si son contexte $C(w)$ contient la paire $(\varepsilon, \varepsilon)$. Il est alors clair que si une classe de la congruence syntaxique \sim_L contient un mot de L , elle est alors entièrement contenu dans L .

L'application canonique de A^* dans $M(L) = A^*/\sim$ qui associe à tout mot w sa classe $[w]$ est un morphisme. D'après la remarque précédente, ce morphisme reconnaît L puisque $L = \bigcup_{w \in L} [w]$. \square

On procède maintenant à la preuve de la proposition.

Preuve. En utilisant les lemmes 1.137 et 1.136, on obtient immédiatement que, si $M(L)$ divise M , alors M reconnaît aussi L .

Réciproquement, si M reconnaît L , il existe un morphisme $\mu : A^* \rightarrow M$ tel que $L = \mu^{-1}(P)$ pour une partie $P \subseteq M$. On note π la morphisme canonique de A^* dans $M(L)$. Le monoïde $M' = \mu(A^*)$ est un sous-monoïde de M . On va montrer que $M(L)$ est un quotient de M' ce qui prouvera que $M(L)$ divise M .

On montre que pour tous mots w et w' , si $\mu(w) = \mu(w')$, alors on a aussi $\pi(w) = \pi(w')$. Une paire (u, v) appartient au contexte $C(w)$ si uwv appartient à L , c'est-à-dire si $\mu(u)\mu(w)\mu(v)$ appartient à P . Ceci montre que les contextes $C(w)$ et $C(w')$ sont égaux et que $\pi(w) = \pi(w')$.

Pour tout élément m de M' , on peut définir $\pi'(m) = \pi(w)$ où w est un mot tel que $\mu(w) = m$ puisque $\pi(w)$ est indépendant du choix de w . On vérifie alors sans difficulté que π' ainsi défini est un morphisme surjectif de M' dans $M(L)$.

$$\begin{array}{ccc} A^* & \xrightarrow{\mu} & M' \subseteq M \\ \downarrow \pi & \swarrow \pi' & \\ M(L) & & \end{array}$$

□

Pour le calcul du monoïde syntaxique, on utilise la proposition suivante.

Proposition 1.138. *Le monoïde syntaxique $M(L)$ d'un langage rationnel L est égal au monoïde des transitions de l'automate minimal de L .*

Preuve. D'après la proposition 1.120, le monoïde des transitions de l'automate minimal de L reconnaît L . Il est donc divisible par le monoïde syntaxique $M(L)$ d'après la proposition 1.135. Soient deux mots w et w' ayant des images différentes dans le monoïde des transitions de l'automate minimal. Puisque cet automate est déterministe, il existe un état p tel que $p \cdot w \neq p \cdot w'$. Soient q et q' les états $p \cdot w$ et $p \cdot w'$. Puisque cet automate est minimal, il existe un mot v tel que $q \cdot v$ est final et $q' \cdot v$ n'est pas final (ou l'inverse). Il existe aussi un mot u tel que $i \cdot u = p$. On en déduit que la paire (u, v) appartient à $C(w)$ mais pas à $C(w')$ et que w et w' ont des images différentes dans le monoïde syntaxique de L . □

Exemple 1.139. Soit le langage $L = (ab)^*$ dont l'automate minimal est représenté à la figure 1.22. On construit son monoïde syntaxique en déterminant toutes les relations de transitions de l'automate. Comme l'automate est déterministe, ces relations sont des fonctions. On obtient la table ci-dessous.

	0	1	2
$1 = \varepsilon$	0	1	2
$aba = a$	1	2	2
$bab = b$	2	0	2
$0 = bb = aa$	2	2	2
ab	0	2	2
ba	2	1	2
bb	2	2	2
aba	1	2	2
bab	2	0	2

TABLE 1.1 – Fonctions de transitions de l'automate minimal de $(ab)^*$

Le monoïde associé comporte 6 éléments $\{1, 0, \alpha, \beta, \alpha\beta, \beta\alpha\}$ qui vérifient les relations suivantes. Ceci constitue en fait une présentation par générateurs et relations de ce monoïde.

$$\alpha\beta\alpha = \alpha \quad \beta\alpha\beta = \beta \quad \alpha^2 = \beta^2 = 0$$

La reconnaissance par monoïdes des langages rationnels permet de résoudre rapidement certaines questions. Pour illustrer le propos, nous allons montrer la proposition suivante dont la preuve est grandement facilitée par les monoïdes. Le plus intéressant n'est pas le résultat en soi mais la technique de preuve. Le résultat est en fait un cas particulier d'une construction beaucoup plus générale. Pour la définition d'une substitution cf. définition 2.17.

Proposition 1.140. *Soit σ une substitution de A^* dans B^* et soit $K \subset B^*$ un langage rationnel. Les deux langages $\{w \mid \sigma(w) \cap K \neq \emptyset\}$ et $\{w \mid \sigma(w) \subseteq K\}$ sont rationnels.*

Pour un monoïde M , l'ensemble $\mathfrak{P}(M)$ des parties de M est naturellement muni d'une structure de monoïde par le produit $P \cdot Q = \{pq \mid p \in P \text{ et } q \in Q\}$ pour deux parties $P, Q \subseteq M$.

Preuve. On va montrer que si K est reconnu par le monoïde M , alors les deux langages $L = \{w \mid \sigma(w) \cap K \neq \emptyset\}$ et $L' = \{w \mid \sigma(w) \subseteq K\}$ sont reconnus par le monoïde $\mathfrak{P}(M)$. Soit $\mu : B^* \rightarrow M$ un morphisme de B^* dans un monoïde fini M tel que $K = \mu^{-1}(P)$ pour une partie $P \subseteq M$. On définit le morphisme $\hat{\mu} : A^* \rightarrow \mathfrak{P}(M)$ par $\hat{\mu}(a) = \mu(\sigma(a))$. On vérifie que $\hat{\mu}(w) = \mu(\sigma(w))$ pour tout mot $w \in A^*$. Ceci implique que $L = \hat{\mu}^{-1}(Q)$ et $L' = \hat{\mu}^{-1}(Q')$ où Q et Q' sont respectivement les ensembles $\{T \mid T \cap P \neq \emptyset\}$ et $\{T \mid T \subseteq P\}$ des parties de M qui intersectent P et qui sont incluses dans P . \square

1.12 Langages sans étoile

Le but de cette partie est d'illustrer l'utilisation du monoïde syntaxique et de montrer la puissance de cet outil. On introduit la classe des langages appelés *sans étoile* et on prouve le théorème de Schützenberger qui donne une élégante caractérisation de ces langages en terme de groupes contenus dans leurs monoïdes syntaxiques.

Définition 1.141 (Langage sans étoile). La famille des *langages sans étoile* est la plus petite famille \mathcal{E} telle que :

- i) $\emptyset \in \mathcal{E}$ et $\{a\} \in \mathcal{E}$ pour toute lettre a ;
- ii) \mathcal{E} est close par union, complémentation et produit. Ceci signifie que si les langages K et L appartiennent à \mathcal{E} , alors les langages $K + L$, $A^* \setminus L$ et KL appartiennent aussi à \mathcal{E} .

Les langages sans étoile sont aussi les langages de hauteur d'étoile généralisée 0 (cf. section 1.10 p. 57).

Exemple 1.142. Quelques exemples et contre-exemples de langages sans étoile.

- $A^* = A^* \setminus \emptyset$ est un langage sans étoile.
- $A^*aA^*bA^*$ est sans-étoile.
- $\{\varepsilon\} = A^* \setminus \bigcup_{a \in A} A^*aA^*$ est sans étoile.
- $(ab)^+ \in \mathcal{E}$ car il s'écrit aussi $(ab)^+ = (aA^* \cap A^*b) \setminus A^*(aa + bb)A^*$.
- $(ab)^* \in \mathcal{E}$ car il s'écrit $(ab)^* = (ab)^+ + \varepsilon$.
- $(aa)^*$ et $(aa)^+$ ne sont pas sans étoile.

Le fait que le langage $(aa)^*$ ne soit effectivement pas sans étoile n'est pas évident *a priori*. Il est facile de montrer qu'un langage est sans étoile en donnant explicitement une expression sans étoile pour ce langage. Par contre, il est plus ardu de prouver qu'un

langage n'est pas sans étoile. La suite de cette partie est consacrée à une caractérisation algébrique des langages sans étoile qui permet facilement de prouver qu'un langage donné n'est pas sans étoile.

Définition 1.143 (Groupe contenu dans un monoïde). On dit qu'un groupe G est *contenu* dans un monoïde M si :

1. $G \subseteq M$
2. $\forall (g, g') \in G^2, g \cdot_G g' = g \cdot_M g'$

Un monoïde M est *apériodique* s'il ne contient que des groupes triviaux, c'est-à-dire réduits à un seul élément.

Lorsqu'un groupe G est contenu dans un monoïde M , l'élément neutre de G ne coïncide par nécessairement avec l'élément neutre de M . Lorsque les éléments neutres sont identiques, on parle de *sous-groupe* d'un monoïde.

Les résultats suivants montrent que les monoïdes apériodiques sont à l'opposé des groupes. Soit x un élément d'un monoïde M . On considère l'ensemble $\{x^k \mid k \geq 0\}$ des puissances de x qui est en fait le sous-monoïde engendré par x . Si M est fini, il existe deux entiers $l < k$ tels que $x^k = x^l$. Si k est en outre choisi le plus petit possible, les éléments $1, x, x^2, \dots, x^{k-1}$ sont distincts et on a le diagramme suivant, communément appelé *poêle à frire*.

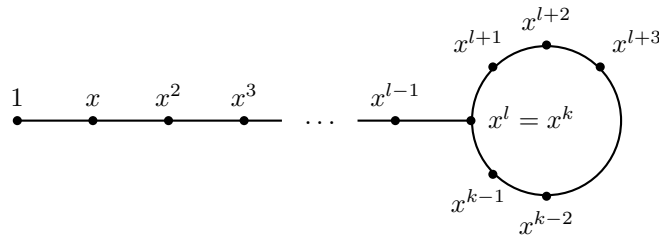


FIGURE 1.26 – Diagramme de la poêle à frire

Si k est le plus petit entier pour lequel il existe $l < k$ tel que $x^k = x^l$, les entiers k et l sont uniques. On note alors $l_x = l$ et $p_x = k - l$. Le lemme suivant caractérise les groupes et les monoïdes apériodiques parmi les monoïdes finis.

Lemme 1.144. *Pour un monoïde fini M , les trois propriétés suivantes sont équivalentes.*

1. M est un groupe.
2. Pour tout $x \in M$, on a $l_x = 0$ et donc $x^{p_x} = 1$.
3. Il existe un entier ω tel que $x^\omega = 1$ pour tout $x \in M$.

Pour un monoïde fini M , les trois propriétés suivantes sont équivalentes.

1. M est apériodique.
2. Pour tout $x \in M$, on a $p_x = 1$ et donc $x^{l_x+1} = x^{l_x}$.
3. Il existe un entier ω tel que $x^{\omega+1} = x^\omega$ pour tout $x \in M$.

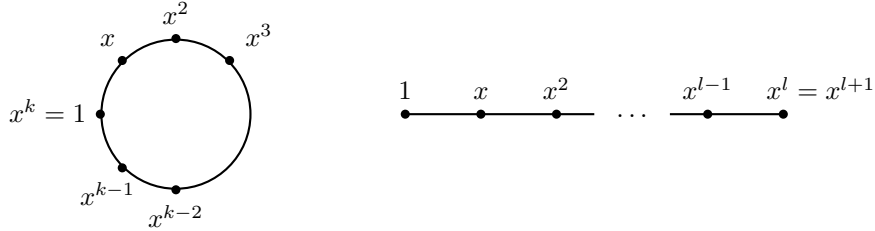


FIGURE 1.27 – Cas des groupes et des monoïdes aperiodiques

L'énoncé de ce lemme se traduit en termes imagés de la manière suivante. Dans un groupe, la poêle à frire est donc *sans manche* alors qu'au contraire, elle est *réduite au manche* dans un monoïde aperiodique.

Le cas des groupes a été énoncé pour bien montrer l'orthogonalité entre les deux cas mais il n'est pas utilisé dans la suite. La preuve est laissée à titre d'exercice.

Preuve. Pour le cas des monoïdes aperiodiques, la preuve que la condition 1 implique la condition 2 découle de la remarque suivante. L'ensemble $G_x = \{x^{l_x}, x^{l_x+1}, \dots, x^{l_x+p_x-1}\}$ est un monoïde isomorphe à $\mathbb{Z}/p_x\mathbb{Z}$. Son élément neutre est x^m où m est l'unique entier entre l_x et $l_x + p_x - 1$ tel que $m \equiv 0 \pmod{p_x}$. Si M est aperiodique, on a donc nécessairement $p_x = 1$.

La preuve que la condition 2 implique la condition 3 est obtenue en posant $\omega = \max\{l_x \mid x \in M\}$.

Il reste à montrer que la condition 3 implique la condition 1. Si G est un groupe contenu dans M , on a l'égalité $g^\omega = g^{\omega+1}$. Cette égalité implique immédiatement que g est l'identité de G et que le groupe G est trivial. \square

Exemple 1.145. Le monoïde syntaxique du langage $(aa)^+$ est constitué des trois éléments $\{1, \alpha, \alpha^2\}$ avec la relation $\alpha^3 = \alpha$. Le sous-ensemble $\{\alpha, \alpha^2\}$ est isomorphe au groupe $\mathbb{Z}/2\mathbb{Z}$.

Le théorème suivant caractérise en terme de monoïde syntaxique les langages sans étoile. Il permet en particulier de décider effectivement si un langage donné est sans étoile et de calculer explicitement une expression sans étoile si c'est possible.

Théorème 1.146 (Schützenberger 1965). *Pour tout langage rationnel L , les trois conditions suivantes sont équivalentes.*

1. *Le langage L est sans étoile.*
2. *Le monoïde syntaxique $M(L)$ de L est aperiodique.*
3. *Le langage L est reconnu par un monoïde aperiodique.*

Le monoïde syntaxique du langage $(aa)^*$ est le groupe $\mathbb{Z}/2\mathbb{Z}$ qui n'est bien sûr pas aperiodique. Le théorème précédent montre donc que ce langage n'est pas sans étoile.

On peut remarquer qu'il suffit qu'un langage soit reconnu par un monoïde aperiodique pour que son monoïde syntaxique soit aperiodique. On vérifie sans difficulté qu'un monoïde qui divise un monoïde aperiodique est encore aperiodique. La preuve du théorème se décompose en les sept lemmes techniques suivants.

Le lemme suivant traduit directement en terme de langages le résultat du lemme 1.144. La preuve est immédiate en utilisant la définition de la congruence syntaxique.

Lemme 1.147. *Le monoïde syntaxique d'un langage rationnel L est apériodique si et seulement si il existe un entier ω tel que*

$$\forall x, y, z \in A^* \quad xy^\omega z \in L \iff xy^{\omega+1}z \in L.$$

Pour un langage tel que $M(L)$ apériodique, on appelle *indice* et on note $i(L)$ le plus petit entier ω tel que $\forall x, y, z \in A^* \quad xy^\omega z \in L \iff xy^{\omega+1}z \in L$.

Lemme 1.148. *Pour tous langages $K, L \subseteq A^*$ rationnels, on a*

$$\begin{aligned} i(\{a\}) &= 2 \\ i(K + L) &\leq \max(i(K), i(L)) \\ i(KL) &\leq i(K) + i(L) + 1 \\ i(A^* \setminus K) &= i(K) \end{aligned}$$

Preuve. Seule la relation $i(KL) \leq i(K) + i(L) + 1$ ne découle pas directement des définitions. Supposons que le mot $xy^n z$ appartienne à KL . Il se factorise donc $xy^n z = uv$ où u et v appartiennent respectivement à K et à L . Si $n \geq i(K) + i(L) + 1$, soit le nombre d'occurrences de y dans u est supérieur à $i(K)$, soit le nombre d'occurrences de y dans v est supérieur à $i(L)$. Dans les deux cas, on conclut en utilisant la définition de $i(K)$ ou de $i(L)$. \square

Le lemme précédent permet de montrer par récurrence sur (la taille de) l'expression rationnelle sans étoile que le monoïde syntaxique d'un langage sans étoile est apériodique. La réciproque est plus technique.

Dans un monoïde apériodique, l'élément neutre est le seul élément inversible puisque l'ensemble des éléments inversibles est un groupe. Le lemme suivant raffine cette remarque.

Lemme 1.149 (Simplification). *Soient p, m et q trois éléments d'un monoïde apériodique fini. L'égalité $m = pmq$ implique les deux égalités $m = pm = mq$.*

Une conséquence directe du lemme est que l'élément neutre est le seul élément inversible à droite (ou à gauche). Si on a $mp = 1_M$, alors on a aussi $mpm = m$ et $mp = m$ d'après le lemme.

Preuve. D'après le lemme 1.144, il existe un entier ω tel que $s^\omega = s^{\omega+1}$ pour tout élément s du monoïde. On écrit alors les égalités

$$m = pmq = p^\omega m q^\omega = p^\omega m q^{\omega+1} = mq$$

et on procède pareillement pour $m = pm$. \square

Le lemme précédent permet de montrer le lemme suivant.

Lemme 1.150. *Pour tout élément m d'un monoïde apériodique fini M , on a l'égalité*

$$\{m\} = (mM \cap Mm) \setminus J \quad \text{où} \quad J = \{x \in M \mid m \notin MxM\}.$$

Preuve. Tout d'abord m appartient à $mM \cap Mm$ mais il n'appartient pas à J puisque $m \in MmM$. On en déduit l'inclusion $\{m\} \subseteq (mM \cap Mm) \setminus J$ qui est vraie même si M n'est pas apériodique.

On montre maintenant l'inclusion inverse. Soit maintenant m' un élément de $(mM \cap Mm) \setminus J$. Puisque $m' \in mM \cap Mm$, il existe deux éléments p et q de M tels que $m' = mp = qm$. Puisque m' n'appartient pas à J , il existe aussi deux éléments s et r de M tels que $m = sm'r$. En combinant $m' = mp$ et $m = sm'r$, on obtient $m = smpr$ qui donne $m = mpr$ par le lemme de simplification et donc $m = m'r$. En combinant cette dernière égalité avec $m' = qm$, on obtient $m' = qm'r$ et donc $m' = m'r$ par le lemme de simplification. \square

Lemme 1.151. *Soit $\mu : A^* \rightarrow M$ un morphisme dans un monoïde apériodique fini M et soit $m \neq 1_M$ un élément de M différent de l'élément neutre. On a alors l'égalité*

$$\mu^{-1}(m) = (UA^* \cap A^*V) \setminus (A^*WA^*)$$

où les trois ensembles $U, V, W \subset A^*$ de mots sont donnés par

$$U = \bigcup_{\substack{n\mu(a)M=mM \\ n \notin mM}} \mu^{-1}(n)a, \quad V = \bigcup_{\substack{Mm=M\mu(a)n \\ n \notin Mm}} a\mu^{-1}(n)$$

et

$$W = \{a \mid m \notin M\mu(a)M\} \cup \left(\bigcup_{\substack{m \in M\mu(a)nM \cap Mn\mu(b)M \\ m \notin M\mu(a)n\mu(b)M}} a\mu^{-1}(n)b \right).$$

Preuve. Puisque $m \neq 1_M$, les deux ensembles mM et Mm ne contiennent pas l'élément neutre 1_M d'après le lemme de simplification.

D'après le lemme précédent, un mot w vérifie $\mu(w) = m$ si et seulement si $\mu(w) \in mM \cap Mm$ et $m \in M\mu(w)M$.

On commence par montrer que $\mu(w) \in mM$ si et seulement si w se factorise uav tel que $n = \mu(u)$ vérifie $n \notin mM$ et $n\mu(a)M = mM$. L'appartenance $\mu(w) \in Mm$ se traite de manière symétrique. Soit u le plus long préfixe de w tel que $\mu(u) \notin mM$. Ce mot u existe puisque le mot vide est un préfixe de w et $\mu(\varepsilon) = 1_M$ n'appartient pas à mM . Ce mot u est par définition différent de w . Le mot w se factorise $w = uav$ tel que $n = \mu(u)$ vérifie $n \notin mM$ et $n\mu(a)M = mM$.

Soit w un mot tel que $m \notin M\mu(w)M$. On montre que w se factorise soit $w = uav$ où $m \notin M\mu(a)M$ soit $w = uavbt$ tel que $n = \mu(v)$ vérifie $m \in M\mu(a)nM \cap Mn\mu(b)M$ et $m \notin M\mu(a)n\mu(b)M$. Soit v' un plus petit facteur de w tel que $m \notin M\mu(v')M$. Un tel mot existe car $m \notin M\mu(w)M$. Si v' est réduit à une lettre a , on a $w = uav$ où $m \notin M\mu(a)M$. Si v' est de longueur au moins 2, v' s'écrit $v' = avb$ avec les propriétés requises. \square

Pour tout m , on pose $r(m) = |MmM|$. La preuve que $\mu^{-1}(m)$ est un langage sans étoile est faite par récurrence sur $|M| - r(m)$:

– si $r(m) = |M|$, alors $m = 1_M$ et on a

$$\mu^{-1}(1_M) = \{a \mid \mu(a) = 1_M\}^* = A^* \setminus A^*\{a \mid \mu(a) \neq 1_M\}A^*$$

– sinon, on écrit $\mu^{-1}(m) = (UA^* \cap A^*V) \setminus A^*WA^*$ grâce au lemme précédent. On applique alors l'hypothèse de récurrence aux langages U , V et W .

Pour que cette preuve fonctionne, il reste à prouver les résultats suivants qui assure que les valeurs de n utilisées dans les définitions de U , V et W du lemme précédent vérifient bien $r(n) > r(m)$ et que l'hypothèse de récurrence peut être appliquée.

Lemme 1.152. *Pour tous éléments m et n d'un monoïde apériodique fini M , on a l'implication*

$$\left. \begin{array}{l} m \in nM \\ n \notin mM \end{array} \right\} \implies r(n) > r(m).$$

Preuve. De la relation $m \in nM$, on tire l'inclusion $MmM \subset MnM$. Pour montrer que cette inclusion est stricte, on montre que $n \notin MmM$. Puisque $m \in nM$, il existe un élément p tel que $np = m$. Si on suppose par l'absurde que $n \in MmM$, il existe deux éléments s et r tels que $n = smr$. En combinant ces deux égalités, on déduit que $n = snpr$ et donc $n = npr$ par le lemme de simplification. Ceci conduit à $n = mr$ qui contredit $n \notin mM$. Ceci prouve l'inégalité $r(n) > r(m)$. \square

Lemme 1.153. *Pour tous éléments m , n , α et β d'un monoïde apériodique fini M , on a l'implication*

$$\left. \begin{array}{l} m \in M\alpha nM \cap Mn\beta M \\ m \notin M\alpha n\beta M \end{array} \right\} \implies r(n) > r(m).$$

Preuve. De la relation $m \in M\alpha nM$, on tire l'inclusion $MmM \subset MnM$. Pour montrer que cette inclusion est stricte, on montre que $n \notin MmM$. Puisque $m \in M\alpha nM \cap Mn\beta M$, il existe quatre éléments p , q , p' et q' tels que $m = p\alpha nq = p'n\beta q'$. Si on suppose par l'absurde que $n \in MmM$, il existe deux éléments s et r tels que $n = smr$. En combinant cette dernière égalité avec $m = p'n\beta q'$, on obtient $n = sp'n\beta q'r$ et donc $n = n\beta q'r$ par le lemme de simplification. En substituant dans $m = p\alpha nq$, on obtient $m = p\alpha n\beta q'r$ qui contredit $m \notin M\alpha n\beta M$. Ceci prouve l'inégalité $r(n) > r(m)$. \square

On pourra consulter [Pin84] pour un exposé plus complet.

Exemple 1.154. Soit L le langage $(ab)^*$. Son monoïde syntaxique est égal à $M = \{1, \alpha, \beta, \alpha\beta, \beta\alpha, 0\}$ avec les relations $\alpha\beta\alpha = \alpha$, $\alpha^2 = 0$, $\beta\alpha\beta = \beta$ et $\beta^2 = 0$. En appliquant les formules ci-dessus, on obtient

$$\begin{aligned} \mu^{-1}(1) &= \{\varepsilon\} \\ \mu^{-1}(0) &= A^*(aa + bb)A^* \\ \mu^{-1}(\alpha\beta) &= (ab)^+ = (aA^* \cap A^*b) \setminus A^*(aa + bb)A^*. \end{aligned}$$

Exercice 1.155. Donner une expression sans étoile pour le langage $(ab + ba)^*$.

Solution. On calcule le monoïde syntaxique du langage $(ab + ba)^*$. Ce monoïde a 14 éléments et il est apériodique. En appliquant l'algorithme donné par la preuve du théorème du Schützenberger, on obtient la formule

$$\begin{aligned} (ab + ba)^* &= (ab)^* + (ba)^* + (K \setminus L) \quad \text{où } K \text{ et } L \text{ sont donnés par} \\ K &= ((ab)^+b + (ba)^+a)A^* \cap A^*(a(ab)^+ + b(ba)^+), \\ L &= A^*(a(ab)^*aa + b(ba)^*bb)A^*. \end{aligned}$$

1.13 Compléments

1.13.1 Conjecture de Černý

On s'intéresse ici à un problème ouvert connu sous le nom de conjecture de Černý. Pour un état p d'un automate déterministe \mathcal{A} et un mot w , on note $p \cdot w$ l'unique état q , s'il existe, tel qu'il y ait un chemin dans \mathcal{A} de p à q étiqueté par w . Un automate déterministe \mathcal{A} est dit *synchronisant* s'il existe un mot w tel que $p \cdot w$ est constant pour tout état p de \mathcal{A} . La lecture du mot w amène l'automate dans un unique état q quel que soit l'état de départ. Le mot w est alors dit *synchronisant* pour \mathcal{A} .

Une question naturelle est de savoir la longueur minimale d'un mot synchronisant d'un automate synchronisant à n états. La conjecture de Černý énonce qu'un tel mot est de longueur au plus $(n-1)^2$. Cette borne ne peut être améliorée. Pour tout entier n , il existe un automate synchronisant \mathcal{C}_n dont le mot synchronisant le plus court est de longueur $(n-1)^2$. La meilleure borne connue pour l'instant est $(n-1)^3/6$ même si la conjecture a déjà été prouvée pour des familles particulières d'automates. Cette conjecture est particulièrement fascinante. Son énoncé est relativement simple mais elle résiste malgré de nombreux efforts.

L'automate \mathcal{C}_n est défini de la manière suivante. Son ensemble d'états est $Q_n = \{0, \dots, n-1\}$. La lettre a laisse invariants tous les états sauf l'état 0 qui est envoyé sur 1. La lettre b effectue une permutation circulaire des états.

$$q \cdot a = \begin{cases} 1 & \text{si } q = 0 \\ q & \text{sinon} \end{cases} \quad \text{et} \quad q \cdot b = \begin{cases} q+1 & \text{si } q < n \\ 0 & \text{si } q = n \end{cases}$$

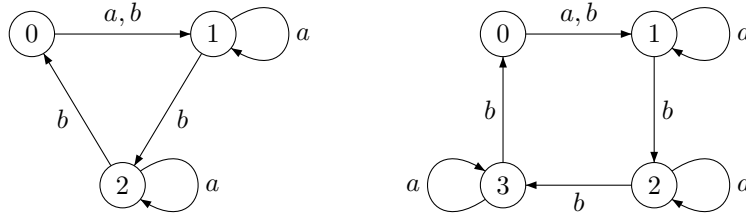


FIGURE 1.28 – Les automates \mathcal{C}_3 et \mathcal{C}_4

Exemple 1.156. Les automates \mathcal{C}_3 et \mathcal{C}_4 de la figure 1.28 sont synchronisants. Ils sont effectivement synchronisés par les mots ab^2a et ab^3ab^3a de longueur 4 et 9. Il est un peu plus délicat de prouver que ce sont les mots synchronisants les plus courts de \mathcal{C}_3 et \mathcal{C}_4 . On montre de même que le mot synchronisant le plus court de \mathcal{C}_n est le mot $(ab^{n-1})^{n-2}a$ de longueur $(n-1)^2$.

1.13.2 Rationnels d'un monoïde quelconque

Jusqu'à maintenant, on a uniquement étudié les parties des monoïdes libres. La définition d'une partie rationnelle s'étend à n'importe quel monoïde. Il est intéressant de considérer d'autres monoïdes comme les monoïdes $A^* \times B^*$ ou plus généralement $A_1^* \times \dots \times A_k^*$, les monoïdes commutatifs libres ou le groupe libre.

Les opérations rationnelles s'étendent de manière évidente aux parties d'un monoïde quelconque. Pour des parties K et L d'un monoïde M , le produit KL et l'étoile K^* sont définies de la manière suivante.

$$KL = \{kl \mid k \in K \text{ et } l \in L\} \quad \text{et} \quad K^* = \{k_1 \cdots k_n \mid k_1, \dots, k_n \in K\}.$$

L'étoile K^* est en fait le plus petit sous-monoïde de M contenant K et il est appelé le sous-monoïde *engendré* par K . La notation comporte une certaine ambiguïté puisque K^* peut noter le monoïde libre sur l'alphabet K où le sous-monoïde engendré par K dans M . Cette confusion est sans conséquence dans la mesure où on identifie très souvent une suite finie k_1, \dots, k_n avec son produit $k_1 \cdots k_n$ dans M . L'application qui envoie toute suite k_1, \dots, k_n sur $k_1 \cdots k_n$ est d'ailleurs un morphisme du monoïde libre K^* dans le monoïde M .

Définition 1.157. Soit M un monoïde. La famille notée $\text{Rat } M$ des *parties rationnelles* est la plus petite famille de parties de M telle que

- $\emptyset \in \text{Rat } M$ et $\{m\} \in \text{Rat } M$ pour tout élément m de M ;
- $\text{Rat } M$ est close pour les opérations rationnelles (l'union, le produit et l'étoile).

Dans le cas où le monoïde M est finiment engendré, on peut se contenter, dans la définition précédente, de supposer que les singletons $\{m_1\}, \dots, \{m_n\}$ sont rationnels où $\{m_1, \dots, m_n\}$ est une partie génératrice de M .

Un automate sur un monoïde M est un automate dont les étiquettes des transitions sont des éléments de M . Plus formellement, un automate \mathcal{A} sur M est un quintuplet (Q, M, E, I, F) où Q est un ensemble fini d'états et E est un sous-ensemble de $Q \times M \times Q$. L'*étiquette* d'un chemin

$$q_0 \xrightarrow{m_1} q_1 \xrightarrow{m_2} \cdots \xrightarrow{m_n} q_n$$

est le produit $m_1 \cdots m_n$ dans M des étiquettes des transitions qui constituent le chemin. Le *comportement* d'un automate est l'ensemble des étiquettes des chemins acceptants (c'est-à-dire commençant dans un état initial et se terminant dans un état final). Le théorème de Kleene s'étend facilement à un monoïde quelconque. La preuve faite dans le cas d'un monoïde libre reste valide.

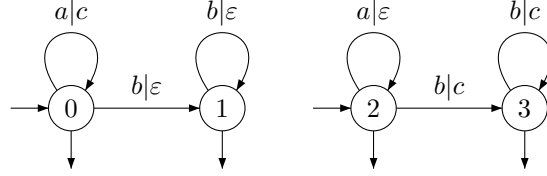
Théorème 1.158. Une partie d'un monoïde M est rationnelle si et seulement si elle est le comportement d'un automate sur M .

La construction du groupe libre ainsi que ses parties rationnelles sont décrites à la section 2.7.3 (p. 119).

Pour illustrer ces notions, on va donner quelques exemples dans le cas d'un monoïde $A^* \times B^*$ pour deux alphabets A et B . Une partie rationnelle de $A^* \times B^*$ est aussi appelée une *relation rationnelle* ou encore une *transduction rationnelle*. Un élément de $A^* \times B^*$ est une paire (u, v) de mots sur A et B qui est souvent notée $u|v$ en particulier pour les étiquettes des automates.

Exemple 1.159. La partie rationnelle $(a, c)^*(b, \varepsilon)^* + (a, \varepsilon)^*(b, c)^*$ du monoïde $\{a, b\}^* \times \{c\}^*$ est le comportement du transducteur de la figure 1.29.

Exemple 1.160. L'automate de l'addition donné à la figure 3.12 (p. 165) est en fait un transducteur dont le comportement est la partie du monoïde $\{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$ constituée des triplets (u, v, w) tels que la valeur binaire de w est égale à la somme des valeurs binaires de u et v .

FIGURE 1.29 – Un transducteur pour $(a, c)^*(b, \varepsilon)^* + (a, \varepsilon)^*(b, c)^*$

Une partie K d'un monoïde M est dite *reconnaissable* s'il existe un morphisme μ de M dans un monoïde fini N et une partie P de N tels que $K = \mu^{-1}(P)$. L'ensemble des parties reconnaissables de M est notée $\text{Rec } M$. Il découle directement de la définition que la famille des parties reconnaissables est close pour les opérations booléennes (union, intersection et complémentation). Un monoïde est dit *finiment engendré* s'il est égal au sous-monoïde engendré par une de ses parties finies.

Proposition 1.161. *Si M est finiment engendré, alors $\text{Rec } M \subseteq \text{Rat } M$.*

Les deux familles $\text{Rec } M$ et $\text{Rat } M$ coïncident lorsque M est un monoïde libre mais l'inclusion de la proposition précédente peut être stricte si M n'est pas libre. Si M est un monoïde de la forme $A^* \times B^*$, les parties reconnaissables sont les unions finies d'ensembles de la forme $K \times L$ où K et L sont respectivement des parties rationnelles de A^* et B^* . La diagonale $\Delta = \{(w, w) \mid w \in A^*\} = \{(a, a) \mid a \in A\}^*$ est une partie rationnelle de $A^* \times A^*$ qui n'est pas reconnaissable.

Preuve. Soit A un ensemble fini qui engendre M et soit μ un morphisme de M dans un monoïde fini N tel que $K = \mu^{-1}(P)$ pour une partie $P \subseteq N$. On construit l'automate \mathcal{A} sur M égal à $(N, A, E, \{1_N\}, P)$ dont l'ensemble des transitions est $E = \{(n, a, n\mu(a)) \mid a \in A \text{ et } n \in N\}$. Il est facile de vérifier que pour tout m de M , il existe un chemin d'étiquette m de 1_N à $\mu(m)$. \square

Proposition 1.162. *Soit M un monoïde. Si K est une partie reconnaissable et L une partie rationnelle de M , alors $K \cap L$ est une partie rationnelle.*

Preuve. Soit μ un morphisme de M dans un monoïde fini N tel que $K = \mu^{-1}(P)$ pour une partie $P \subseteq N$. Soit $\mathcal{A} = (Q, M, E, I, F)$ un automate sur M dont le comportement est L . On construit un automate \mathcal{A}' dont le comportement est $K \cap L$. L'ensemble des états est $Q \times N$, les états initiaux sont ceux de $I \times \{1_N\}$ et les états finaux sont ceux de $F \times P$. Les transitions sont les triplets de la forme $((p, n), m, (q, n\mu(m)))$ où (p, m, q) est une transition de \mathcal{A} . \square