Table des matières

Jean-Eric Pin (LIAFA, CNRS et univ. Paris 7)
Automates réversibles : combinatoire, algèbre et topologie
Leçon donnée le jeudi 6 octobre 2005
Rédigée par Jérémie Chalopin.

Chapitre 1

Jean-Éric Pin. Automates Réversibles : Combinatoire, Algèbre et Topologie

Je vais commencer mon exposé par quelques rappels sur les automates, puis je vous en présenterai une version algébrique. Ensuite, je vous présenterai le sujet de l'exposé, les automates réversibles. Pour étudier ces automates, on ira d'abord se promener dans le groupe libre, puis on fera un peu de topologie pro-groupe avant de revenir à l'algèbre. Je vous présenterai alors la caractérisation algébrique des langages reconnus par cette classe d'automates. Pour terminer, je donnerai des algorithmes permettant de décider si un langage rationnel est reconnu par un automate réversible et je vous présenterai quelques problèmes ouverts dans ce contexte.

1.1 Rappels sur les automates

1.1.1 Mots, langages et automates

Un *alphabet* que je noterai par la suite A est un ensemble dont les éléments sont appelés des *lettres*. Un *mot* sur un alphabet A est une suite finie de lettres de A et on distingue un mot particulier : le mot vide, noté 1, qui ne contient aucune lettre. Par exemple, si l'alphabet A est $\{a, b, c\}$, $1, a, bab$ et $aacbab$ sont des mots sur A. On peut aussi *concaténer* les mots, c'est-à-dire les écrire bout à bout : par exemple, si on concatène *abra* et *cadabra*, on obtient un nouveau mot qui est *abracadabra*. L'ensemble de tous les mots sur un alphabet A est noté A^*, et lorsqu'on munit cet ensemble du produit de concaténation, on obtient un monoïde de neutre 1. Un *langage* est une partie de A^* : c'est un ensemble de mots sur A.

Un *automate* est un quintuplet $\mathcal{A} = (Q, A, E, I, F)$ où Q est un ensemble fini appelé l'ensemble des états, A est un alphabet, E est un sous-ensemble de $Q \times A \times Q$, appelé l'ensemble des *transitions*, I et F sont des parties de Q, appelées respectivement l'ensemble des états *initiaux* et l'ensemble des états *finaux*. Mais cette définition formelle n'est pas très agréable et on préfère généralement faire des dessins.

Sur la figure 1.1, j'ai dessiné un automate dont les états sont les sommets étiquetés $1, 2, 3, 4, 5$. On représente les états initiaux (resp. finaux) en leur ajoutant une flèche entrante (resp. sortante). Dans cet exemple, les états $1, 3$ et 4 sont initiaux et les états 2, 3 et 5 sont finaux. Les transitions sont représentées par des flèches étiquetées entre les états. S'il y a une flèche allant d'un état p à un état q étiquetée par une lettre a, cela signifie qu'on peut aller de l'état p à l'état q en lisant un a.

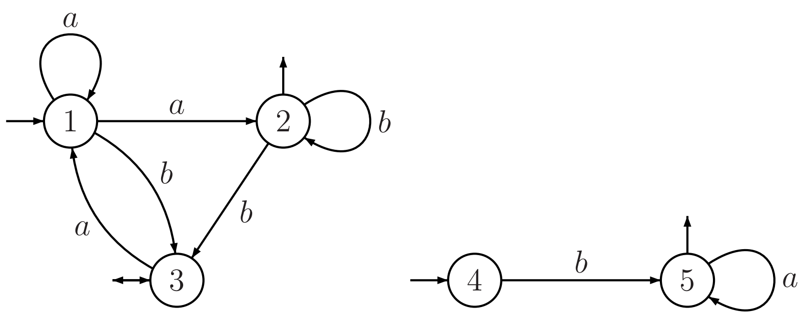


FIG. 1.1 – Un exemple d'automate.

Une lecture d'un mot u dans un automate $\mathcal{A}$ est un chemin dans l'automate $\mathcal{A}$ dont l'étiquette est le mot u. Une lecture est réussie si elle part d'un état initial et se termine dans un état final. Le langage *reconnu* par un automate $\mathcal{A}$ est l'ensemble des mots ayant au moins une lecture réussie dans $\mathcal{A}$. On dit qu'un langage est *reconnaissable* s'il existe un automate fini qui le reconnaît. Deux automates sont *équivalents* s'ils reconnaissent le même langage.

1.1.2 Automates déterministes

Je vais maintenant parler d'une classe particulière d'automates, les automates déterministes. Un automate déterministe est un quintuplet $\mathcal{A} = (Q, A, q_0, \cdot, F)$ où Q est un ensemble fini appelé l'ensemble des états, A est un alphabet, $q_0 \in Q$ est l'état initial, $F \subseteq Q$ est l'ensemble des états finaux. Enfin, la fonction[1] $(q, a) \to q{\cdot}a$ de $Q \times A$ dans Q est la fonction de transition de $\mathcal{A}$.

Il y a deux propriétés qui distinguent les automates déterministes des automates en général. D'une part, un automate déterministe n'a qu'un seul état initial. D'autre part, il n'y a jamais d'ambiguïté lors de la lecture d'un mot. C'est-à-dire que lorsqu'on est dans un état et qu'on doit lire une lettre, il y a au plus une façon de la lire. Autrement dit, chaque lettre définit une fonction (appelée aussi *action*) de l'ensemble des états dans lui-même.

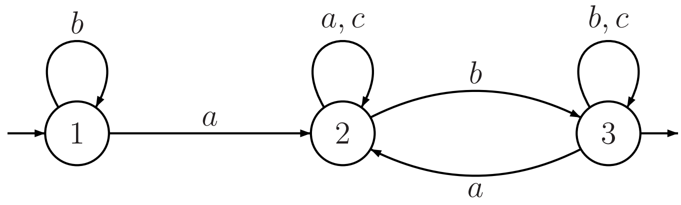


FIG. 1.2 – Un exemple d'automate déterministe.

Un exemple d'automate déterministe est présenté sur la figure 1.2. Le seul état initial est l'état 1. Dans cet automate, lorsqu'on lit *acbbca* depuis l'état initial 1, on arrive dans l'état 2 qui n'est pas final : le mot n'est pas reconnu par cet automate. Depuis l'état 1, on ne peut pas lire la lettre c : la fonction de transition n'est pas définie pour ce couple état-lettre. Ainsi, aucun mot commençant par un c n'est reconnu par cet automate.

[1]Je résiste ici à l'influence anglo-saxonne... Donc pour moi une *fonction* désigne ce que les anglophones appellent « partial function » et une *application* ce qu'ils appellent « function ».

Voici maintenant un résultat classique de la théorie des automates dont je ne ferai pas la démonstration.

Théorème 1.1.1. *Tout automate est équivalent à un automate déterministe.*

Sur la figure 1.3, l'automate $\mathcal{A}_1$ n'est pas déterministe mais l'automate $\mathcal{A}_2$ est un automate déterministe qui reconnaît le même langage que $\mathcal{A}_1$.

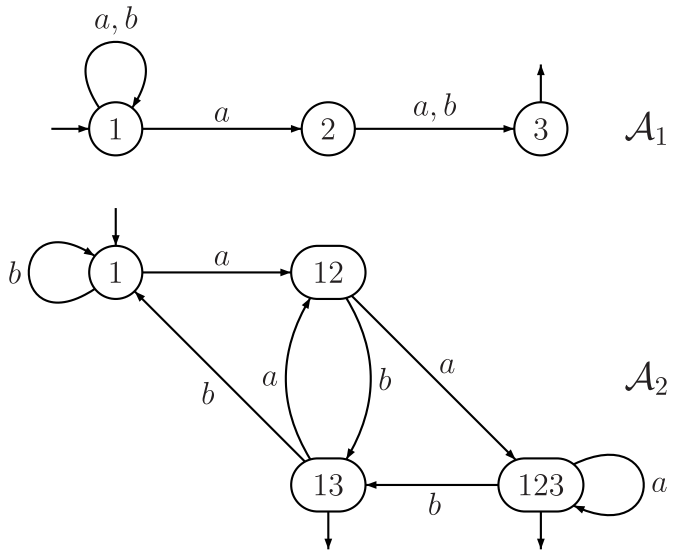


Fig. 1.3 – L'automate déterministe $\mathcal{A}_2$ est équivalent à l'automate non déterministe $\mathcal{A}_1$.

Dans un automate déterministe, on peut éliminer tous les états qui ne sont pas accessibles à partir de l'état initial, ou à partir desquels on ne peut pas atteindre un état final. On appelle cela *émonder* l'automate et cela ne modifie pas le langage reconnu par l'automate.

On définit aussi une relation d'équivalence sur l'ensemble des états : deux états p et q sont *équivalents* si, pour tout mot u, l'état $p{\cdot}u$ est final si et seulement si l'état $q{\cdot}u$ l'est aussi. On peut identifier des états équivalents sans changer le langage reconnu. L'automate obtenu après cette identification est appelé *l'automate minimal*. Fait remarquable, il ne dépend que du langage reconnu par l'automate. Autrement dit, bien qu'il existe une définition mathématique plus précise, on peut interpréter le mot « minimal » comme suit : l'automate minimal d'un langage reconnaissable L est l'automate déterministe reconnaissant L qui a le moins d'états possible. Pour la suite, il suffit de retenir qu'on dispose d'algorithmes simples pour le calculer à partir de n'importe quel automate reconnaissant le langage.

1.1.3 Langages rationnels

Si on voit les langages comme des ensembles de mots, on peut définir quelques opérations classiques. Si L_1 et L_2 sont des langages de A^*, on note $L_1 + L_2$ l'*union* de L_1 et de L_2 et on note L_1L_2 le *produit* (de concaténation) de L_1 et de L_2, c'est-à-dire l'ensemble des mots u_1u_2 où $u_1 \in L_1$ et $u_2 \in L_2$. Étant donné un langage L de A^*, on note L^* l'ensemble des mots u qui peuvent s'écrire $u = u_1u_2 \dots u_n$ avec $n \geqslant 0$ et $u_1, u_2, \dots, u_n \in L$. On peut aussi noter que L^* est le sous-monoïde de A^* engendré par L.

Dans l'exemple présenté sur la figure 1.4, on a décrit le langage reconnu par l'automate à l'aide d'une expression rationnelle : quand on lit un mot à partir de l'état 1 on peut lire autant de b que l'on veut, puis on peut lire a, aa ou ab pour arriver dans un état final. Le langage reconnu par cet automate est donc $b^*a + b^*aa + b^*ab$.

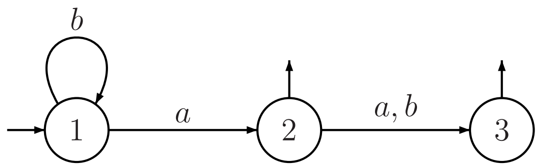


FIG. 1.4 – Le langage reconnu par cet automate est $L = b^*a + b^*aa + b^*ab$.

La classe des *langages rationnels* est la plus petite classe de langages contenant les langages finis, et fermée par union, produit et étoile. Le théorème suivant, dû à Kleene, est la base de la théorie des automates.

Théorème 1.1.2 ([Kle])**.** *Un langage est rationnel si et seulement si il est reconnaissable.*

Et voici maintenant un corollaire intéressant de ce théorème :

Corollaire 1.1.1. *Les langages rationnels sont fermés par intersection et par complémentation (dans A^*).*

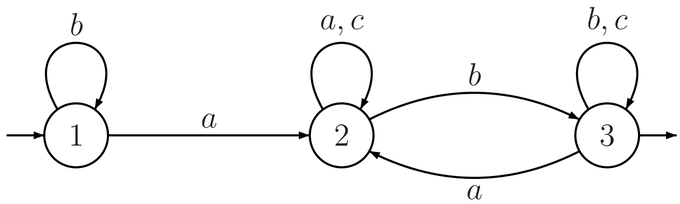
Si on essaie de montrer ce résultat à partir des expressions rationnelles qui décrivent ces langages, c'est très difficile. En revanche, si on travaille avec des automates, cela devient beaucoup plus simple. Partons d'un automate déterministe reconnaissant un langage L. S'il manque des transitions, c'est à dire, si au moins l'une des fonctions $q \to q{\cdot}a$ n'est pas partout définie, on ajoute un nouvel état 0 et on pose $q{\cdot}a = 0$ si $q{\cdot}a$ n'est pas déjà défini. En particulier, on pose $0{\cdot}a = 0$ pour chaque lettre a. Il suffit maintenant d'échanger les états finaux et non finaux pour obtenir un automate qui reconnaît le complémentaire de L dans A^*. Cet algorithme est illustré plus loin sur la figure 1.14.

1.2 L'approche algébrique

On va maintenant remplacer les automates par des monoïdes. Pour cela, étant donné un automate dont les transitions sont déterministes, on considère que chaque lettre est une fonction de l'ensemble des états dans lui-même.

1.2.1 Automates déterministes et monoïdes de transition

Si on considère l'automate de la figure 1.5, chacune des lettres a, b, c définit une fonction de l'ensemble des états dans lui-même.



1	1	2	3
a	2	2	2
b	1	3	3
c	-	2	3

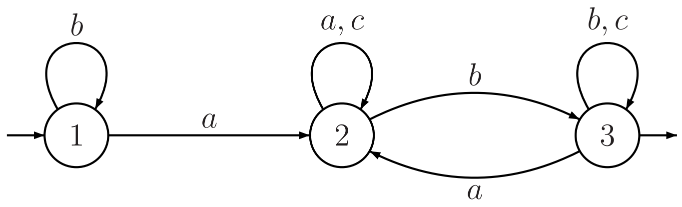
FIG. 1.5 – Un automate et les fonctions définies par les lettres sur les états de l'automate.

Par exemple, la lettre a envoie tous les états sur l'état 2 ; la lettre c définit une identité partielle puisqu'elle n'est pas définie pour l'état 1 et qu'elle envoie les états 2 et 3 sur eux-mêmes.

On veut désormais calculer le monoïde engendré par les fonctions définies par les lettres. Pour cela, on compose les fonctions définies par les éléments que l'on connaît (initialement ce sont seulement les actions des lettres de l'alphabet). Si on obtient une nouvelle fonction, on a obtenu un nouvel élément du monoïde. Si on obtient une fonction déjà connue, on a obtenu la valeur du produit des deux éléments. Par exemple, si on calcule l'action de aa sur les états de l'automate de la figure 1.5, on remarque qu'on obtient la même action que celle définie par a, on ajoute donc $aa = a$ à notre liste de relations. Si on calcule l'action de ab, on obtient la fonction[2] qui envoie tous les états sur l'état 3. On réitère le procédé jusqu'à connaître tous les éléments du monoïde engendré ainsi que la table de multiplication du monoïde.

Cela nous donne un algorithme pour convertir un automate déterministe en monoïde fini, et le monoïde obtenu est appelé le *monoïde de transition*. Si on part d'un automate fini, le monoïde obtenu est fini, puisque c'est un sous-monoïde du monoïde des fonctions de Q dans Q, qui est lui-même fini. Cela nous assure de la terminaison de l'algorithme.

Sur la figure 1.6, on a représenté le même automate que sur la figure 1.5 ainsi que tous les éléments du monoïde de transition et les relations obtenues lors de l'exécution de l'algorithme.



1	1	2	3
a	2	2	2
b	1	3	3
c	-	2	3
ab	3	3	3
bc	-	3	3
ca	-	2	2

$$\text{Relations :}\quad \begin{array}{lll} aa = a & bb = b & abc = ab \\ ac = a & cb = bc & bca = ca \\ ba = a & cc = c & cab = bc \end{array}$$

FIG. 1.6 – Un automate et une présentation de son monoïde de transition.

On a ainsi obtenu une représentation abstraite de l'automate et on va passer d'une représentation à l'autre de façon très souple. Il existe un morphisme naturel de A^* dans le monoïde de transition M d'un automate $\mathcal{A}$. Ce morphisme $\varphi : A^* \to M$ est défini par $\varphi(a) = a$ pour tout $a \in A$. Autrement dit $\varphi(u)$ est la fonction de Q dans lui-même définie par u. Pour calculer l'image d'un mot par le morphisme, j'utilise la présentation du monoïde de transition obtenue par l'algorithme précédent. Dans l'exemple de la figure 1.6, le mot cab s'envoie sur la fonction définie par bc.

Plus généralement deux mots u et v tels que $\varphi(u) = \varphi(v)$ ont la même action sur l'automate et ils seront donc simultanément acceptés ou rejetés. Mais il faut bien noter que la définition du monoïde de transition ne fait pas intervenir les états initiaux et finaux : la connaissance seule du monoïde ne nous permet pas de savoir si un mot est accepté ou pas.

[2] Signalons une petite subtilité que j'ai passée sous silence : comme on écrit les actions à droite, l'action définie par le mot ab est en fait la fonction notée habituellement $b \circ a$.

Néanmoins, pour tout élément m du monoïde, les mots de $\varphi^{-1}(m)$ sont ou bien tous acceptés par l'automate, ou bien tous rejetés. Par conséquent, il suffit de connaître les éléments du monoïde dont les antécédents par φ sont acceptés par l'automate. Dans l'exemple précédent, un mot u est accepté par l'automate si et seulement si $u \in \varphi^{-1}(\varphi(ab))$.

1.2.2 Reconnaissance par morphisme

Je vais maintenant vous donner une définition abstraite des langages reconnaissables où l'on n'a plus du tout d'automate, mais seulement des monoïdes.

Définition 1.2.1. *Soit M un monoïde et L un langage de A^*. On dit que M reconnaît L s'il existe un morphisme de monoïde $\varphi : A^* \to M$ et une partie P de M telle que $L = \varphi^{-1}(P)$.*

Et la proposition suivante nous assure que la reconnaissance par monoïde fini est équivalente à la reconnaissance par automate fini.

Proposition 1.2.1. *Un langage est reconnu par un monoïde fini si et seulement si il est reconnu par un automate déterministe fini.*

Pour passer de la reconnaissance par automates à la reconnaissance par monoïde, il suffit de considérer le monoïde de transition d'un automate déterministe reconnaissant le langage, et c'est ce que je vous ai expliqué précédemment.

Maintenant, on va partir d'un morphisme de monoïde et je vais vous montrer comment construire un automate. Pour cela, on va construire le graphe de Cayley du monoïde. On a un monoïde M et un morphisme de monoïdes $\varphi : A^* \to M$ qui définit une action de A sur M, qui est la multiplication à droite par $\varphi(a)$: $m{\cdot}a = m\varphi(a)$.

Le *graphe de Cayley* (M, A) est un graphe dirigé dont les sommets sont les éléments du monoïde. Pour chaque lettre $a \in A$ et chaque élément $m \in M$, il y a une flèche étiquetée a entre m et $m{\cdot}a$. Si on regarde ce graphe dirigé comme un automate, où on choisit 1, l'élément neutre de M, comme état initial et $\varphi(L)$ comme ensemble d'états finaux, on a obtenu un automate qui reconnaît le langage L.

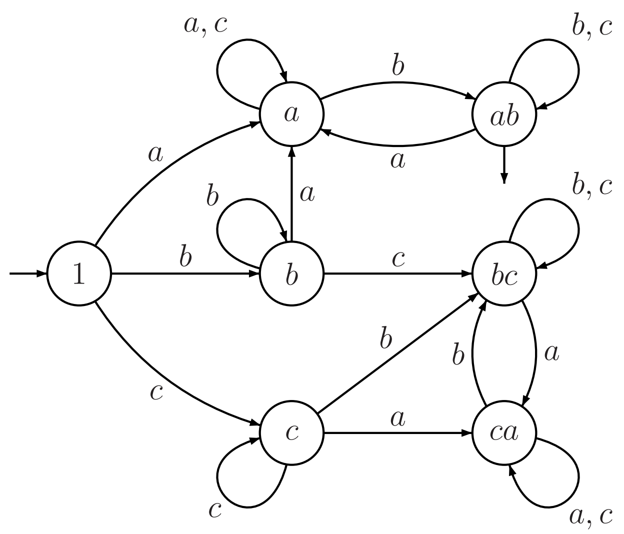


FIG. 1.7 – Le graphe de Cayley vu comme un automate.

Sur la figure 1.7, on reconnaît le graphe de Cayley du monoïde obtenu précédemment. À chaque élément du monoïde $\{1, a, b, c, ab, bc, ca\}$ est associé un sommet et les flèches nous indiquent la multiplication (à droite) des éléments de M par les éléments de $\varphi(A)$. Si on choisit 1 comme état initial et ab comme état final, l'ensemble des mots reconnus est exactement le langage reconnu par l'automate de la figure 1.6.

1.2.3 Monoïde syntactique

Je vous ai parlé précédemment de l'automate minimal associé à un langage. Il y a un monoïde particulier associé à cet automate minimal.

Définition 1.2.2 (algorithmique)**.** *Le monoïde syntactique d'un langage est le monoïde de transition de son automate minimal.*

Ce monoïde est canonique, au sens où j'ai un algorithme qui me permet de le construire : on calcule d'abord l'automate minimal du langage, puis on calcule le monoïde de transition de cet automate. C'est cet objet qui va nous servir dans la deuxième partie de cet exposé. Je vais vous donner maintenant une autre définition de cet objet, qui est purement algébrique.

On dit que deux mots u et v sont équivalents s'ils ont le même contexte dans L, c'est-à-dire que si on ajoute autour de u et de v un préfixe x et un suffixe y, alors xuy est dans L si et seulement si xvy est dans L. Cette relation d'équivalence est appelée la *congruence syntactique* de L et elle permet de donner une autre définition du monoïde syntactique.

Définition 1.2.3 (algébrique)**.** *Le monoïde syntactique d'un langage $L \subset A^*$ est le monoïde quotient de A^* par la congruence syntactique de L : $u \sim_L v$ si et seulement si, pour tout $x, y \in A^*$, $xvy \in L \iff xuy \in L$.*

On va maintenant rajouter une relation d'ordre sur les états de l'automate minimal $\mathcal{A} = (Q, A, \cdot, q_0, F)$ d'un langage L. Étant donnés deux états $p, q \in Q$, on dit que $p \leqslant q$ si pour tout $u \in A^*, q{\cdot}u \in F \implies p{\cdot}u \in F$. C'est-à-dire qu'à chaque fois que je peux aller de q dans un état final en lisant un mot u, en lisant u à partir de p, j'arrive aussi dans un état final.

La relation $\leqslant$ est une relation d'ordre partiel sur les états de l'automate minimal. On considère seulement des automates minimaux car sinon, la relation $\leqslant$ ne vérifie pas la propriété d'antisymétrie (on a des états équivalents qui ne sont pas égaux).

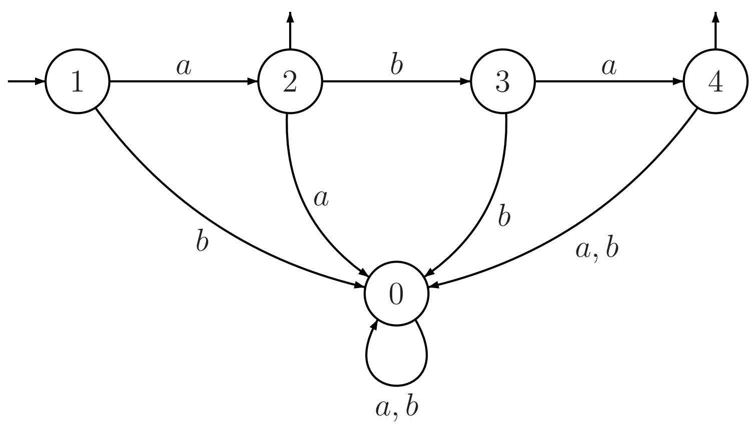


FIG. 1.8 – Exemple d'automate minimal ordonné. Ici, $2 \leqslant 4$, $1 \leqslant 3$ et $1, 2, 3, 4 \leqslant 0$.

Dans l'automate de la figure 1.8, $2 \leqslant 4$, puisque le seul mot accepté à partir de l'état 4 est le mot vide qui est aussi accepté à partir de l'état 2.

Maintenant qu'on a un ordre sur l'automate minimal, on va pouvoir obtenir un ordre sur le monoïde de transition de cet automate, c'est-à-dire sur le monoïde syntactique. À partir de l'ordre sur les états, on définit naturellement un ordre sur les fonctions :

Définition 1.2.4 (algorithmique)**.** *Le monoïde syntactique ordonné d'un langage L est le monoïde de transition de l'automate minimal de L, ordonné par $u \leqslant v$ si et seulement si pour tout $q \in Q$, $q{\cdot}u \leqslant q{\cdot}v$.*

On peut aussi obtenir cet ordre sur la représentation abstraite du monoïde syntactique.

Définition 1.2.5 (algébrique)**.** *Le monoïde syntactique ordonné est le monoïde syntactique, muni de l'ordre induit par le préordre syntactique $\leqslant_L$ défini sur A^* par : $u \leqslant_L v$ si et seulement si, pour tout $x, y \in A^*$, $xvy \in L \implies xuy \in L$.*

J'ai décrit la relation $\leqslant_L$ sur les mots, et c'est donc seulement un préordre, puisqu'il est clair que si $u \leqslant_L v$ et $v \leqslant_L u$, on a $u \sim_L v$. Mais lorsqu'on redescend dans le monoïde syntactique, on obtient une relation d'ordre sur les classes d'équivalence. Cette relation d'ordre est sympathique parce qu'elle est stable par multiplication.

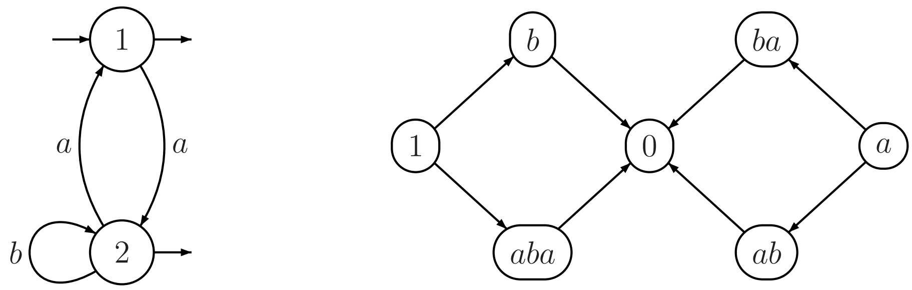


FIG. 1.9 – L'automate minimal et l'ordre syntactique de $L = (ab^*a)^*(1 + a)$.

Sur la figure 1.9, j'ai représenté à gauche l'automate minimal reconnaissant $L = (ab^*a)^*(1+a)$ et à droite, j'ai donné une représentation de son monoïde syntactique ordonné. Sur cet exemple, $1 \leqslant b$. En effet, pour tous mots u, v, si $ubv \in L$, alors il faut qu'en lisant u depuis l'état initial, on arrive dans l'état 2 pour pouvoir lire le b et qu'en lisant v depuis l'état 2, on arrive dans un état final. Par conséquent, si $ubv \in L$, en lisant uv depuis l'état initial, on arrive dans un état final et donc $uv \in L$. En revanche, $1 \nleqslant ab$ puisque $1(ab)ba \in L$ mais $1(1)ba \notin L$.

1.3 Automates réversibles

1.3.1 Définition et exemples

Je vais maintenant aborder le sujet de l'exposé proprement dit : les automates réversibles. Dans un automate déterministe, on se souvient que la configuration de gauche de la figure 1.10 est interdite, c'est-à-dire qu'à partir d'un état q, je ne peux pas lire une lettre a de deux manières différentes. Dans un automate réversible, on interdit aussi la configuration duale, qui est représentée à droite sur la figure 1.10, c'est-à-dire qu'on ne peut pas trouver deux états différents qui permettent d'aller dans un même état en lisant la même lettre. Les transitions d'un automate réversible sont donc déterministes et codéterministes.

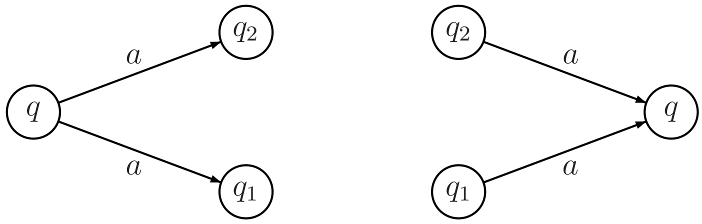


Fig. 1.10 – Configurations interdites dans un automate réversible.

Autrement dit, un automate *réversible* est un automate dans lequel chaque lettre induit une fonction injective de l'ensemble des états dans lui-même. Dans le cas où les fonctions sont des applications, ce sont des permutations et l'automate est dans ce cas appelé *automate de permutations* ou *automate à groupe.*

Il reste à définir quels sont les états initiaux et finaux. Si on veut conserver un automate déterministe (resp. codéterministe), il faut un seul état initial (resp. final). Mais ici, on va perdre le déterminisme et le codéterminisme, puisque dans un automate réversible, on autorise plusieurs états initiaux et plusieurs états finaux.

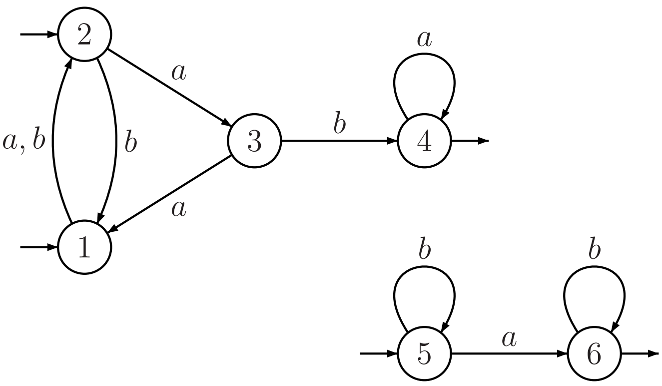


Fig. 1.11 – Un exemple d'automate réversible.

L'automate de la figure 1.11 est un premier exemple d'automate réversible. On peut vérifier que les lettres a et b définissent des fonctions injectives de l'ensemble des états dans lui même.

Maintenant qu'on a défini les automates réversibles, on peut définir les langages réversibles :

Définition 1.3.1. *Un langage est réversible s'il est accepté par un automate réversible (muni de plusieurs états initiaux et de plusieurs états finaux).*

Et le but de mon exposé est de répondre à la question suivante :

Problème 1.3.1. *Peut-on décider si un langage rationnel donné est réversible ?*

Autrement dit, étant donné un automate ou une expression rationnelle, est-ce qu'on peut décider si le langage reconnu par cet automate, ou représenté par cette expression rationnelle, est reconnaissable par un automate réversible.

On pourrait penser que si un langage est reconnu par un automate réversible, alors l'automate minimal de ce langage est réversible, et il suffirait de calculer cet automate minimal, mais cela ne marche pas. En effet, si on considère le langage $\{a, ac, bc\}$, l'automate de gauche de la figure 1.12 est un automate réversible qui reconnaît ce langage, mais l'automate minimal de ce langage, représenté à droite sur la figure 1.12 n'est pas réversible.

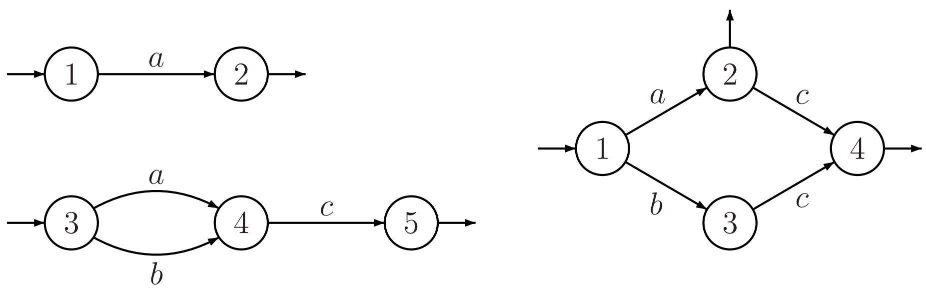


FIG. 1.12 – Automates réversible et minimal de $\{a, ac, bc\}$.

Je vais commencer par vous donner quelques exemples de langages réversibles.

– Les langages finis sont réversibles : on peut faire un automate par mot et ensuite on fait l'union disjointe de tous ces automates.
– Les langages à groupe sont réversibles : ces langages sont reconnus par des groupes finis, ou si on préfère par des automates de permutations (qui sont des cas particuliers d'automates réversibles).
– Le langage miroir d'un langage réversible est réversible. Étant donné un langage L, le miroir de L est le langage L' est constitué de tous les mots de L lus de droite à gauche. Si L est reconnu par un automate réversible, l'automate obtenu en retournant les transitions et en intervertissant les états initiaux et finaux est un automate réversible qui reconnaît L'.
– Toute *combinaison booléenne positive* (i.e. union finie d'intersections finies) de langages réversibles est réversible. Il est facile de voir qu'une union finie de langages réversibles est réversible puisqu'il suffit de faire l'union disjointe des automates reconnaissant chacun des langages. Pour l'intersection finie, c'est un peu plus difficile à voir, mais cela est vrai aussi.

1.3.2 Une première description des langages réversibles

Je vais maintenant vous donner une première description des langages réversibles. Si L est un langage réversible, la proposition suivante donne une propriété du complémentaire de L, noté L^c.

Proposition 1.3.1. *Soit L un langage réversible de A^*. Alors*

(1) *L^c est une combinaison booléenne positive de langages de la forme R ou A^*aR où R est un langage à groupe,*

(2) *L^c est une combinaison booléenne positive de langages de la forme R ou RaA^* où R est un langage à groupe.*

Si on arrive à prouver (2), (1) sera un corollaire puisque les langages réversibles sont clos par passage au miroir. Puisque les langages réversibles sont clos par intersection et union finies, on peut se ramener au cas où l'automate a un seul état initial et un seul état final. Je vais vous présenter la preuve sur un exemple, en utilisant l'automate $\mathcal{A}$ de la figure 1.13.

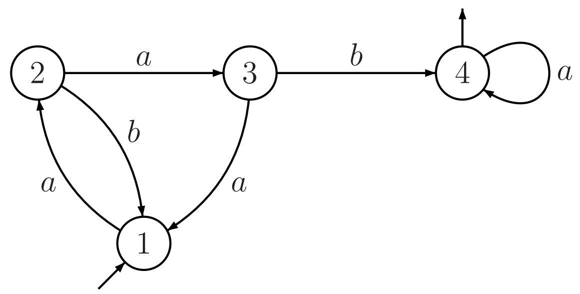


Fig. 1.13 – Automate réversible $\mathcal{A}$ servant d'exemple pour la preuve de la Proposition 1.3.1.

On commence par construire l'automate reconnaissant le complémentaire du langage reconnu par $\mathcal{A}$, qui est représenté sur la figure 1.14. Pour obtenir cet automate, on a d'abord ajouté un état (l'état 0), et on a complété $\mathcal{A}$ en ajoutant des transitions étiquetées b entre les états 1 et 0 et entre 4 et 0 (ce sont les transitions dessinées avec des tirets). Ensuite, on a échangé les états finaux et non finaux. L'automate ainsi obtenu n'est plus un automate réversible, puisque l'état 0 peut être atteint depuis trois états différents en lisant la lettre b.

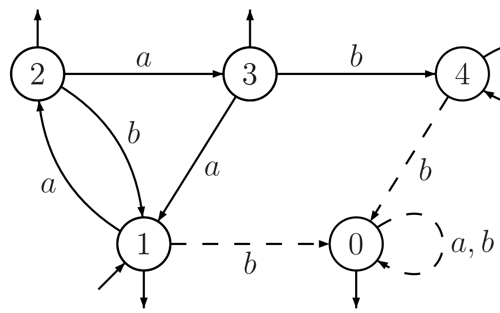


Fig. 1.14 – L'automate reconnaissant le complémentaire du langage reconnu par $\mathcal{A}$.

Maintenant, je vais vous présenter une construction qui va revenir plusieurs fois au cours de l'exposé. Dans l'automate réversible, je vais rajouter des transitions en pointillé de manière à transformer les fonctions injectives définies par les lettres en des bijections. Sur la figure 1.15, j'ai ajouté des transitions (qui sont en pointillé) à l'automate $\mathcal{A}$ de telle sorte que a et b définissent des bijections de l'ensemble des états de $\mathcal{A}$ dans lui même. Il n'y a pas de règle pour compléter les injections en bijections : je peux rajouter les flèches en pointillé comme je veux, tant que j'obtiens une bijection. L'automate $\mathcal{A}'$ ainsi obtenu est un automate à groupe.

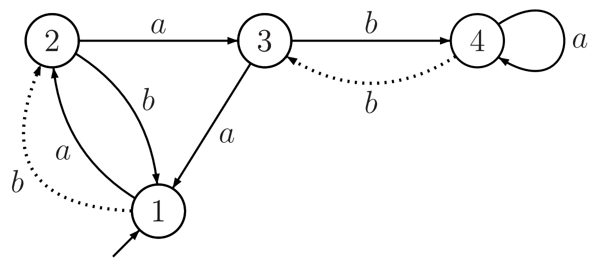


Fig. 1.15 – L'automate $\mathcal{A}'$ avec les flèches en pointillé obtenu à partir de $\mathcal{A}$.

Notons $R(q)$ le langage reconnu par l'automate à groupe obtenu (avec les flèches en pointillé) en prenant q comme état final. J'affirme que $L^c = R(1) \cup R(2) \cup R(3) \cup R(1)bA^* \cup R(4)bA^*$.

En effet, regardons par exemple les mots de $R(2)$. Si un mot u appartient à $R(2)$, cela signifie qu'en lisant u dans $\mathcal{A}'$ depuis l'état initial, on arrive dans l'état 2. On considère maintenant deux cas. Ou bien lors de la lecture de u dans $\mathcal{A}'$, on n'a jamais utilisé de flèche en pointillé, et alors dans l'automate du complémentaire de $\mathcal{A}$, si on lit u depuis l'état initial, on arrive dans l'état 2 qui est un état final : $u \in L^c$. Ou alors lorsqu'on a lu u dans $\mathcal{A}'$, on a utilisé une flèche en pointillé, et cela signifie que si on lit u dans l'automate du complémentaire, on va prendre une transition qui arrive dans l'état 0, dans lequel on va rester pour lire la fin de u : puisque 0 est un état final dans l'automate du complémentaire, $u \in L^c$.

On peut faire la même chose pour $R(1)$ et $R(3)$. Considérons maintenant un mot u de $R(1)bA^*$ que l'on peut écrire $u = u_1bu_2$ avec $u_1 \in R(1)$. Si on emprunte une flèche en pointillé en lisant dans $\mathcal{A}'$ le préfixe u_1, alors, pour les mêmes raisons qu'auparavant, on aboutit dans l'état 0 lorsqu'on lit u dans l'automate du complémentaire, et on a donc $u \in L^c$. Si on n'utilise jamais de flèche en pointillé en lisant u_1 dans $\mathcal{A}'$, alors si on lit u_1 dans l'automate du complémentaire, on termine dans l'état 1 et lorsqu'on lit le b qui suit u_1, on arrive dans l'état 0 dans lequel on reste pour lire la fin du mot u : là encore, on a $u \in L^c$. On peut faire le même raisonnement pour les mots de $R(4)bA^*$.

Je vous ai montré l'inclusion dans un sens, mais cela marche dans les deux sens et on a bien $L^c = R(1) \cup R(2) \cup R(3) \cup R(1)bA^* \cup R(4)bA^*$.

1.3.3 Une première condition nécessaire

Je vais maintenant vous donner une première condition nécessaire pour qu'un langage soit réversible. C'est une condition très simple et vraiment algébrique, qui sera un élément clé de la caractérisation qu'on va obtenir. Rappelons qu'un élément e d'un monoïde est un *idempotent* si $ee = e$.

Proposition 1.3.2. *Les idempotents du monoïde syntactique d'un langage réversible commutent.*

Pour montrer cette proposition, on regarde les idempotents du monoïde de transition d'un automate réversible acceptant un langage L. Ces idempotents doivent aussi être des fonctions injectives puisqu'on est dans un automate réversible. Or une fonction injective idempotente est forcément une identité partielle : par conséquent, ces idempotents commutent. Et puisqu'on peut montrer que si les idempotents du monoïde de transition d'un automate reconnaissant un langage L commutent, alors les idempotents du monoïde syntactique de L commutent, on obtient cette première condition nécessaire.

1.4 Le groupe libre

1.4.1 Définition

Je vais maintenant vous parler du *groupe libre*, qui a une définition mathématique très classique.

Étant donné un alphabet A, je prends une copie disjointe de A, que je note $\bar{A}$, et je note $\tilde{A}$ l'union disjointe $A \cup \bar{A}$. Chaque lettre a de A a donc une copie $\bar{a}$ dans $\bar{A}$. Le groupe libre sur A, noté $FG(A)$, est le quotient de $\tilde{A}^*$ par les relations $a\bar{a} = 1 = \bar{a}a$ pour tout $a \in A$. Cela correspond à dire que $\bar{a}$ est l'inverse formel de a dans $FG(A)$ et justifie la convention $\bar{\bar{a}} = a$. On note aussi $\bar{u}$ l'inverse d'un élément u : par exemple, l'inverse de $u = a\bar{b}\bar{a}ba$ est $\bar{u} = \bar{a}\bar{b}ab\bar{a}$.

On note que la réduction qui consiste à éliminer les $a\bar{a}$ et les $\bar{a}a$ dans un mot de $\tilde{A}^*$ est *confluente* : quelle que soit la manière dont on élimine les $a\bar{a}$ et les $\bar{a}a$, on obtient le même résultat. Cela nous permet d'avoir un représentant canonique dans $\tilde{A}^*$ pour chaque classe de $FG(A)$.

Dans le groupe libre, comme pour n'importe quel monoïde, il existe également une définition des parties rationnelles.

Définition 1.4.1. *La classe des parties rationnelles du groupe libre est la plus petite classe $\mathcal{R}$ de parties du groupe libre telle que :*

(1) *chaque sous-ensemble fini du groupe libre appartient à $\mathcal{R}$,*

(2) *si S et T sont dans $\mathcal{R}$, alors ST et $S \cup T$ y sont aussi,*

(3) *si S est dans $\mathcal{R}$, alors S^*, le sous-monoïde engendré par S, y est aussi.*

Comme pour les monoïdes, le produit ST de deux ensembles S et T est l'ensemble des $\{st \mid s \in S, t \in T\}$. On peut aussi remarquer que si un ensemble S est rationnel, le sous-groupe engendré par S, noté $\langle S\rangle$, est aussi rationnel, puisque $\langle S\rangle = (S \cup \bar{S})^*$ où $\bar{S}$ est l'ensemble des inverses des éléments de S (qui est lui aussi rationnel).

1.4.2 Automates réversibles dans le groupe libre

La définition des automates dans le groupe libre est assez naturelle. On garde la même définition pour les automates, mais on peut lire les transitions dans les deux sens. C'est-à-dire que s'il y a une transition d'un état p à un état q étiquetée par la lettre a, on peut aller de p à q en lisant a et aller de q à p en lisant $\bar{a}$. Autrement dit, pour chaque transition de p à q étiquetée a, on a implicitement une transition de q à p étiquetée $\bar{a}$.

L'automate de la figure 1.16 qui reconnaît le langage $\{a, ac, bc\}$ dans le monoïde libre reconnaît $\{a\} \cup a\langle\bar{b}a\rangle c$ dans le groupe libre.

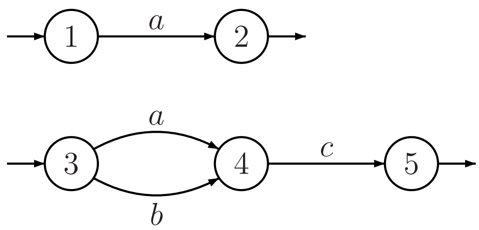


Fig. 1.16 – Dans le groupe libre, cet automate reconnaît le langage $\{a\} \cup a\langle\bar{b}a\rangle c$.

Dans un automate réversible, les lettres définissent des fonctions injectives de l'ensemble des états dans lui même. Et par conséquent, les fonctions réciproques de ces fonctions existent aussi. Il est donc assez naturel de regarder les automates réversibles dans le groupe libre. On va maintenant décrire quelles sont les parties du groupe libre qui sont reconnues par des automates réversibles.

1.4.3 Sous-groupes rationnels du groupe libre

Le résultat suivant est classique, même si sa présentation l'est moins ; c'est une caractérisation des sous-groupes du groupe libre qui sont reconnus par un automate réversible. Rappelons qu'un sous-groupe est finiment engendré s'il admet un ensemble fini de générateurs.

Théorème 1.4.1. *Soit H une partie du groupe libre. Sont équivalents :*

(1) *H est un sous-groupe rationnel,*

(2) *H est un sous-groupe finiment engendré,*

(3) *H est reconnu par un automate réversible dont l'unique état initial est aussi l'unique état final.*

Dans la troisième assertion du théorème, puisque l'élément neutre du groupe H est le mot vide, on demande que l'état initial et final de l'automate coïncident.

Je vais commencer par vous prouver qu'un automate réversible avec un unique état initial et final reconnaît un sous-groupe finiment engendré du groupe libre. Je vais vous faire la preuve sur l'exemple de la figure 1.17.

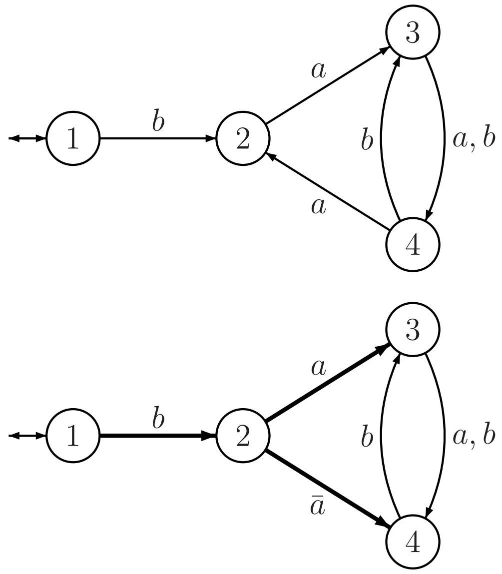


FIG. 1.17 – Un automate et un arbre couvrant de cet automate.

On commence par prendre un arbre couvrant dirigé de l'automate dont la racine est l'état initial. Les flèches de l'arbre couvrant sont dirigés de la racine vers les feuilles. Sur la figure 1.17, j'ai représenté un arbre couvrant de l'automate avec les flèches en gras. Il faut noter que pour obtenir cet arbre couvrant, j'ai renversé une flèche étiquetée a et je l'ai étiquetée $\bar{a}$.

Maintenant, pour vous montrer que l'automate reconnaît un sous-groupe finiment engendré, je vais vous expliquer comment on trouve les générateurs. En fait, cette construction permet de démontrer la formule de Schreier, qui est un résultat classique de la combinatoire du groupe libre.

Pour trouver les générateurs, on va regarder les flèches qui ne sont pas dans l'arbre. À chacune de ces transitions, on associe un mot obtenu de la manière suivante. À une transition de p à q étiquetée par une lettre a, on associe le mot $ua\bar{v}$, où u est l'étiquette du seul chemin dans l'arbre qui va de l'état initial à p et v est l'étiquette du seul chemin dans l'arbre qui va de l'état initial à q.

Sur l'exemple de la figure 1.17, on associe à la transition étiquetée a de 3 à 4, le mot $(ba)a(\overline{b\bar{a}}) = baaa\bar{b}$. En effet, le seul chemin qui va de 1 à 3 dans l'arbre est étiqueté par ba et le seul chemin de 1 à 4 dans l'arbre porte l'étiquette $b\bar{a}$. De même, à la transition de 3 à 4 étiquetée b, on associe le mot $baba\bar{b}$ et à la transition de 4 à 3 étiquetée b, on associe le mot $b\bar{a}b\bar{a}\bar{b}$.

Ensuite, c'est un petit exercice de vérifier que le langage reconnu par l'automate est exactement le langage engendré par les mots que l'on vient d'obtenir. Étant donné un mot u du langage reconnu par l'automate, on regarde le chemin qui va de l'état initial à l'état final dans l'automate qui reconnaît ce mot. À chaque fois qu'on utilise une transition qui n'est pas dans l'arbre, on note le générateur associé pour obtenir un mot qui se réduit à u. Par exemple, si on considère le mot $b\bar{a}baaabab\bar{b}$, avec cette méthode on obtient le mot $b\bar{a}b\bar{a}\bar{b}{\cdot}baaa\bar{b}{\cdot}baba\bar{b}$ qui se réduit bien à $b\bar{a}baaabab\bar{b}$. Le sous-groupe du groupe libre reconnu par l'automate de la figure 1.17 est donc $\langle baaa\bar{b}, baba\bar{b}, b\bar{a}b\bar{a}\bar{b}\rangle$.

Je vais maintenant vous montrer comment obtenir un automate réversible à partir d'un groupe finiment engendré. Je vais vous présenter cette construction pour le groupe engendré par $a\bar{b}a$ et $abba$. On commence par représenter ces générateurs par un bouquet de cercles. C'est-à-dire qu'on construit un automate avec un seul état initial et final et que pour chaque générateur u, on dessine un cycle étiqueté par u qui boucle autour de l'état initial et final. L'automate de gauche de la figure 1.18 correspond à cette construction pour le langage $\langle a\bar{b}a, abba\rangle$. On note que pour éviter les lettres de type $\bar{a}$, il suffit de retourner les flèches lors de la construction ; c'est ce que j'ai fait sur la figure 1.18 pour le générateur $a\bar{b}a$.

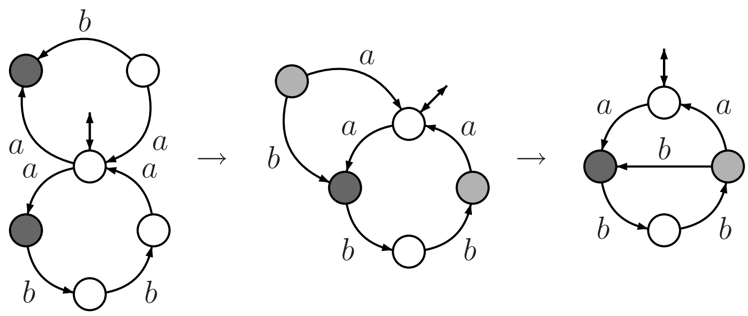


Fig. 1.18 – Construction d'un automate réversible acceptant $\langle a\bar{b}a, abba\rangle$.

Cet automate n'est pas réversible, mais je vais vous expliquer comment arriver à un automate réversible à partir de celui-ci. À chaque fois qu'on rencontre l'une des configurations décrite sur la figure 1.10, on identifie les états q_1 et q_2. Si on observe l'automate de gauche de la figure 1.18, on voit qu'en lisant un a depuis l'état initial, on peut atteindre deux états différents : on identifie les deux états pour obtenir un automate équivalent qui est représenté au centre de la figure 1.18. Ce nouvel automate n'est toujours pas réversible puisqu'on peut arriver dans l'état initial en lisant un a depuis deux états différents : on identifie ces deux états pour obtenir l'automate de droite de la figure 1.18, qui est un automate réversible avec un unique état initial et final.

1.4.4 Parties réversibles du groupe libre

On va maintenant considérer les parties réversibles du groupe libre qui sont définies de manière très naturelle.

Définition 1.4.2. *Une partie du groupe libre est réversible si elle est acceptée par un automate réversible.*

Le théorème suivant nous donne une caractérisation des parties réversibles du groupe libre.

Théorème 1.4.2. *Une partie du groupe libre est réversible si et seulement si elle est union finie de classes latérales gauches de sous-groupes finiment engendrés du groupe libre.*

Pour montrer cela, on se ramène au cas où l'automate a un seul état initial i et un seul état final f. Soit H le sous-groupe finiment engendré reconnu en prenant f comme état initial et final. Si u est un mot reconnu par l'automate, c'est-à-dire, si $i{\cdot}u = f$, la partie reconnue par l'automate est uH. En effet, si w est un mot reconnu par l'automate avec i comme état initial et f comme état final, $i{\cdot}w = f$, et $i{\cdot}u\bar{u}w = f$, et alors $\bar{u}w \in H$. Par exemple, la partie reconnue par l'automate de la figure 1.19 est $a\bar{b}\langle baaa\bar{b}, baba\bar{b}, b\bar{a}b\bar{a}\bar{b}\rangle$.

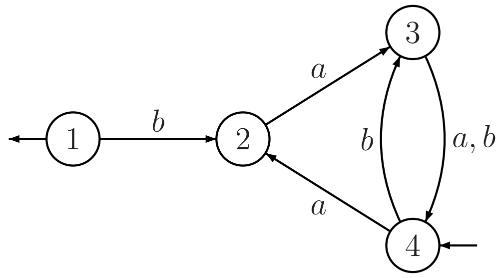


FIG. 1.19 – Dans le groupe libre, cet automate reconnaît $a\bar{b}\langle baaa\bar{b}, baba\bar{b}, b\bar{a}b\bar{a}\bar{b}\rangle$.

Je ne vous fais pas la démonstration mais la réciproque est également vraie.

On a obtenu une description assez agréable des parties réversibles, mais cela ne nous permet pas de décider facilement si un langage est réversible. D'une part, si on a une partie rationnelle, il n'est pas évident de voir si cette partie est une union finie de classes latérales gauches de sous-groupes finiment engendrés. D'autre part, on n'a pour le moment qu'une description des parties réversibles du groupe libre et pas du monoïde libre.

Je vais vous donner une caractérisation un peu plus algorithmique des parties réversibles du groupe libre. Le théorème suivant est l'analogue du théorème de Kleene pour les parties réversibles du groupe libre. Ce résultat permet d'obtenir un procédé algorithmique pour engendrer ces parties.

Théorème 1.4.3. *Les parties réversibles du groupe libre forment la plus petite classe $\mathcal{F}$ de parties telles que*

(1) $\varnothing \in \mathcal{F}$ *et, pour tout* $g \in FG(A)$, $\{g\} \in \mathcal{F}$,
(2) *si* $S_1, S_2 \in \mathcal{F}$, *alors* $S_1 \cup S_2 \in \mathcal{F}$,
(3) *si* $S \in \mathcal{F}$ *et* $g \in FG(A)$, *alors* $gS \in \mathcal{F}$,
(4) *si* $S \in \mathcal{F}$, *alors* $\langle S\rangle \in \mathcal{F}$.

1.4.5 Retour au monoïde libre

Le monoïde libre A^* peut être considéré comme un sous-monoïde du groupe libre $FG(A)$. Un langage de A^* est donc réversible si et seulement si c'est la trace sur le monoïde libre d'un langage réversible du groupe libre.

Théorème 1.4.4. *Un langage L de A^* est réversible si et seulement si $L = K \cap A^*$, où K est une union finie de classes latérales de sous-groupes finiment engendrés du groupe libre $FG(A)$.*

Cette caractérisation n'est toujours pas très utilisable d'un point de vue algorithmique, puisque décider si un langage est de cette forme ne semble pas facile.

1.5 Topologie pro-groupe

Maintenant, je vais vous parler un petit peu de topologie profinie, en l'occurrence pro-groupe. Ce terme a été introduit par M. Hall Jr. en 1949, et depuis cela a été beaucoup étudié. Il faut noter que la topologie p-adique est un cas particulier de topologie profinie.

La topologie qui nous intéresse ici est basée sur le théorème suivant. En termes savants, ce théorème dit que le monoïde libre est résiduellement fini pour les groupes finis, mais vous constaterez que la preuve qui suit est beaucoup plus élémentaire que celle que l'on donne habituellement.

Théorème 1.5.1. *Deux mots distincts u et v de A^* peuvent toujours être séparés : il existe un groupe fini G et un morphisme de monoïde $\varphi : A^* \to G$ tel que $\varphi(u) \neq \varphi(v)$.*

Un premier exemple est le groupe $\mathbb{Z}/2\mathbb{Z}$ qui permet de séparer les mots de longueur paire des mots de longueur impaire : il suffit de considérer le morphisme $\varphi : A^* \to G$ tel que $\varphi(u) = |u| \bmod 2$, où $|u|$ est la longueur du mot u. En utilisant le même groupe, on peut séparer les mots qui contiennent un nombre pair de a des mots qui en contiennent un nombre impair : il suffit de considérer le morphisme $\varphi : A^* \to G$ défini par $\varphi(a) = 1$ et $\varphi(b) = 0$, pour tout $b \in A \setminus \{a\}$.

Je vais maintenant vous montrer comment séparer deux mots distincts quelconques. Je vais faire la preuve sur les mots *abab* et *abba*. Ces mots ont la même longueur, le même nombre de a et de b : on ne peut pas utiliser les techniques de l'exemple précédent.

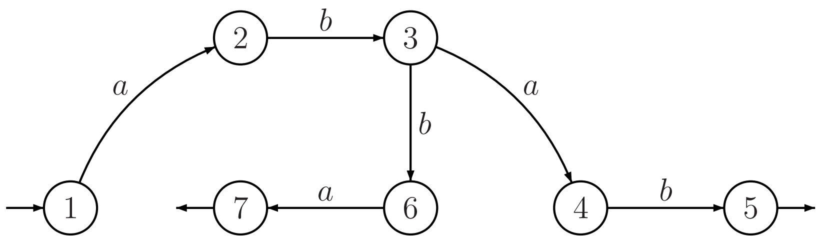


FIG. 1.20 – Un automate qui reconnaît *abab* si l'état 5 est final et *abba* si l'état 7 est final.

On commence par construire un automate déterministe à partir des deux mots *abab* et *abba* comme sur la figure 1.20. Cet automate est réversible ; il reconnaît *abab* si l'état 5 est final et *abba* si l'état 7 est final.

Ensuite, encore une fois, on ajoute des flèches en pointillé pour obtenir un automate de permutations comme sur la figure 1.21. Je vais vous montrer que le groupe de permutations obtenu à partir de cet automate sépare les deux mots. En effet, la permutation définie par le mot *abab* envoie 1 sur 5, alors que la permutation définie par *abba* envoie 1 sur 7.

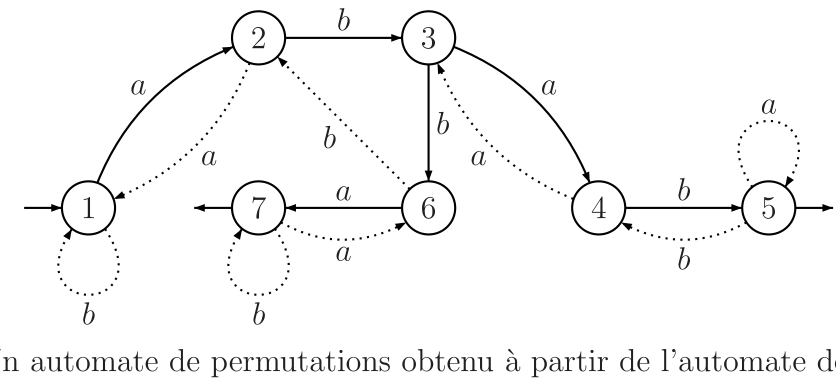


FIG. 1.21 – Un automate de permutations obtenu à partir de l'automate de la figure 1.20.

On peut maintenant définir une distance entre deux mots en fonction de la taille du plus petit groupe qui les sépare. Et on va dire que plus la taille du groupe qui sépare les deux mots est grande, plus les mots sont proches. Il est très difficile d'avoir de l'intuition sur cette définition lorsqu'on la rencontre pour la première fois. Mais si vous avez l'habitude de travailler avec la topologie p-adique, cela devrait être plus facile.

Définition 1.5.1. *Pour tout $u, v \in A^*$, on pose*
- $r(u, v) = \min \{|G| \mid G$ *est un groupe fini qui sépare u et v*$\}$,
- $d(u, v) = 2^{-r(u,v)}$ *(avec* $\min \varnothing = \infty$ *et* $2^{-\infty} = 0$*).*

La fonction d qu'on vient de définir est une distance *ultramétrique*, c'est-à-dire qu'elle satisfait à l'inégalité triangulaire renforçée : $d(u, w) \leqslant \max(d(u, v), d(v, w))$.

En plus, cette distance a d'autres propriétés très agréables :
- Le produit $(u, v) \to uv$ est uniformément continu.
- Les morphismes de monoïde de A^* dans B^* sont uniformément continus.
- Les morphismes de monoïde de A^* dans un groupe fini discret sont uniformément continus.

En termes savants, la topologie définie par la distance d est la topologie initiale définie par les morphismes sur un groupe fini. La propriété suivante est une propriété intéressante de cette topologie qui va nous être utile par la suite.

Proposition 1.5.1. *Une suite $(u_n)_{n \geqslant 0}$ de A^* converge vers un mot u si et seulement si pour tout morphisme $\varphi : A^* \to G$ où G est un groupe fini, $(\varphi(u_n))_{n \geqslant 0}$ est ultimement égal à $\varphi(u)$.*

Un autre corollaire intéressant est qu'une partie de A^* reconnue par un groupe fini est à la fois ouverte et fermée : c'est un *clopen*. En effet, dans un groupe fini muni de la topologie discrète, toute partie est à la fois ouverte et fermée et les morphismes de A^* dans un groupe fini sont continus. Ainsi un langage reconnu par ce groupe, qui est l'image réciproque d'une partie du groupe, est à la fois ouverte et fermée.

Je vais maintenant vous présenter un petit théorème amusant.

Théorème 1.5.2 ([Hal, Reu])**.** *Pour tout mot $u \in A^*$,* $\lim_{n \to \infty} u^{n!} = 1$.

La preuve de ce théorème est très simple. On considère un groupe fini G et un morphisme de monoïde $\varphi : A^* \to G$. Posons $g = \varphi(u)$; si k est l'ordre de G, par le théorème de Lagrange, on a $g^k = 1$. Par conséquent, pour $n \geqslant k, \varphi(u^{n!}) = g^{n!} = 1$ et la suite $\varphi(u^{n!})$ est donc ultimement égale à $\varphi(1)$.

Cette preuve est élémentaire, mais ce résultat à des conséquences intéressantes, en particulier pour notre problème.

Proposition 1.5.2 ([Reu])**.** *Les langages réversibles sont fermés.*

Il faut noter que la réciproque de la Proposition 1.5.2 n'est pas vraie. Le langage a^*b^* est fermé mais n'est pas réversible.

Je vais maintenant vous faire la preuve de la Proposition 1.5.2 sur l'exemple de la figure 1.22.

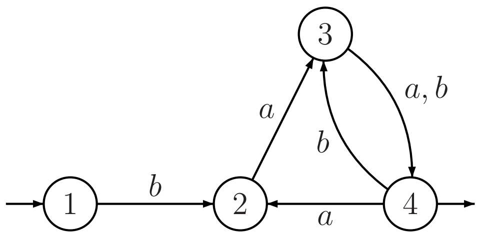


Fig. 1.22 – Le langage reconnu par cet automate est fermé.

Soit L le langage reconnu par l'automate réversible de la figure 1.22. Pour montrer que L est fermé, on va prouver que son complémentaire L^c est ouvert. Pour cela on va montrer que pour tout mot $u \notin L$, il existe un ouvert contenant u et disjoint de L.

Considérons le mot $u = bababa$ qui n'appartient pas au langage L. Je commence par rajouter à l'automate les flèches nécessaires pour pouvoir lire ce mot. Dans l'automate de la figure 1.22, je peux lire *baba*, mais pas *bababa* : je rajoute donc des états et j'obtiens l'automate de la figure 1.23 qui est toujours réversible.

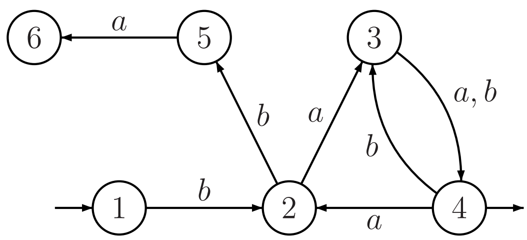


Fig. 1.23 – Dans cet automate réversible, on peut lire le mot *bababa*.

Maintenant, je rajoute des flèches en pointillé pour obtenir un automate de permutations comme sur la figure 1.24. On regarde désormais le groupe G engendré par les permutations associées à a et à b et on note $\varphi : A^* \to G$ le morphisme naturel. On pose $g = \varphi(u) = \varphi(bababa)$. Puisque G est discret, $\{g\}$ est ouvert, et comme φ est continue, $U = \varphi^{-1}(g)$ est aussi ouvert. Si $x \in U$, $\varphi(x) = \varphi(u)$ et donc $1 \cdot x = 6$ mais si $x \in L$, $1 \cdot x = 4$. Ainsi, U est un ouvert qui contient u et qui est disjoint de L.

Cette construction permet aussi de faire une preuve d'un théorème de M. Hall qui dit que les sous-groupes finiment engendrés du groupe libre sont fermés.

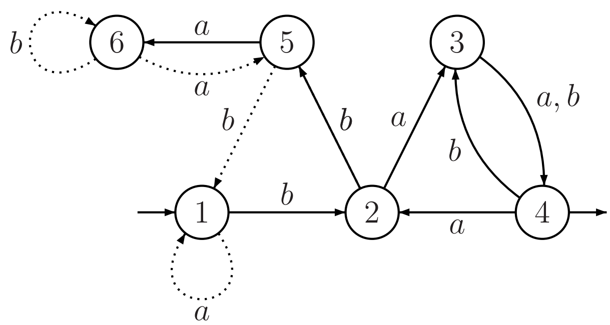


Fig. 1.24 – Automate de permutations obtenu à partir de l'automate de la figure 1.23.

1.6 Un lemme d'itération

Il existe différents lemmes d'itération (ou *lemmes de la pompe*) qui sont très utilisés en informatique. Je vais vous présenter ici un lemme d'itération pour les langages réversibles.

Corollaire 1.6.1. *Soit L un langage réversible et soient $x, u, y \in A^*$. Si, pour tout $n > 0$, $xu^ny \in L$, alors $xy \in L$.*

La preuve de ce corollaire est facile. Puisque le produit est continu, on a $\lim_{n\to\infty} xu^{n!}y = xy$. Or puisque L est réversible, il est fermé, et donc $xy \in L$.

On est repassé d'une propriété topologique, qui disait que les langages réversibles étaient fermés, à une propriété combinatoire satisfaite par les langages réversibles. Je vais maintenant vous présenter une version algébrique de cette propriété.

Proposition 1.6.1. *Soient L un langage reconnaissable et M son monoïde syntactique ordonné. Sont équivalents :*

(1) *L vérifie le lemme d'itération,*

(2) *pour tout idempotent $e \in M$, $1 \leqslant e$.*

Posons $P = \varphi(L)$. Écrire $1 \leqslant e$ signifie que pour tout $m, n \in M$, si $men \in P$, alors $mn \in P$.

Si L vérifie le lemme d'itération, considérons $m, e, n \in M$ tels que $men \in P$ et $ee = e$. Soient $u, x, y \in A^*$ tels que $\varphi(u) = e, \varphi(x) = m, \varphi(y) = n$. Alors pour tout $p \geqslant 1$, $\varphi(xu^py) = me^pn = men \in P$. Donc si L vérifie le lemme d'itération, $xy \in L$ et alors $mn \in P$.

Réciproquement, si pour tout idempotent $e \in M$, $1 \leqslant e$, considérons $u, x, y \in A^*$ tels que pour tout $n > 0$, $xu^ny \in L$, c'est-à-dire que, pour tout n, $\varphi(x)\varphi(u)^n\varphi(y) \in P$. Puisque M est un monoïde fini, il existe n_0 tel que $\varphi(u)^{n_0}$ soit idempotent. On a donc $1 \leqslant \varphi(u)^{n_0}$ et puisque $\varphi(x)\varphi(u)^{n_0}\varphi(y) \in P$, il vient $\varphi(x)\varphi(y) \in P$ et donc $xy \in L$. Le langage L vérifie donc le lemme d'itération.

On a donc obtenu deux conditions nécessaires auxquelles doivent satisfaire les idempotents du monoïde syntactique d'un langage réversible : les idempotents commutent et les idempotents sont plus grand que 1 pour l'ordre syntactique.

1.7 Caractérisation algébrique

Voici maintenant le théorème principal qui nous donne une caractérisation algébrique des langages réversibles.

Théorème 1.7.1. *Soit L un langage reconnaissable et M son monoïde syntactique ordonné. Alors L est réversible si et seulement si*

(1) *les idempotents de M commutent,*

(2) *pour tout idempotent $e \in M$, $1 \leqslant e$.*

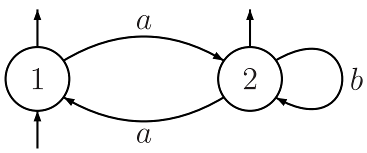
Il faut noter qu'on a bien obtenu le résultat recherché : ce théorème nous permet de décider si un langage rationnel est réversible. En effet, étant donné un langage rationnel, on peut calculer l'automate minimal de ce langage, puis son monoïde syntactique, son ordre syntactique et il suffit de vérifier que les propriétés (1) et (2) sont vérifiées. On donc bien un algorithme de décision ; je reviendrai plus tard sur les questions de complexité.

On a déjà vu que les conditions (1) et (2) étaient nécessaires. La preuve de la réciproque est la partie la plus difficile, je vais maintenant vous donner quelques éléments de preuve. Pour cela, on va regarder de très près le graphe de Cayley du monoïde syntactique du langage. Je vais commencer par vous présenter deux propriétés qui vont nous servir.

Une composante fortement connexe d'un graphe est un sous-graphe induit par un ensemble de sommets tel qu'il existe un chemin dans ce sous-graphe entre deux sommets quelconques de ce sous-graphe. Une composante fortement connexe du graphe de Cayley d'un monoïde est dite *régulière* si l'un de ses sommets est un idempotent.

La première propriété qui va nous servir est la proposition suivante qui n'est pas très facile à prouver et dont la preuve repose sur la théorie des semigroupes finis.

Proposition 1.7.1. *Si les idempotents de M commutent, l'automate défini par une composante régulière est réversible.*



1	1	2
a	2	1
b	-	2
ab	2	-
ba	-	1
aba	1	-
bab	-	-

Relations :
$aa = 1$
$bb = b$

Fig. 1.25 – L'automate minimal et une présentation du monoïde syntactique de $(ab^*a)^*(1+a)$.

Considérons par exemple le langage $L = (ab^*a)^*(1 + a)$. Sur la figure 1.25, j'ai représenté l'automate minimal de L, ainsi qu'une présentation de son monoïde syntactique M. Les idempotents de M sont 1, b, aba et bab et ils commutent.

Le graphe de Cayley de M est représenté sur la figure 1.26. J'ai distingué les quatre composantes régulières du graphe de Cayley de M. Si on se restreint à chacune des composantes connexes régulières, l'automate défini par chacune de ces composantes est réversible.

La seconde propriété dont on a besoin est la suivante. Cette propriété est au cœur de la preuve. C'est une propriété combinatoire de type Ramsey qui est un peu technique et difficile à prouver.

Proposition 1.7.2 ([Ash1])**.** *Soit φ un morphisme de A^* dans un monoïde M dont les idempotents commutent. Alors il existe un entier $N > 0$ tel que tout mot w de A^* se factorise en $w = u_0v_1u_1 \cdots v_ku_k$, avec $u_1, \ldots, u_{k-1}$ non vides et*

(1) *les $\varphi(v_i)$ sont des éléments réguliers de M,*

(2) *ces facteurs réguliers v_i sont maximaux,*

(3) *la longueur totale des autres facteurs est $\leqslant N$.*

Dire que les $\varphi(v_i)$ sont des éléments réguliers de M, cela signifie qu'ils apparaissent dans des composantes fortement connexes (dans le graphe de Cayley de M) dans lesquelles il y a des idempotents.

De plus, on veut que les v_i soient maximaux, cela signifie que si on ajoute la dernière lettre de u_{i-1} (resp. la première lettre de u_i) devant v_i (resp. derrière v_i), on obtient un mot dont l'image par le morphisme φ n'est plus un élément régulier de M.

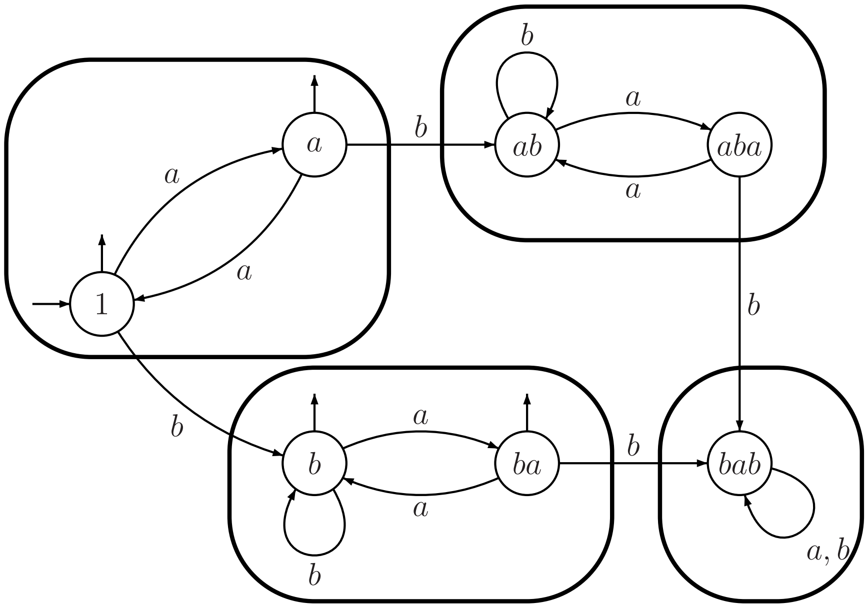


FIG. 1.26 – Le graphe de Cayley du monoïde syntactique de L et ses composantes régulières.

Autrement dit, on peut décomposer chaque mot de A^* en parties régulières et en parties singulières de telle sorte que la taille des parties singulières soit uniformément bornée.

Considérons un mot w et sa factorisation $w = u_0 v_1 u_1 \cdots v_k u_k$ qui nous est donnée par la Proposition 1.7.2. Pour chacun des v_i, on note $\mathcal{B}_i$ l'automate réversible obtenu en considérant la composante fortement connexe du graphe de Cayley de M à laquelle appartient $\varphi(v_i)$ et en prenant l'élément idempotent comme état initial et $\varphi(v_i)$ comme état final.

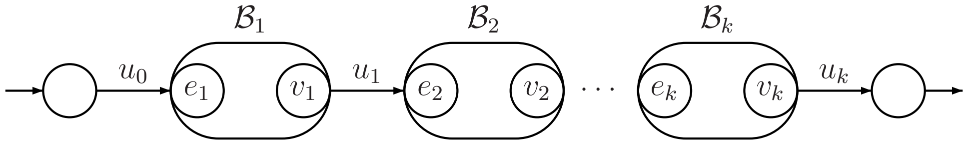


FIG. 1.27 – Construction d'un automate réversible reconnaissant w.

Ensuite, on recolle les $\mathcal{B}_i$ comme sur la figure 1.27 pour obtenir un automate réversible reconnaissant w. Pour recoller les automates, on rajoute un chemin reconnaissant u_i entre l'état final de $\mathcal{B}_i$ et l'état initial de $\mathcal{B}_{i+1}$ pour $i \in \{1, \ldots, k-1\}$, et on ajoute un chemin initial pour lire u_0, et un chemin final pour lire u_k.

On peut montrer en utilisant les propriétés de la factorisation que l'automate ainsi obtenu est réversible. De plus, si le mot $w \in L$, on peut montrer que le langage de l'automate ainsi construit est inclus dans L.

Par ailleurs, il n'y a qu'un nombre fini d'automates de ce type, puisque le graphe de Cayley du monoïde syntactique de L ne contient qu'un nombre fini de composantes fortement connexes et que la longueur des parties singulières du mot w est bornée par N. Cela permet d'exprimer L comme une union finie, certes très large, de langages réversibles, et L est donc un langage réversible.

1.8 Synthèse des résultats

On va maintenant faire la synthèse des résultats et je vous présenterai aussi quelques résultats qui n'ont pas été prouvés ici mais qui peuvent être démontrés avec les mêmes outils.

Je ne vous ai prouvé qu'une direction de la caractérisation algébrique suivante (Proposition 1.3.1), mais c'est bien une propriété caractéristique des langages réversibles. La quatrième assertion du théorème est une variante de cette propriété que je n'avais pas indiquée précédemment.

Théorème 1.8.1. *Soit L un langage de A^*. Sont équivalents :*

(1) *L est réversible,*

(2) *L^c est une combinaison booléenne positive de langages de la forme R ou A^*aR où R est un langage à groupe,*

(3) *L^c est une combinaison booléenne positive de langages de la forme R ou RaA^* où R est un langage à groupe,*

(4) *L^c est une combinaison booléenne positive de langages de la forme R, R_1aR_2 où R, R_1 et R_2 sont des langages à groupe.*

Le théorème suivant résume toutes les autres caractérisations des langages réversibles que je vous ai présentées.

Théorème 1.8.2. *Soient L un langage rationnel et M son monoïde syntactique ordonné. Sont équivalents :*

(1) *L est réversible,*

(2) *$L = K \cap A^*$, où K est une union finie de classes latérales de sous-groupes finiment engendrés du groupe libre $FG(A)$,*

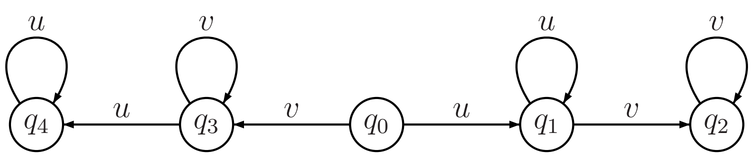
(3) *les idempotents de M commutent et, pour chaque idempotent e de M, $1 \leqslant e$,*

(4) *les idempotents de M commutent et L est fermé.*

Je vais maintenant vous parler des questions algorithmiques liées au problème de décision que je vous ai présenté. On voudrait avoir des algorithmes efficaces pour décider si un langage rationnel est réversible.

En fait, on va retraduire les conditions sur les idempotents du monoïde syntactique M d'un langage L en conditions sur l'automate minimal de L. La première question est de décider si les idempotents de M commutent et on a le résultat suivant.

Théorème 1.8.3. *Soient L un langage rationnel, $\mathcal{A}$ son automate minimal et M son monoïde syntactique ordonné. Les idempotents de M commutent si et seulement si $\mathcal{A}$ ne contient aucune configuration de la forme*



avec $u, v \in A^*$ et $q_2 \neq q_4$.

On peut maintenant se ramener à un problème de théorie des graphes pour obtenir un algorithme polynomial en temps qui permet de décider si les idempotents de M commutent.

La seconde condition que l'on veut tester peut aussi s'exprimer comme une condition que doit vérifier l'automate minimal.

Théorème 1.8.4. *Soient L un langage rationnel, $\mathcal{A}$ son automate minimal et M son monoïde syntactique ordonné. On a $1 \leqslant e$ pour tout idempotent $e \in M$ si et seulement si $\mathcal{A}$ ne contient aucune configuration de la forme*

$$\begin{tikzcd} q_2 & p \arrow[l, "v"'] \arrow[r, "u"] & q \arrow[loop above, "u"] \arrow[r, "v"] & q_1 \end{tikzcd}$$

avec $u, v \in A^*$ et $q_1 \in F$ et $q_2 \notin F$.

Là encore, on peut se ramener à un problème de théorie des graphes, et on obtient un algorithme polynomial en temps pour décider si on a $1 \leqslant e$ pour tout idempotent $e \in M$.

À partir des Théorèmes 1.8.2, 1.8.3 et 1.8.4, on obtient une caractérisation des langages réversibles en termes de propriétés que doit vérifier l'automate minimal.

Théorème 1.8.5. *Soit $\mathcal{A}$ l'automate minimal d'un langage L. Alors L est réversible si et seulement si $\mathcal{A}$ ne contient aucune des deux configurations précédentes.*

De plus, on a maintenant un algorithme efficace pour décider si un langage rationnel est réversible.

Corollaire 1.8.1. *On peut tester en temps polynomial en n si un langage accepté par un automate déterministe à n états est réversible.*

Toutefois, cet algorithme ne permet pas de trouver effectivement un automate réversible reconnaissant un langage réversible. Un de mes étudiants [Hea] a d'ailleurs exhibé une suite de langages réversibles K_n dont l'automate minimal possède $O(n)$ états alors que le nombre minimal d'états d'un automate réversible reconnaissant K_n est en $O(\rho^n)$ où $\rho = \left(\frac{9}{8}\right)^{\frac{1}{12}}$.

1.9 Pour aller plus loin...

Je vais maintenant vous donner quelques indications sur des problèmes connexes et sur des questions ouvertes.

1.9.1 Sur la topologie du groupe libre

Le théorème suivant, dont je vous ai parlé précédemment, est dû à M. Hall Jr. et date de 1950.

Théorème 1.9.1 ([Hal]). *Tout sous-groupe finiment engendré du groupe libre est fermé.*

Avec Reutenauer, nous avions conjecturé une généralisation de ce théorème [PR] ; ce qui nous a permis de recevoir des lettres de gens très célèbres. Cette généralisation a été prouvée en 1993 par Ribes et Zalesskii, mais la preuve est assez difficile.

Théorème 1.9.2 ([RZ]). *Tout produit de sous-groupes finiment engendrés du groupe libre est fermé.*

Depuis, d'autres démonstrations de ce résultat ont été présentées, dont une via la théorie des modèles, due à Herwig et Lascar en 1997 [HL].

Un corollaire de ce théorème est que la fermeture (topologique) d'un langage rationnel est un langage rationnel, ce qui n'est pas facile à prouver. Et en plus, on dispose d'un algorithme pour la calculer, ce qui est encore plus difficile à montrer.

Cela a des conséquences remarquables en théorie des semigroupes finis. Pour en savoir plus sur ce sujet, je vous renvoie à [Hen, AS].

1.9.2 Problèmes ouverts

Je vous ai présenté une topologie construite à partir des groupes, mais on peut se restreindre à des classes de groupes particulières. On peut définir de la même façon, des topologies pro-p-groupe, pro-groupe résoluble, pro-groupe nilpotent.

On a vu que la fermeture d'un langage rationnel est toujours rationnelle dans la topologie pro-groupe. On dispose d'un algorithme qui permet de la calculer pour les topologies pro-groupe, pro-p-groupe [RZ2, MSW], pro-groupe nilpotent [MSW], mais il n'y a pas à ce jour d'algorithme connu pour la topologie pro-groupe résoluble.

Ce problème se ramène à décider si un automate réversible peut être complété, quitte à rajouter des états et des flèches, en un automate à groupe résoluble. Considérons par exemple le groupe symétrique à cinq éléments S_5. Ce groupe est engendré par le cycle (12345) et la transposition (12) : sur la figure 1.28, les actions des lettres a et b engendrent tout le groupe. Si on enlève la flèche étiquetée b qui envoie 5 sur 5, qui est dessinée avec des tirets sur la figure 1.28, on se demande si on peut, en rajoutant des états et des flèches, obtenir un groupe de permutations résoluble, sachant que S_5 ne l'est pas. En fait, personne ne sait répondre à cette question : on ne sait même pas le faire sur un exemple.

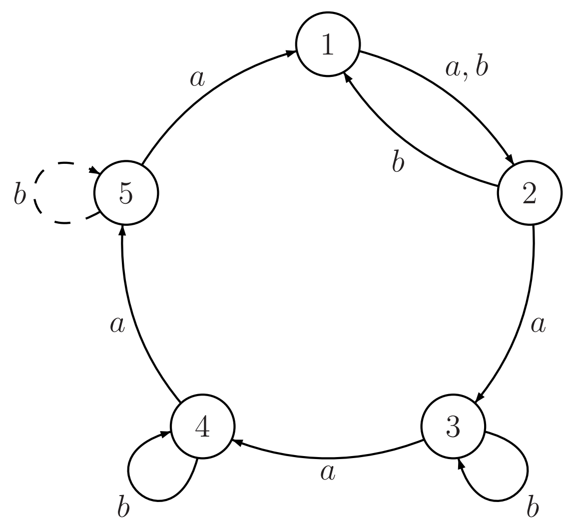


FIG. 1.28 – Les permutations définies par a et b engendre le groupe symétrique à cinq éléments.

Au lieu de regarder des classes de groupes particulières, on peut aussi construire des topologies profinies pour d'autres variétés de monoïdes finis, comme par exemple les monoïdes commutatifs ou les monoïdes dont les idempotents commutent. Le monoïde libre A^* est alors muni d'une structure d'espace métrique dont la complétion est un monoïde compact.

Ces objets encore très mal connus sont la clé de la solution de nombreux problèmes de la théorie des automates. Les automates réversibles sont l'un des exemples que l'on peut décrire simplement mais il y a beaucoup d'autres problèmes que l'on peut exposer de manière très élémentaire et dont la solution passe vraisemblablement par l'étude de ces topologies.

Bibliographie

[Ash1] C.J. Ash : *Finite semigroups with commuting idempotents*, J. Austral. Math. Soc. (Series A) **43** (1987), p. 81–90.

[Ash2] C.J. Ash : *Inevitable Graphs : A proof of the type II conjecture and some related decision procedures*, Int. Jour. Alg. and Comp. **1** (1991), p. 127–146.

[AS] K. Auinger and B. Steinberg : *On power groups and embedding theorems for relatively free profinite monoids*, Math. Proc. Cambridge Philos. Soc. **138** (2005), p. 211–232.

[Hal] M. Hall Jr. : *A topology for free groups and related groups*, Ann. of Maths **52** (1950), p. 127–139.

[Hea] P.-C. Héam, *A lower bound for reversible automata*, Theoret. Informatics Appl. **34** (2000), p. 331–341.

[Hen] K. Henckel, S. Margolis, J.-E. Pin et J. Rhodes, *Ash's type II theorem, profinite topology and Malcev products*, Int. J. Alg. Comput. **1** (1991), p. 411–436.

[HL] B. Herwig et D. Lascar : *Extending partial automorphisms and the profinite topology on free groups*, Trans. Amer. Math. Soc. **352** (2000), p. 1985–2021.

[Kle] S.C. Kleene : *Representation of events in nerve nets and finite automata*, Automata Studies (C.E. Shannon and J. McCarthy eds.), Princeton University Press, Princeton, New Jersey, (1956), p. 3–42.

[MSW] S. Margolis, M. Sapir, and P. Weil : *Closed subgroups in pro-***V** *topologies and the extension problem for inverse automata*, Internat. J. Algebra Comput. **11**, (2001), p. 405–445.

[Pin1] J.-E. Pin : *On the languages recognized by finite reversible automata*, Proceedings of the 14th ICALP, LNCS 267 (1987), p. 237–249.

[Pin2] J.-E. Pin : *Topologies for the free monoid*, Journal of Algebra **137** (1991), p. 297–337.

[Pin3] J.-E. Pin : *On reversible automata*, Proceedings of the first LATIN conference, Saõ-Paulo, LNCS 583 (1992), p. 401–416.

[Pin4] J.-E. Pin : *Topologie p-adique sur les mots*, Journal de théorie des nombres de Bordeaux **5** (1993), p. 263–281.

[PR] J.-E. Pin et C. Reutenauer : *A conjecture on the Hall topology for the free group*, Notices of the London Math. Society **23** (1991), p. 356–362.

[Reu] C. Reutenauer : *Une topologie du monoïde libre*, Semigroup Forum **18** (1979), p. 33–49. *Correction*, Semigroup Forum **22** (1981), p. 93–95.

[RZ] L. Ribes and P.A. Zalesskii : *On the profinite topology on a free group*, Bull. London Math. Soc. **25** (1993), p. 37–43.

[RZ2] L. Ribes and P.A. Zalesskii : *The pro-p topology of a free group and algorithmic problems in semigroups*, Int. J. Algebra Comput. **4** (1994), p. 359–374.

[Sta] J. Stallings : *Topology of finite graphs*, Invent. Math. **71** (1983), p. 551–565.