Contribution à l'algorithmique des automates : complexité et aspects topologiques

Pierre-Cyrille Héam

2 mai 2001

Table des matières

Table des matières 6

Introduction 7

1 Rappels de quelques notions 17
1.1 Introduction . . . 18
1.2 Remarques sur les notations . . . 18
1.3 Semigroupes et monoïdes . . . 19
1.4 Mots et langages . . . 19
1.4.1 Mots . . . 19
1.4.2 Monoïde libre et langages . . . 21
1.4.3 Groupe libre . . . 22
1.5 Parties et expressions rationnelles . . . 23
1.6 Langages rationnels - Automates finis . . . 25
1.6.1 Automates . . . 25
1.6.2 ε-transitions . . . 27
1.6.3 Notations . . . 27
1.6.4 Automates déterministes . . . 28
1.6.5 Automates complets . . . 29
1.6.6 Automates émondés . . . 30
1.6.7 Automate minimal . . . 30
1.6.8 Produit d'automates . . . 32
1.6.9 Complémentaire d'un langage rationnel . . . 34
1.6.10 Graphe d'un automate . . . 34
1.6.11 Langage vide . . . 35
1.6.12 Inclusion de langages . . . 35
1.6.13 Mots reconnus par un automate . . . 35
1.6.14 Automates et expressions rationnelles . . . 36
1.7 Parties rationnelles du groupe libre . . . 36
1.7.1 Parties rationnelles . . . 36
1.7.2 Sous-groupes rationnels . . . 38
1.8 Langages rationnels et monoïdes finis . . . 42

1.8.1 Langages reconnus par un monoïde fini 42
1.8.2 Langages à groupe 44
1.9 Sur la classification des langages rationnels 44
1.9.1 Variétés de langages et de monoïdes finis 45
1.9.2 Langages sans-étoile 46
1.9.3 Hiérarchies de concaténation 47
1.9.4 Variétés de groupes finis 48
1.10 Décomposition unitaire-préfixielle 49
1.10.1 Langages unitaires et langages préfixes 49
1.10.2 Algorithme de décomposition unitaire-préfixielle 50

2 Topologie et parties rationnelles 57
2.1 Introduction et motivations 58
2.2 Topologies pro-**H** 59
2.2.1 Distances pro-**H** 59
2.2.2 Topologies pro-**H** 60
2.2.3 Sous-groupes rationnels ouverts-fermés 62
2.2.4 Topologies sur le monoïde libre 63
2.2.5 Propriétés des langages ouverts et des langages fermés . 63
2.3 Calcul de la clôture d'un sous-groupe rationnel 65
2.3.1 Généralités 65
2.3.2 Théorème de Hall 65
2.3.3 Calcul de la fermeture pro-p d'un sous-groupe rationnel 66
2.4 Calcul de la clôture d'une partie rationnelle 70
2.4.1 Algorithme de Pin-Reutenauer 70
2.4.2 Topologie profinie et parties rationnelles 73
2.4.3 Clôture d'une partie rationnelle donnée par un automate 73
2.4.4 Calcul dans le cas profini 89
2.5 Calcul de la clôture d'un langage rationnel 90
2.5.1 Clôture d'un langage rationnel donné par un automate . 90
2.5.2 Calcul dans le cas profini 91
2.5.3 Calcul dans le cas le pire 96
2.6 Propriétés topologiques des parties rationnelles 100
2.6.1 Langages denses 100
2.6.2 Points d'accumulation 102
2.6.3 Fermeture d'une partie positive 108
2.6.4 Langages non rationnels et topologie profinie 111
2.7 Conclusion 112

3 Automates réversibles 113
3.1 Introduction 113
3.2 Définitions et propriétés caractéristiques 114
3.2.1 Définitions et premières propriétés 114

3.2.2 Propriétés algébriques 117
3.2.3 Automates réversibles et groupe libre 118
3.2.4 Complémentaire d'un automate réversible 118
3.3 Automates réversibles et topologie 121
3.3.1 Propriétés topologiques des langages réversibles 121
3.3.2 Langages fermés et automates réversibles 125
3.4 Algorithmes pour automates réversibles 128
3.4.1 Test de réversibilité 128
3.4.2 Calcul d'un automate réversible 129
3.5 Conclusion 139

4 Hiérarchie des langages à groupe 141
4.1 Introduction et motivations 141
4.2 Niveau 0 : les langages à groupe 142
4.2.1 Résultats connus sur les langages à groupes 143
4.2.2 Propriétés topologiques 144
4.3 Niveau 1/2 : les langages ouverts 148
4.3.1 Résultats connus sur le niveau 1/2 148
4.3.2 Optimisation du test du niveau 1/2 149
4.3.3 Automates localement à groupe 152
4.4 Niveau 1 153
4.4.1 Résultats connus sur le niveau 1 153
4.4.2 Résultats connus sur la variété **ER** 155
4.4.3 Caractérisation des langages reconnaissables par un automate déterministe localement réversible 156
4.4.4 Amélioration du test du niveau 1 161
4.5 Niveau 3/2 et automates localement réversibles 161
4.6 Conclusion 164

5 Hiérarchie de Straubing-Thérien 167
5.1 Introduction et motivations 167
5.2 Niveau 1/2 168
5.2.1 Propriétés connues 168
5.2.2 Automates saturés 170
5.2.3 Propriétés topologiques 179
5.3 Niveau 1 179
5.3.1 Caractérisation des langages de niveau 1 179
5.3.2 Automates déterministes localement triviaux 180
5.4 Niveau 3/2 181
5.4.1 Résultats connus 182
5.4.2 Automates localement triviaux 183
5.4.3 Calcul d'une expression rationnelle polynomiale 183
5.4.4 Propriétés topologiques 184

5.5 Langages sans-étoile et topologie profinie 188
5.6 Conclusion 190

6 Algorithmique des monoïdes 191
6.1 Introduction et motivations 191
6.2 Calcul des noyaux d'un monoïde fini 192
6.2.1 Produit de Mal'cev et décidabilité 192
6.2.2 Cas d'une variété close par extension 193
6.2.3 Calcul du noyau profini 196
6.3 Conclusion 197

Index des notations 198

Index des algorithmes 199

Index 201

Bibliographie 204

Introduction

Cette thèse s'inscrit dans le cadre de l'étude algorithmique des langages rationnels. Plus particulièrement, nous y étudions les algorithmes relatifs à plusieurs classes d'automates (automates à groupe, automates réversibles,...) qui interviennent dans des problèmes d'origines assez diverses, qui vont de la théorie de l'apprentissage à la vérification en passant par la topologie. Notre but sera soit d'améliorer la complexité des algorithmes existants, soit d'établir des bornes de complexité plus précises, soit enfin dans certains cas de caractériser les langages reconnus par telle ou telle classe d'automates.

Les notions de *mot* (suite de lettres) et de *langage* (ensemble de mots) ont été introduites dans les années cinquante par les linguistes dans le but de donner un cadre formel à l'étude des langues naturelles. N. Chomsky mit alors en place une hiérarchie de langages formels dont le premier niveau est constitué des langages *reconnaissables*, c'est-à-dire des langages qui peuvent être décrits à l'aide d'une machine de calcul très simple : *un automate fini.*

L'étude théorique et mathématique des automates finis, motivée par le développement de la théorie des machines de calcul, démarra concrètement avec le résultat fondateur de S.C. Kleene (1956) qui montra que les langages reconnaissables sont exactement les langages *rationnels*, c'est-à-dire les langages que l'on peut décrire à partir des langages finis en utilisant les opérations union, concaténation et étoile (une telle expression d'un langage est appelée *expression rationnelle* ou *expression régulière*). Un pas décisif de cette théorie fut franchi en 1959, quand M.O. Rabin et D. Scott montrèrent comment simuler un automate fini par un automate fini déterministe. La théorie des automates et des langages rationnels a connu depuis de nombreux développements théoriques, motivés notamment par des applications très diverses en informatique. Les automates permettent en effet de modéliser ou de simuler de nombreux problèmes allant de la recherche de motifs dans des textes ou des séquences génétiques au calcul formel, en passant par la compression de données, l'arithmétique des ordinateurs ou la théorie du contrôle. L'optimisation des algorithmes sous-jacents nécessite une étude approfondie des langages correspondants. Comme on le verra dans cette thèse, cette étude peut conduire à l'introduction de notions assez abstraites, de nature algébrique ou topologique, qui se révèlent nécessaires pour la justification d'algorithmes d'apparence beaucoup plus concrète.

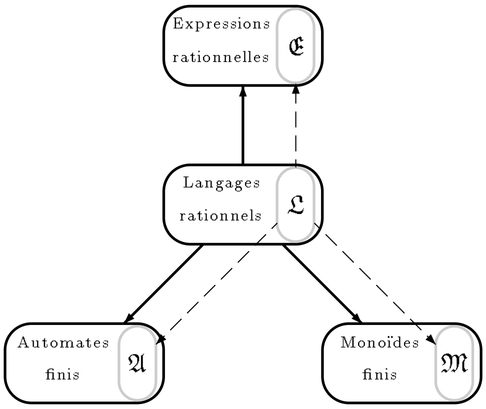
* *
*

Pour les langages rationnels, la complexité du passage d'une forme de représentation à une autre est assez bien connue. Et même si la recherche est encore active dans le domaine (notamment pour le passage d'une expression rationnelle à un automate fini), on dispose actuellement d'algorithmes dont la complexité est proche de la complexité optimale.

Cependant, comme nous l'avons déjà souligné, de nombreux problèmes liés aux automates proviennent d'un cadre extérieur à la théorie des langages rationnels et n'utilisent en fait qu'une classe réduite de langages. On est alors confronté à des problèmes spécifiques de cette classe, ou à des problèmes généraux pour lesquels il convient d'utiliser les propriétés particulières de cette classe afin d'obtenir des algorithmes plus efficaces. On aboutit là, plus généralement, à deux types de problèmes.

- Le premier type de problème est le problème du test. Etant donnée une classe $\mathfrak{L}$ de langages reconnaissables, comment peut-on (et avec quelle efficacité) tester si un langage appartient à $\mathfrak{L}$? La question se déclinant bien sûr à chaque fois selon la représentation du langage : par un automate, par une expression rationnelle ou par un monoïde fini. Par exemple, il est facile de savoir si un langage donné par un automate émondé est fini (il suffit de tester s'il contient des boucles).
- Le second type de problème est lié à la représentation du langage. En effet, une classe $\mathfrak{L}$ de langages reconnaissables est souvent donnée par un type particulier d'automate (définissant une classe $\mathfrak{A}$ d'automates), ou par un type particulier d'expression rationnelle ou de monoïde fini (définissant respectivement une classe $\mathfrak{E}$ d'expressions rationnelles et une classe $\mathfrak{M}$ de monoïdes). Le problème est alors, pour un langage de $\mathfrak{L}$ donné, de calculer, le plus efficacement possible, un automate de $\mathfrak{A}$

(ou une expression de $\mathfrak{E}$, ou un monoïde de $\mathfrak{M}$) qui le décrit.



C'est dans le cadre de cette problématique que j'ai effectué mes recherches.

* *
*

Je me suis d'abord intéressé à la classe des langages reconnaissables par un automate réversible[1] ou langages réversibles. Cette classe a été introduite par D. Angluin [Ang82] en théorie de l'apprentissage et constitue une généralisation de la classe des langages "reset-free". Elle apparaît, de façon curieuse, dans des domaines très divers de la théorie des automates : pour le calcul de la hauteur d'étoile [LS00], en théorie des codes [Hal84], ou, plus récemment, en liaison avec les modèles quantiques [MC00, KW97] : ainsi, A. Ambrainis et R. Freivalds ont montré [AF98] que la classe des langages réversibles correspond exactement à la classe des langages reconnaissables par un type particulier d'automate quantique.

Le problème du test a été résolu, pour la classe des langages réversibles, par J.-E. Pin [Pin92] : on peut décider en temps polynomial si un langage, donné par un automate déterministe, est reconnaissable par un automate réversible.

1. Un automate réversible est un automate fini dans lequel chaque lettre induit une fonction (partielle) injective de l'ensemble des états dans lui-même. Mais on autorise plusieurs états initiaux (resp. finaux)

La justification de l'algorithme proposé utilise une topologie introduite par M. Hall [Hal50] en 1950 en théorie des groupes. Mais curieusement, même si on sait *décider* en temps polynomial si un langage rationnel (donné par un automate déterministe) est réversible, on ne sait pas *trouver* de façon efficace un automate réversible qui le reconnaît. C'est ce dernier problème qui a attiré mon attention.

- J'ai établi une borne inférieure en espace (et donc en temps) exponentielle pour le problème du calcul d'un automate réversible. Plus précisément, il existe une famille de langages K_n représentables par des automates déterministes de taille $O(n)$ et représentables par des automates réversibles dont la taille ne peut être inférieure à $O(r^n)$ avec $r > 1$. Ce résultat est décrit dans [Héa01b]. Déterminer une borne supérieure "raisonnable" reste pour le moment un problème ouvert que j'étudie actuellement.

- Un autre de mes résultats confirme que la notion de réversibilité est étroitement liée à la topologie de Hall. En utilisant des arguments topologiques (convergence) je montre que l'automate minimal d'un langage qui est ouvert ou fermé pour cette topologie a toutes ses composantes fortement connexes réversibles [2].

* *

*

Ces travaux m'ont naturellement conduit à approfondir l'étude de la topologie de Hall, appelée aussi topologie profinie. Un bref historique permettra de mieux situer mes travaux dans ce domaine. La topologie de Hall fut introduite à l'origine sur le groupe libre [Hal50] et fut étendue au monoïde libre par C. Reutenauer [Reu79, Reu81]. Elle permit d'abord de fournir un angle d'attaque [Pin88, PR91] à une conjecture relative à des problèmes de décidabilité en théorie des semigroupes finis proposée par J. Rhodes [Rho87] : la décidabilité de certaines variétés de semigroupes finis se réduit au calcul de la fermeture (pour la topologie profinie) de parties rationnelles du groupe libre. La conjecture de Rhodes fut finalement prouvée par C.J. Ash [Ash91] et celle de [PR91] par L. Ribes et P.A. Zalesskii [RZ93]. Mis à part le problème du test pour les automates réversibles déjà mentionné [Pin92], la topologie de Hall conduit à un algorithme polynomial pour tester si un langage rationnel, donné par un automate déterministe, est de niveau 1/2 de la hiérarchie des langages à groupes [Pin94].

- En premier lieu, j'ai étudié la complexité du calcul de la fermeture (topologique) d'un langage rationnel. On sait en effet [PR91, RZ93] que cette

2. C'est-à-dire que chaque lettre définit une fonction (partielle) dont la restriction à chaque composante fortement connexe est injective.

fermeture est encore rationnelle. Je donne un algorithme linéaire dans le cas d'une partie rationnelle du groupe libre donnée par un automate. J'en déduis un algorithme en temps $O(n^3)$ pour calculer la fermeture d'un langage donné par un automate à n états. Je montre aussi que ces résultats peuvent s'étendre à d'autres topologies (voisines de celle de Hall), à condition que l'on sache calculer la fermeture d'un sous-groupe rationnel du groupe libre. Ces résultats ont fait l'objet d'une publication [Héa01a]. Enfin, je montre que la taille de l'automate minimal de la fermeture d'un langage donné par un automate déterministe est au plus exponentielle et que, dans le pire des cas, cette complexité est atteinte.

- Dans la continuité de ce travail, j'ai étudié les propriétés topologiques des langages rationnels et je montre que l'ensemble des points d'accumulation d'un langage rationnel est aussi rationnel et calculable en temps polynomial (lorsque le langage d'origine est donné par un automate déterministe). Ce résultat a été publié dans [Héa01d].

- Je me suis intéressé ensuite à une propriété mise en avant par J. Almeida et M. Delgado [AD99, AD00] lorsqu'ils ont cherché à relier les résultats de L. Ribes et P.A. Zalesskii [RZ93], de C.J. Ash [Ash91] et de B. Herwig et D. Lascar [HL00] : certaines parties rationnelles du groupe libre ne peuvent pas être obtenues comme la fermeture d'un langage rationnel du monoïde libre. Je montre que l'on peut décider en temps $O(n^3)$ si une partie rationnelle du groupe libre donnée par un automate déterministe est la fermeture profinie d'un langage rationnel. De plus, dans le cas d'un sous-groupe rationnel du groupe libre, l'algorithme devient linéaire. Ce résultat a été publié dans [Héa01d].

- En dernier lieu, j'ai effectué aussi un retour sur les problèmes originaux rappelés plus haut relatifs aux problèmes de décisions en algèbre. J'utilise mes algorithmes pour calculer en temps $O(n^3)$ le noyau profini d'un monoïde fini. On retrouve la même complexité que dans [Pin88] mais l'algorithme semble plus efficace pour tester si un élément particulier est dans le noyau. Dans le cas des noyau pro-p, j'obtiens une complexité en $O(n^5)$. Ce résultat a été publié dans [Héa01a].

* *
*

La troisième partie de ma thèse est consacrée aux hiérarchie de concaténation, et on verra que la topologie de Hall joue un rôle important dans l'une d'elle.

A l'origine, ces hiérarchies sont apparues de façon naturelle comme mesure de complexité des expressions rationnelles : elles sont basées sur le nombre d'imbrications entre les opérations booléennes d'une part, et le produit de concaténation d'autre part. W. Thomas [Tho82] en a donné une interprétation logique très convaincante, puisqu'on retrouve la hiérarchie (dite Σ_n) basée sur l'alternance des quantificateurs.

Je me suis intéressé à deux de ces hiérarchies. La première fut introduite implicitement par D. Thérien [Thé81] et explicitement par H. Straubing [Str85]. Les langages de bas niveau (inférieur à deux) dans cette hiérarchie ont été plus particulièrement étudiés. Il n'y a rien à dire sur les langages de niveau 0 : $\emptyset$ et A^*.

- Les langages de niveau 1/2 sont les unions finies de langages de la forme

$$A^* a_1 A^* a_2 A^* \cdots A^* a_k A^* \tag{1}$$

où les a_i sont des lettres. On sait que ces langages correspondent exactement aux idéaux de mélange[3] [Hig52]. On sait tester en temps polynomial [PW97] si un langage, donné par un automate déterministe à n états, est de niveau 1/2. J'ai tout d'abord souligné que ces langages sont aussi exactement ceux reconnus par des automates dont les composantes fortement connexes sont des automates de permutations à un seul état et j'en ai déduit un algorithme sensiblement plus efficace (en $O(n^2)$) pour tester si un langage est de niveau 1/2. De plus, je montre que calculer une expression du type (1) pour un langage de niveau 1/2 donné par un automate peut se faire en temps exponentiel et que cette complexité est atteinte dans le pire des cas.

- Les langages de niveau 1 sont les langages que l'on peut obtenir comme une combinaison booléenne de langages de niveau 1/2. Les fortes propriétés de ce niveau [Sim75, Alm90, ST85] ont permis l'élaboration d'un algorithme polynomial de décision d'appartenance à ce niveau (pour un langage donné par un automate déterministe) [Ste85]. Pour souligner un parallèle avec un résultat que nous verrons plus tard, je redémontre le résultat suivant [Sim75] : tout langage du niveau 1 est reconnaissable par un automate déterministe dont les composantes fortement connexes sont réduites à un élément.

- Je me suis ensuite intéressé aux langages de niveau 3/2. Ce sont les langages pouvant s'exprimer comme une union finie de langages du type :

$$A_0^* a_1 A_1^* \cdots A_{k-1}^* a_k A_k^* \tag{2}$$

où les a_i sont des lettres et les A_i des sous-ensembles de l'alphabet. C'est actuellement le niveau le plus élevé que l'on sache décider. Le premier algorithme de décision a été donné par M. Arfi [Arf87, Arf91] alors que le premier algorithme polynomial a été donné par J.-E. Pin et P. Weil [PW97]. Récemment, ils ont été utilisés pour la vérification de systèmes [ABJ98, AAB99] et

3. Un langage L est un idéal de mélange si, pour tout mot u et tout sous-mot v de u, $v \in L$ implique $u \in L$.

A. Bouajjani, A. Muscholl et T. Touili [BMT01] ont montré que ce niveau est stable par semi-commutation et que l'on peut décider en temps polynomial si un langage donné par un automate déterministe est de niveau 3/2 (sans considérer que la taille de l'alphabet est une constante, contrairement à l'algorithme de [PW97]). J'ai établi une borne inférieure en espace (et donc en temps) exponentielle pour le problème du calcul d'une expression du type (2) : il existe une famille de langages K_n représentables par des automates déterministes de taille $O(n)$ et représentables par des expressions du type (2) dont la taille ne peut être inférieure à $O(\rho^n)$ avec $\rho > 1$. Ce résultat est décrit dans [Héa01c]. De plus, je montre que les langages du niveau 3/2 ont de remarquables propriétés topologiques : la fermeture d'un langage de niveau 3/2, pour la topologie de Hall, est un langage de niveau 3/2.

- Les langages de niveau 2 sont les langages qui peuvent s'exprimer comme une combinaison booléenne de langages de niveau 3/2. On sait actuellement décider si un langage est de niveau 2 que dans des cas restreints [Str88, Cow93, PW97]. Je montre que ce niveau est aussi le niveau à partir duquel on perd la stabilité par fermeture profinie : je donne l'exemple d'un langage de niveau 2 dont la fermeture n'appartient ni au niveau 2, ni à aucun niveau de cette hiérarchie.

* *
*

L'autre hiérarchie que j'ai étudiée et qui est étroitement liée à la topologie de Hall est celle dite des "langages à groupe".

- Le niveau 0 de cette hiérarchie est constitué des langages à groupe, c'est-à-dire des langages dont l'automate minimal est un automate de permutation. Bien qu'on n'en connaisse pas de description satisfaisante en terme d'expression rationnelle, ils interviennent fréquemment en théorie des automates. Citons par exemple les problèmes de complexité étudiés par J.-C. Birget [Bir92] ou encore l'étude de la génération aléatoire d'automates abordée par C. Nicaud [Nic00]. De plus, à l'instar des langage réversibles, la classe des langages à groupe correspond exactement à une classe de langages reconnaissables par un type particulier d'automate quantique [BP99]. Enfin, ils interviennent bien entendu dans l'étude de la topologie de Hall. De façon plus précise, les langages à groupe constituent une base d'ouverts-fermés pour cette topologie et sont caractérisés par la propriété suivante : un langage rationnel est de niveau 0 si et seulement s'il est à la fois ouvert et fermé pour la topologie profinie. Je me suis tout d'abord intéressé à cette propriété et j'ai montré que l'hypothèse de rationnalité était nécessaire : il existe des langages non rationnels qui sont à la fois ouverts et fermés pour la topologie profinie.

- Le niveau 1/2 de la hiérarchie des langages à groupe est constitué des langages qui sont une union finie de langages du type

$$L_0 a_1 L_1 \cdots L_{k-1} a_k L_k \tag{3}$$

où les L_i sont des langages à groupe et les a_k des lettres de l'alphabet. M. Beaudry, F. Lemieux et D. Thérien [BLT97] ont montré qu'un langage est de niveau 1/2 si et seulement si c'est une union finie de produits de langages à groupe, si et seulement s'il est reconnaissable par une structure algébrique finie particulière appelée *loop*. Enfin, ces langages interviennent aussi dans l'étude de la topologie de Hall : un langage rationnel est ouvert si et seulement s'il est de niveau 1/2 [Pin94], ce qui conduit à un algorithme polynomial [PW97] pour décider si un langage donné par un automate déterministe est de niveau 1/2. J'ai commencé à montrer le parallèle entre cette hiérarchie et celle de Straubing-Thérien (au niveau des automates), en remarquant qu'un langage est de niveau 1/2 si et seulement s'il est reconnaissable par un automate dont les composantes fortement connexes sont des automates de permutations. Ceci m'a conduit à améliorer le test de décision pour ce niveau : on peut tester en temps $O(n^3)$ si un langage donné par un automate déterministe est de niveau 1/2 dans la hiérarchie des langages à groupe.

- Les langages de niveau 1 sont les langages qui peuvent s'exprimer comme une combinaison booléenne de langages du niveau 1/2. Ils possèdent de jolies caractérisations syntaxiques [HR91, Pin95a, MP87] qui conduisent à un algorithme polynomial [PW97] pour le problème de décision correspondant. Je montre ici que tout langage de niveau 1 est reconnaissable par un automate déterministe dont les composantes fortement connexes sont réversibles. Ensuite, en utilisant une construction de S. Eilenberg [Eil74, Eil76], je donne une caractérisation syntaxique précise des langages reconnaissables par ce type d'automates[4].

- On ne sait pas actuellement décider si un langage est de niveau 3/2 dans la hiérarchie des langages à groupe. J'ai cependant établi un résultat analogue à celui sur les langages de niveau 3/2 de la hiérarchie de Straubing-Thérien : un langage est de niveau 3/2 dans la hiérarchie des langages à groupe si et seulement s'il est reconnaissable par un automate dont les composantes fortement connexe sont réversibles, si et seulement si c'est une union finie de langages du type :

$$R_0 a_1 R_1 \cdots R_{k-1} a_k R_k \tag{4}$$

où les R_i sont des langages réversibles et les a_i des lettres.

4. Il s'agit des langages dont les idempotents du monoïde syntaxique engendrent un monoïde $\mathcal{R}$-trivial.

* *
*

Ces résultats sont présentés dans six chapitres dont voici une brève description.

Le premier chapitre est un rappel des notions basiques sur les langages rationnels. J'y décris en particulier les différents modes de reconnaissabilité : par expression rationnelle, par automate et par monoïde fini.

Dans le second chapitre, je commence par présenter la topologie de Hall, dite aussi topologie profinie, ainsi que d'autres topologies qui en sont voisines. Je montre comment calculer, pour certaines de ces topologies, la fermeture d'un langage donné par un automate fini. Je montre aussi que, dans certains cas, l'ensemble des points d'accumulation d'un langage rationnel est aussi un langage rationnel. Je finis en donnant un résultat de décidabilité concernant les parties rationnelles fermées du groupe libre.

Je décris dans le troisième chapitre les propriétés topologiques des langages reconnaissables par un automate réversible et je montre que si un langage est donné par un automate déterministe et s'il est reconnaissable par un automate réversible, alors, dans le pire des cas, un tel automate possède au moins un nombre exponentiel d'états.

J'étudie dans le quatrième chapitre la hiérarchie des langages à groupe : je commence par donner différentes propriétés topologiques des langages à groupe, puis j'étudie la classe des langages reconnaissables par un automate déterministe dont les composantes fortement connexes sont réversibles : j'en donne une caractérisation syntaxique précise et je montre qu'elle contient strictement la classe des langages de niveau 1. J'en déduis enfin une caractérisation des langages du niveau 3/2 : ce sont exactement les langages reconnaissables par un automate (non nécessairement déterministe) dont les composantes fortement connexes sont réversibles.

Le cinquième chapitre est consacré à l'étude de la hiérarchie de Straubing-Thérien. J'y donne quelques propriétés topologiques des langages des premiers niveaux en insistant sur leur représentation par des automates particuliers. J'établis aussi des bornes inférieures de complexité pour le calcul d'expressions rationnelles particulières liées aux niveaux 1/2 et 3/2.

Dans le dernier chapitre, je présente une application des résultats du chapitre 2 à la théorie des monoïde finis. Plus particulièrement, j'étudie comment l'approche topologique permet de calculer efficacement les noyaux des monoïdes finis.

Chapitre 1

Rappels de quelques notions

Sommaire

1.1	**Introduction**	**18**
1.2	**Remarques sur les notations**	**18**
1.3	**Semigroupes et monoïdes**	**19**
1.4	**Mots et langages**	**19**
1.4.1	Mots	19
1.4.2	Monoïde libre et langages	21
1.4.3	Groupe libre	22
1.5	**Parties et expressions rationnelles**	**23**
1.6	**Langages rationnels - Automates finis**	**25**
1.6.1	Automates	25
1.6.2	ε-transitions	27
1.6.3	Notations	27
1.6.4	Automates déterministes	28
1.6.5	Automates complets	29
1.6.6	Automates émondés	30
1.6.7	Automate minimal	30
1.6.8	Produit d'automates	32
1.6.9	Complémentaire d'un langage rationnel	34
1.6.10	Graphe d'un automate	34
1.6.11	Langage vide	35
1.6.12	Inclusion de langages	35
1.6.13	Mots reconnus par un automate	35
1.6.14	Automates et expressions rationnelles	36
1.7	**Parties rationnelles du groupe libre**	**36**
1.7.1	Parties rationnelles	36
1.7.2	Sous-groupes rationnels	38
1.8	**Langages rationnels et monoïdes finis**	**42**
1.8.1	Langages reconnus par un monoïde fini	42
1.8.2	Langages à groupe	44

1.9 Sur la classification des langages rationnels 44
1.9.1 Variétés de langages et de monoïdes finis 45
1.9.2 Langages sans-étoile 46
1.9.3 Hiérarchies de concaténation 47
1.9.4 Variétés de groupes finis 48
1.10 Décomposition unitaire-préfixielle 49
1.10.1 Langages unitaires et langages préfixes 49
1.10.2 Algorithme de décomposition unitaire-préfixielle . . 50

1.1 Introduction

Nous rappelons dans ce chapitre les résultats fondamentaux de la théorie des langages rationnels dans laquelle s'inscrit cette thèse. Nous posons ici les notations et définitions utiles pour la suite, sans pour autant chercher à faire un cours sur le sujet : nous souhaitons juste éviter toute ambiguïté.

Nous commencerons par de brefs rappels sur les monoïdes et semigroupes (pour plus de détails, notamment dans le cadre de l'étude des langages rationnels, se reporter à [Pin84]). Nous poursuivrons avec les bases de la théorie des mots (pour un exposé plus détaillé, le lecteur se reportera à [Eil74] ou à [Lot83]). Nous aborderons ensuite la théorie des langages rationnels et des automates finis qui pourra être approfondie par la lecture de [Ber79, Eil74, HU80].

Nous supposons que le lecteur est familier avec les concepts mathématiques fondamentaux tels que morphisme, groupe, quotient, opérations booléennes sur les ensembles, relation d'équivalence, convergence ...

1.2 Remarques sur les notations

Nous précisons ici quelques conventions d'écriture.

Si X et Y sont deux ensembles, φ une application de X dans Y et $x \in X$, l'image de x par φ sera notée $x\varphi$. Contrairement à l'usage anglo-saxon, on rappelle qu'une application de X vers Y est définie pour tout $x \in X$, alors qu'une fonction de X dans Y n'est pas nécessairement définie pour tout $x \in X$.

Si E est un ensemble fini, on note $|E|$ son cardinal. Si E et F sont deux ensembles tels que $F \subseteq E$, on note $E \setminus F$ l'ensemble des éléments de E qui ne sont pas dans F. S'il n'y a pas d'ambiguïté sur E, on notera aussi F^c.

Si G est un groupe et $K \subseteq G$, on note $\langle K \rangle$ le sous-groupe de G engendré par K.

On notera $\mathbb{N}$ l'ensemble des entiers positifs.

1.3 Semigroupes et monoïdes

Définitions 1.1 *Un* semigroupe *est un couple $(S, \cdot)$, où S est un ensemble et $\cdot$ une loi de composition interne associative sur S. Un* monoïde *est un triplet $(M, \cdot, 1_M)$ où M est un ensemble, $\cdot$ une loi de composition interne associative sur M et $1_M \in M$ est un élément neutre pour $\cdot$.* semigroupe monoïde

Par abus de langage et de notation, on nomme souvent un semigroupe ou un monoïde par son ensemble sous-jacent, la loi de composition (et l'élément neutre, dans le cas d'un monoïde) sont sous-entendus.

Par exemple, $\mathbb{N}$ muni de la multiplication est un monoïde dont l'élément neutre est 1. L'ensemble des applications d'un ensemble dans lui-même est un monoïde (pour la composition des applications), dont l'élément neutre est l'identité. Tout groupe est un monoïde. L'ensemble $\mathbb{N}\setminus\{0\}$, muni de l'addition, n'est pas un monoïde (il n'y a pas d'élément neutre) ; en revanche, c'est un semigroupe.

On notera multiplicativement la loi interne dans les semigroupes et dans les monoïdes.

Une notion algébrique très utile pour décrire les semigroupes est la notion d'*idempotence*.

Définition 1.2 *Un élément e d'un semigroupe est dit* idempotent *si $e^2 = e$.* idempotent

Proposition 1.3 *Soient S un semigroupe fini et x un élément de S. Il existe un unique idempotent de S qui soit une puissance de x. On le note x^ω.*

Pour un semigroupe S, on note $E(S)$ l'ensemble de ses idempotents.

Soit M un monoïde et $X \subseteq M$. On note X^* le sous-monoïde de M engendré par X, c'est-à-dire le plus petit sous-monoïde de M contenant X. De même, si X est une partie d'un semigroupe S, on note par X^+ le sous-semigroupe de S engendré par X. Par ailleurs, si X et Y sont deux parties d'un monoïde, XY désigne l'ensemble des xy où x décrit X et y décrit Y. On définit alors naturellement et récursivement X^k, pour $k \in \mathbb{N}$, par $X^0 = \{1_M\}$, $X^1 = X$ et $X^{k+1} = X^k X$. On a alors étoile

$$X^* = \bigcup_{k \geq 0} X^k \quad \text{et} \quad X^+ = \bigcup_{k \geq 1} X^k$$

Lorsque $X = \{x\}$ est un singleton, on note souvent, par abus, x^* et x^+ à la place de $\{x\}^*$ et $\{x\}^+$.

1.4 Mots et langages

1.4.1 Mots

Avant d'aborder la notion de *mot*, il nous faut définir les *lettres*.

alphabet lettre

Définitions 1.4 *On appelle* alphabet *tout ensemble fini A. Les éléments de A sont appelés* lettres.

Dans toute la suite, A désigne un alphabet fini dont les éléments sont usuellement notés a, b, c, a_i ou b_j ... selon le contexte.

mot longueur

Définitions 1.5 *Soit A un alphabet fini. On appelle* mot *toute application u d'un ensemble du type $\{0,\ldots,n-1\}$ où $n \in \mathbb{N} \setminus \{0\}$ dans A. L'entier n s'appelle alors* longueur de *u et il est noté $|u|$.*

De façon usuelle, un mot u de longueur n est noté

$$(0u)(1u)(2u)\ldots((n-1)u)$$

Par exemple, le mot aab sur l'alphabet $A = \{a, b\}$ est l'application de $\{0, 1, 2\}$ dans A qui à 0 associe a, à 1 associe a et à 2 associe b.
Avec notre définition, nous n'avons défini que les mots de longueur strictement positive.

mot vide

Définition 1.6 *On appelle* mot vide*, noté ε, l'unique application de l'ensemble vide dans A. Par convention, la longueur du mot vide est nulle.*

Dans la littérature, le mot vide est aussi parfois noté Λ, 1 ou encore λ.

préfixe

Définition 1.7 *On appelle* préfixe *d'un mot u sur l'alphabet A tout mot v sur A tel que $|v| \leq |u|$ et pour tout $0 \leq i \leq |v|-1$, $iu = iv$.*

Cette définition correspond au sens intuitif de préfixe ; les préfixes de $abaab$ sont ε, a, ab, aba, $abaa$ et $abaab$. Il est à noter que le mot vide est préfixe de tout mot et que tout mot est préfixe de lui-même.

On définit de même la notion de *suffixe* d'un mot.

suffixe

Définition 1.8 *On appelle* suffixe *d'un mot u sur l'alphabet A tout mot v sur A tel que $|v| \leq |u|$ et pour tout $0 \leq i \leq |v|-1$, $(i + |u| - |v|)u = iv$.*

Définissons enfin un *sous-mot.*

sous-mot

Définition 1.9 *Soit u un mot sur l'alphabet A. On appelle* sous-mot *de u tout mot v sur A tel qu'il existe une fonction strictement croissante φ de l'ensemble $\{0,\ldots,|v|-1\}$ dans $\{0,\ldots,|u|-1\}$ vérifiant : pour tout i dans l'ensemble de définition de φ, $iv = i\varphi u$.*

Un sous-mot d'un mot u est donc un mot que l'on peut construire à partir de u en y “effaçant” des lettres. Par exemple, $abab$ est un sous-mot de $u = abbaba$: on l'obtient en effaçant le premier b et le dernier a.

Le *miroir* d'un mot u est obtenu en inversant le sens de u. Plus formellement, on le définit comme suit.

miroir

Définition 1.10 *On appelle* miroir *d'un mot u sur l'alphabet A le mot, noté $u^\sim$, défini par $|u^\sim| = |u|$ et pour tout $0 \leq i \leq |u|-1$, $iu^\sim = (|u|-1-i)u$.*

Par exemple, le miroir de $abbaaaa$ est $aaaabba$. Un mot qui est égal à son miroir est appelé *palindrome*. Par exemple, si on omet le caractère blanc et la ponctuation, les phrases “Esope reste ici et se repose” ou “A man, a plan, a canal: Panama” sont des phrases palindromes. palindrome

1.4.2 Monoïde libre et langages

Définition 1.11 *Le produit de deux mots u et v sur A est le mot sur A de longueur $|u|+|v|$, noté uv tel que pour tout $0 \leq i \leq |u|-1$, $i(uv) = iu$ et pour tout $|u| \leq i \leq |v|+|u|-1$, $i(uv) = (i-|u|)v$* produit de mots

Le produit de deux mots correspond simplement à la juxtaposition de ces deux mots. Par exemple, le produit de $abbab$ par bab est le mot $abbabbab$.

La notion de produit nous permet de définir les *facteurs* d'un mot.

Définition 1.12 *Un mot u sur l'alphabet A est* facteur *d'un mot v sur A s'il existe deux mots w_0 et w_1 sur A tels que $v = w_0uw_1$.* facteur

Par exemple, bab est facteur de $aababa$.

On peut aussi définir les suffixes et les préfixes d'un mot en utilisant le produit des mots comme le montre la proposition suivante.

Proposition 1.13 *Soit u un mot sur l'alphabet A. Un mot v sur A est préfixe [resp. suffixe] de u si et seulement s'il existe un mot w sur A tel que $u = vw$ [resp. $u = wv$].*

En particulier tout préfixe et tout suffixe d'un mot sont facteurs de ce mot.

La propriété suivante se vérifie aisément.

Proposition 1.14 *Le produit des mots de A est une loi de composition interne associative possédant un unique élément neutre ε.*

Ce qui justifie (en partie) l'appellation suivante.

Définition 1.15 *L'ensemble des mots sur A se note A^* et s'appelle* monoïde libre *engendré par A.* monoïde libre

La proposition 1.14 justifie l'appellation de monoïde.

Proposition 1.16 *Le monoïde libre sur A est le seul monoïde (à isomorphisme près) tel que pour tout monoïde M et toute application φ de A dans M, il existe un unique morphisme de monoïde de A^* dans M prolongeant φ.*

Définition 1.17 *On appelle* langage *de A^* tout sous-ensemble de A^*.* langage

Un langage est donc un ensemble de mots, qui peut être fini ou infini. Par exemple, l'ensemble des mots sur $A = \{a, b\}$ de longueur paire, ou l'ensemble des mots contenant un nombre premier de a, ou encore l'ensemble des mots admettant $abbab$ comme préfixe sont des langages infinis de A^*.

mélange

Définition 1.18 *Soient u et v deux mots de A^*. Le* mélange *de u et v, noté $u \sqcup\!\sqcup v$, est le langage constitué des mots $u_0v_0u_1v_1 \dots u_kv_k$ tels que $u = u_0u_1 \dots u_k$ et $v_0v_1 \dots v_k = v$, où $u_i \in A^*$ et $v_i \in A^*$.*

Par exemple, le mélange de ab et ba est le langage $\{abba, abab, baba, baab\}$. La définition s'étend aux langages.

Définition 1.19 *On appelle* mélange *de deux langages L_1 et L_2 de A^* le langage noté $L_1 \sqcup\!\sqcup L_2$ constitué de l'union des $u \sqcup\!\sqcup v$ où u décrit L_1 et v décrit L_2.*

Remarquons ici que dans la littérature, le mot *shuffle* est souvent employé à la place de mélange.

1.4.3 Groupe libre

De la même façon que l'on a construit le monoïde libre, on va construire le groupe libre, c'est-à-dire que l'on va agrandir le monoïde libre(i.e. lui ajouter des éléments) afin de lui donner une structure de groupe.

On travaille sur un alphabet A. On note $\overline{A}$ un ensemble équipotent à A et disjoint de A, c'est-à-dire une copie de A. Soient φ une bijection de A sur $\overline{A}$ et $\tilde{A} = A \cup \overline{A}$. On note $\bar{\ }$ la bijection de $\tilde{A}$ dans lui-même définie par :

$$\left\{ \begin{array}{lllll} \tilde{A} & \to & \tilde{A} & & \\ x & \mapsto & \overline{x} & = x\varphi & \text{si } x \in A \\ x & \mapsto & \overline{x} & = x\varphi^{-1} & \text{si } x \in \overline{A} \end{array} \right.$$

On définit alors $\equiv$ comme étant la congruence sur $\tilde{A}^*$ engendrée par les couples $(a\overline{a}, 1)$ pour $a \in \tilde{A}$. Plus simplement, deux mots u et v sur $\tilde{A}$ sont dits équivalents si l'on peut passer de l'un à l'autre en intercalant ou en supprimant des facteurs du type $a\overline{a}$. Par exemple les mots $ab\overline{b}\overline{a}bb$ et $ba\overline{a}b$ sont équivalents car en partant du premier, on peut d'abord supprimer $b\overline{b}$ pour obtenir $a\overline{a}bb$, puis $a\overline{a}$ pour obtenir bb dans lequel on peut intercaler $a\overline{a}$ et obtenir ainsi le second mot.

réduit

Définition 1.20 *Un mot de $\tilde{A}^*$ est dit* réduit *s'il ne contient aucun facteur du type $a\overline{a}$ pour $a \in \tilde{A}$.*

En particulier, en considérant que $A^* \subseteq \tilde{A}^*$ (en fait, A^* s'injecte de façon canonique dans $\tilde{A}^*$), on remarque que tous les mots de A^* sont réduits.

Proposition-Définition 1.21 *Toute classe d'équivalence de la relation $\equiv$ contient un unique mot réduit. Si $u \in \tilde{A}^*$, on note* red$u$ *l'unique mot réduit qui lui est équivalent.*

On prolonge l'application $\bar{\cdot}$ en une application sur $\tilde{A}^*$ de façon récursive en posant, pour tout $u \in \tilde{A}^*$, $\overline{ua} = \overline{a}\,\overline{u}$ et $\overline{\varepsilon} = \varepsilon$.

La relation $\equiv$ est compatible avec le produit (c'est une congruence) sur $\tilde{A}^*$ et avec l'application $\bar{\cdot}$, c'est-à-dire que si u et v sont équivalents, alors $\overline{u}$ et $\overline{v}$ aussi. Il en découle que l'ensemble $\tilde{A}^*/\equiv$ est un groupe, que l'on appelle *groupe libre* engendré par A (on peut vérifier que, si l'on prend une bijection différente de A sur $\overline{A}$, alors on obtient un groupe isomorphe), et on le note $F(A)$. Comme dans le cas du monoïde libre, on peut prolonger les morphismes, ce qui s'exprime par la propriété universelle suivante : groupe libre

Proposition 1.22 *Le groupe libre $F(A)$ est l'unique groupe (à isomorphisme près) tel que, pour tout groupe G et toute application φ de A dans G, il existe un unique morphisme de groupe prolongeant φ de $F(A)$ dans G.*

Pour finir, si K est un sous-ensemble de $\tilde{A}^*$, on note $D(K)$ l'ensemble des mots réduits de K. On note π la projection naturelle de $\tilde{A}^*$ dans $F(A)$. On peut remarquer que, grâce à la proposition 1.21, la restriction de π à $D(\tilde{A}^*)$ est bijective, ce qui permet d'identifier $F(A)$ avec les éléments réduits de $\tilde{A}^*$ et, de ce fait, on peut aussi considérer que A^* est inclus dans $F(A)$. En faisant cela, A^* devient un sous-monoïde de $F(A)$.

1.5 Parties et expressions rationnelles

L'ensemble des langages rationnels de A^* est une famille de langages de A^* possédant suffisamment de propriétés pour en permettre une étude particulière et efficace, tout en restant suffisamment vaste pour décrire un nombre important de problèmes ou de situations.

Définition 1.23 *Soit M un monoïde. On définit la famille des* parties rationnelles *de M comme étant la plus petite famille de parties de M stable par union, produit, étoile (monoïde engendré) et contenant les sous-ensembles finis de M.* partie rationnelle

En particulier, si M est un monoïde, le vide est une partie rationnelle et si M est finiment engendré, M est aussi une partie rationnelle.

Les *expressions rationnelles* sont un moyen descriptif de représenter les parties rationnelles d'un monoïde. Soit M un monoïde engendré par un ensemble fini A. On définit les expressions rationnelles de façon récursive. expression rationnelle

Définition 1.24

- *$\emptyset$ est une expression rationnelle.*

- *Pour tout élément $a \in A$, a est une expression rationnelle.*
- *Si $\mathcal{E}$ et $\mathcal{F}$ sont deux expressions rationnelles, alors $\mathcal{E}\mathcal{F}$ est une expression rationnelle.*
- *Si $\mathcal{E}$ et $\mathcal{F}$ sont deux expressions rationnelles, alors $(\mathcal{E} \cup \mathcal{F})$ est une expression rationnelle.*
- *Si $\mathcal{E}$ est une expression rationnelle, alors $(\mathcal{E})^*$ est une expression rationnelle.*

Une expression rationnelle est donc une formule écrite avec les symboles $a \in A$, $*$ et $\cup$. Il est à noter que lorsqu'il n'y a pas d'ambiguïté, on omet parfois d'écrire les parenthèses. Les expressions rationnelles permettent de décrire des parties d'un monoïde.

Définition 1.25 *On définit l'application ν de l'ensemble des expression rationnelles dans l'ensemble des parties de M par les propriétés suivantes :*

- $\emptyset\nu = \emptyset$
- *Pour tout élément $a \in A$, $a\nu = \{a\}$.*
- *Si $\mathcal{E}$ et $\mathcal{F}$ sont deux expressions rationnelles, alors $(\mathcal{E}\mathcal{F})\nu = (\mathcal{E}\nu)(\mathcal{F}\nu)$.*
- *Si $\mathcal{E}$ et $\mathcal{F}$ sont deux expressions rationnelles, alors $(\mathcal{E} \cup \mathcal{F})\nu = \mathcal{E}\nu \cup \mathcal{F}\nu$.*
- *Si $\mathcal{E}$ est une expression rationnelle, alors $(\mathcal{E})^*\nu = (\mathcal{E}\nu)^*$.*

Prenons le cas de $M = \{a, b\}^*$ et considérons l'expression rationnelle $\mathcal{E} = (ab \cup a^*)^*$. On a $\mathcal{E}\nu = (\{ab\} \cup \{a\}^*)^*$. Remarquons de plus que dans tout monoïde M, $\emptyset^*\nu = \{1_M\}$.

reconnue

Définition 1.26 *Soit $\mathcal{E}$ une expression rationnelle. On appelle partie* associée *à $\mathcal{E}$ ou partie* représentée *par $\mathcal{E}$ ou encore partie* reconnue *par $\mathcal{E}$ le sous-ensemble $\mathcal{E}\nu$ de M.*

Proposition 1.27 *Un sous-ensemble L de M est rationnel si et seulement s'il existe une expression rationnelle $\mathcal{E}$ telle que $\mathcal{E}\nu = L$.*

Une expression rationnelle est donc un moyen de représenter une partie rationnelle. Il est à noter que deux expressions rationnelles différentes peuvent représenter le même sous-ensemble. Comme on va s'intéresser à des problèmes algorithmiques liés aux parties rationnelles, il est nécessaire de définir une notion de *taille* d'une expression rationnelle qui est le nombre de symboles qui la constituent et qui représente approximativement la taille d'un codage de l'expression. On la définit récursivement.

Définition 1.28 *La fonction τ de l'ensemble des expressions rationnelles dans $\mathbb{N}$ appelée taille est définie par :*

- $\emptyset\tau = 0$.

- *Pour tout élément* $a \in A$, $a\tau = 1$.
- *Si* $\mathcal{E}$ *et* $\mathcal{F}$ *sont deux expressions rationnelles, alors* $(\mathcal{E}\mathcal{F})\tau = \mathcal{E}\tau + \mathcal{F}\tau$
- *Si* $\mathcal{E}$ *et* $\mathcal{F}$ *sont deux expressions rationnelles, alors* $(\mathcal{E}\cup\mathcal{F})\tau = \mathcal{E}\tau+\mathcal{F}\tau+1$.
- *Si* $\mathcal{E}$ *est une expression rationnelle, alors* $(\mathcal{E}^*)\tau = 1 + \mathcal{E}\tau$.

Par exemple, la taille de $\mathcal{E} = (ab \cup a^*)^*$ est 6.

Comme on vient de le dire, il n'y pas unicité de la représentation d'un sous-ensemble par expression rationnelle. On peut donc définir la taille d'une partie rationnelle comme la taille minimale d'une expression la reconnaissant. On note alors, pour toute partie rationnelle L,

$$e(L) = \min\{\mathcal{E}\tau \mid \mathcal{E}\nu = L\}$$

1.6 Langages rationnels - Automates finis

1.6.1 Automates

On vient de voir que, dans le cas général, les parties rationnelles sont exactement celles représentées par une expression rationnelle. Dans le cas du monoïde libre, la situation est encore plus agréable car celles-ci sont aussi exactement les langages reconnaissables par des automates finis.

Définitions 1.29 *Un* automate fini *est un quintuplet* (Q, A, E, I, F) *où* — automate

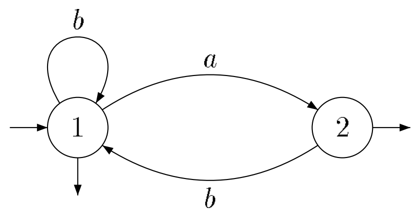
- Q *est un ensemble fini dont les éléments sont appelés* états.
- A *est un alphabet.*
- E *est l'ensemble des* transitions *; c'est un sous-ensemble de* $Q \times A \times Q$. *Si* $(p, a, q) \in E$, p *est appelé* état d'origine, q état d'arrivée *et* a étiquette *de la transition.* — transition
- I *est un sous-ensemble de* Q *dont les éléments sont appelés* états initiaux.
- F *est un sous-ensemble de* Q *dont les éléments sont appelés* états finaux. — états initiaux et finaux

Généralement, un automate sera noté $\mathcal{A}$. Pour simplifier les notations, lorsque I est réduit à un singleton $\{i\}$, on note (Q, A, E, i, F) au lieu de $(Q, A, E, \{i\}, F)$. La même remarque vaut si F est un singleton.

Graphiquement, on représente un automate fini par un graphe dans lequel un état q est représenté par un cercle étiqueté q et la transition (p, a, q) par une flèche étiquetée par a et allant de p à q. Si un état est initial, on le représente par une petite flèche entrant dans le cercle et s'il est final par une petite flèche sortante.

Dans cette thèse tous les automates considérés sont finis et on omettra parfois de le préciser.

Exemple 1.30 Le dessin suivant



représente l'automate $(\{1,2\},\{a,b\},\{(1,a,2),(1,b,1),(2,b,1)\},1,\{1,2\})$.

$\triangledown$

On va maintenant montrer comment décrire des langages grâce aux automates. Pour cela, on utilise la notion de chemin dans un automate.

chemin

Définitions 1.31 *Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate fini. On appelle* chemin *de $\mathcal{A}$ toute suite finie m de transitions de $\mathcal{A}$*

$$(p_1, a_1, q_1), (p_2, a_2, q_2), \ldots, (p_k, a_k, q_k)$$

telle que $q_i = p_{i+1}$. L'entier k est appelé longueur *du chemin et le mot $a_1a_2 \ldots a_k$, noté $[m]_{\mathcal{A}}$, ou $[m]$ s'il n'y a pas d'ambiguïté, est appelé* étiquette *du chemin. De même que pour une transition, p_1 est appelé* origine *du chemin et q_k* état d'arrivée *ou* état terminal. *De plus, si $p_1 \in I$ et $q_k \in F$, le chemin est dit* réussi.

réussi

Exemple 1.32 Dans l'exemple 1.30,

$$(2, b, 1), (1, b, 1), (1, b, 1)$$

est un chemin de longueur 3, d'étiquette bbb, d'origine 2 et d'état terminal 1.

$\triangledown$

On peut maintenant définir le langage reconnu par un automate.

reconnu

Définition 1.33 *Soit $\mathcal{A}$ un automate fini sur l'alphabet A. Un mot $u \in A^*$ est dit* reconnu *par $\mathcal{A}$ s'il existe un chemin réussi de $\mathcal{A}$ d'étiquette u. L'ensemble des mots reconnus par $\mathcal{A}$ est noté $L(\mathcal{A})$.*

Exemple 1.34 Dans l'exemple 1.30, le mot ba est reconnu par l'automate car il est l'étiquette du chemin réussi

$$(1, b, 1), (1, a, 2)$$

$\triangledown$

Définition 1.35 *Un langage L de A^* est dit* reconnaissable *s'il existe un automate $\mathcal{A}$ tel que $L(\mathcal{A}) = L$. On dit alors que $\mathcal{A}$* reconnaît *ou* accepte *L.*

langage reconnaissable

Le lien avec les langages rationnels est donné par un résultat fondamental de S.C. Kleene [Kle56].

Théorème 1.36 *Un langage de A^* est rationnel si et seulement s'il est reconnaissable par un automate fini.*

Exemple 1.37 Dans l'exemple 1.30, on peut montrer que l'automate reconnaît le langage rationnel $(b^*ab)^*(a \cup \emptyset^*)$.

$\triangledown$

1.6.2 ε-transitions

Dans un automate fini, les transitions sont étiquetées par des lettres. On sera amené par la suite à considérer des automates dans lesquels on utilise aussi des transitions étiquetées par ε, que l'on appelle *ε-transitions*. Utiliser une telle transition permet de passer d'un état à un autre sans lire de lettre. On peut montrer assez facilement que permettre ces ε-transitions n'augmente pas le pouvoir d'expression des automates (on ne reconnaît pas plus de langages), et qu'il est même très facile de les *éliminer*, c'est à dire de transformer un automate avec ε-transitions en un automate sans ε-transition (et reconnaissant le même langage). Leur utilité est avant tout pratique : les ε-transitions permettent d'éviter des lourdeurs d'écriture et de clarifier certaines preuves.

ε-transition

Sauf mention du contraire, tous les automates considérés seront sans ε-transition.

1.6.3 Notations

Afin de facilité l'écriture, on utilise les notations suivantes.
Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate fini. On note

- $\mathcal{A}^{p,q}$ (avec $p, q \in Q$), l'automate (Q, A, E, p, q).
- $\mathcal{A}^p$ (avec $p \in Q$), l'automate $(Q, A, E, p, p) = \mathcal{A}^{p,p}$.

- $p \cdot_{\mathcal{A}} a$ (avec $p \in Q$ et $a \in A$), l'ensemble des $q \in Q$ tels que $(p, a, q) \in E$. S'il n'y pas d'ambiguïté sur $\mathcal{A}$, on note simplement $p \cdot a$. De plus, par abus de notation, lorsque $p \cdot a = \{q\}$ est un singleton, on note aussi souvent $p \cdot a = q$.

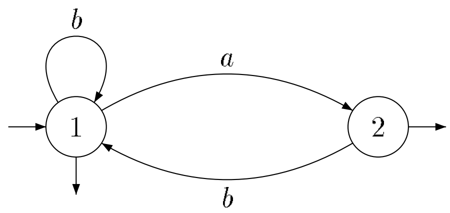
1.6.4 Automates déterministes

déterministe

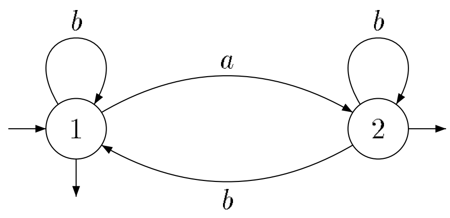
Définition 1.38 *Un automate sur l'alphabet A est dit* déterministe *si pour chacun de ses états p et chaque lettre a, $|p \cdot a| \leq 1$ et si, de plus, il ne possède qu'un seul état initial.*

La première condition traduit le fait que lorsque l'on est dans un état de l'automate et que l'on doit lire une lettre particulière, soit on ne peut pas la lire, soit on ne peut emprunter qu'une seule transition. L'action d'une lettre est déterminée. La seconde condition sert pour initialiser le mouvement. Si l'on veut lire un mot dans l'automate, encore une fois, on n'a pas de choix, on sait où l'on doit commencer : il n'y a qu'un seul état initial.

Exemple 1.39



Un automate déterministe



Un automate non déterministe $(2 \cdot b = \{1, 2\})$

∇

Dans un automate déterministe chaque lettre induit une fonction de l'ensemble des états dans lui-même.

Les automates déterministes s'avèrent extrêmement utiles d'un point de vue algorithmique.

Un résultat fondamental de M.O. Rabin et D. Scott [RS59] nous dit que les automates déterministes ont le même pouvoir d'expression que les automates en général.

Théorème 1.40 *Tout langage rationnel est accepté par un automate déterministe.*

De plus, en partant d'un automate quelconque, on peut construire un automate déterministe reconnaissant le même langage et de taille exponentielle dans le pire des cas. Nous ne décrirons pas cet algorithme classique ici (voir, par exemple, [HU80]).

1.6.5 Automates complets

Définition 1.41 *Un automate sur l'alphabet A est dit* complet *si pour chacun de ses états p et chaque lettre a, $|p \cdot a| \geq 1$.* complet

Un automate complet est donc un automate dans lequel en partant de chaque état, on peut lire n'importe quelle lettre.

Proposition 1.42 *Soit $\mathcal{A}$ un automate fini. Il existe un automate $\mathcal{A}'$ complet tel que $L(\mathcal{A}') = L(\mathcal{A})$.*

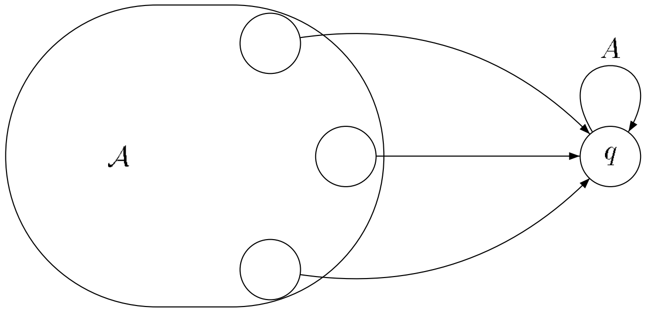
Classiquement, si l'automate $\mathcal{A} = (Q, A, E, I, F)$ n'est pas complet, on lui ajoute un état dans lequel vont toutes les transitions manquantes : soit $q \notin Q$, on pose $Q' = Q \cup \{q\}$,

$$E' = E \cup \{(p, a, q) \mid p \cdot_{\mathcal{A}} a = \emptyset\} \cup \{(q, a, q) \mid a \in A\}$$

On a alors $\mathcal{A}' = (Q', A, E', I, F)$. Il est à noter que, si $\mathcal{A}$ est déterministe, alors $\mathcal{A}'$ aussi.

L'automate $\mathcal{A}'$ ainsi construit s'appelle *complété de $\mathcal{A}$*. Lorsque $\mathcal{A}$ est complet il est égal à son complété. complété

Cette construction peut être illustrée par la figure suivante :



Dans un automate déterministe complet, chaque lettre induit une application de l'ensemble des états dans lui-même.

1.6.6 Automates émondés

accessible **Définition 1.43** *Un état q d'un automate $\mathcal{A}$ est dit* accessible *s'il existe un chemin dans $\mathcal{A}$ partant d'un état initial et arrivant en q.*

co-accessible **Définition 1.44** *Un état q d'un automate $\mathcal{A}$ est dit* co-accessible *s'il existe un chemin dans $\mathcal{A}$ partant de q et arrivant dans un état final.*

émondé **Définition 1.45** *Un automate est dit* émondé *si tous ses états sont à la fois accessibles et co-accessibles.*

Du point de vue du langage reconnu, un état qui n'est pas à la fois accessible et co-accessible ne présente aucun intérêt. En effet, il n'y a alors aucun chemin réussi qui passe par q : dans un automate fini, si l'on supprime tous les états non accessibles et tous les états non co-accessibles ainsi que toutes les transitions partant d' un tel état ou arrivant dans un tel état, on obtient un nouvel automate émondé reconnaissant le même langage. On dit alors que l'on *émonde* l'automate. Si l'automate initial est déterministe, l'automate émondé l'est également.

1.6.7 Automate minimal

Deux automates différents peuvent reconnaître le même langage, même s'ils sont déterministes et émondés.

Définitions 1.46 *Soient $\mathcal{A} = (Q, A, E, i, F)$ et $\mathcal{A}' = (Q', A, E', i', F')$ deux automates déterministes et φ une application de Q dans Q'. On dit que φ est*
morphisme *un* morphisme d'automate *si*

- *Si $(p, a, q) \in E$ alors $(p\varphi, a, q\varphi) \in E'$.*

- $i\varphi = i'$.
- *Si* $p \in F$ *alors* $p\varphi \in F'$.

Si φ est bijective et que sa réciproque est un morphisme d'automate, on dit que c'est un isomorphisme d'automate.

En utilisant les morphismes d'automate, on peut montrer qu'il existe, à isomorphisme près, un unique automate déterministe ayant un nombre minimal d'états reconnaissant un langage rationnel donné. Pour le construire, on a besoin des résultats suivants.

Définitions 1.47 *Soit L un langage de A^*. Pour tout $u \in A^*$, on définit les langages $Lu^{-1} = \{v \in A^* \mid vu \in L\}$ et $u^{-1}L = \{v \in A^* \mid uv \in L\}$ appelés respectivement* quotient droit *et* quotient gauche *de L par rapport à u. La notion de quotient s'étend aux langages : si K est un langage de A^* alors* quotient

$$K^{-1}L = \bigcup_{u \in K} u^{-1}L \quad \text{et} \quad LK^{-1} = \bigcup_{u \in K} Lu^{-1}$$

On utilise parfois le terme de *résiduel* à la place de quotient. Les deux propositions suivantes sont dues à A. Nerode [Ner58].

Proposition 1.48 *Un langage est rationnel si et seulement s'il a un nombre fini de quotients gauches.*

De même,

Proposition 1.49 *Un langage est rationnel si et seulement s'il a un nombre fini de quotients droits.*

On peut alors construire un automate déterministe particulier.

Proposition-Définition 1.50 *Soit L un langage rationnel de A^*. On pose*

- $Q = \{u^{-1}L \mid u \in L\}$
- $E = \{(u^{-1}L, a, v^{-1}L) \mid (ua)^{-1}L = v^{-1}L\}$
- $I = \{L\}$
- $F = \{u^{-1}L \mid u \in L\}$.

L'automate $\mathcal{A} = (Q, A, E, I, F)$ est déterministe et complet et reconnaît L. De plus, si $\mathcal{A}'$ est un automate déterministe complet reconnaissant L, il existe un morphisme surjectif d'automate de $\mathcal{A}'$ sur $\mathcal{A}$. Tout automate déterministe complet reconnaissant L a au moins $|Q|$ états et s'il en a $|Q|$, il est isomorphe à $\mathcal{A}$. L'automate émondé obtenu à partir de $\mathcal{A}$ s'appelle automate minimal *de L. Enfin, on note $m(L)$ le nombre d'états de $\mathcal{A}$.* automate minimal

Par ailleurs, on peut calculer de façon efficace, à partir d'un automate déterministe, l'automate minimal du langage qu'il reconnaît, comme cela a été montré par J. Hopcroft [Hop71]. Une description précise de l'algorithme se trouve dans [HU80]. On dit alors que l'on *minimise* l'automate.

Proposition 1.51 *Soit $\mathcal{A}$ un automate déterministe à n états sur l'alphabet A. On peut calculer en temps $O(|A|n \log n)$ l'automate minimal de $L(\mathcal{A})$.*

Si l'on a deux automates déterministes, on peut donc, en les minimisant, les comparer et savoir ainsi s'ils reconnaissent le même langage. On peut aussi, comme on le verra à la section 1.6.12, tester s'ils se contiennent l'un l'autre.

Une propriété très classique de l'automate minimal nous sera utile par la suite.

Proposition 1.52 *Soient $\mathcal{A} = (Q, A, E, i, F)$ un automate minimal et p et q dans Q tels que pour tout mot $u \in A^*$, $p \cdot u \in F$ si et seulement si $q \cdot u \in F$. Alors $p = q$.*

Ce qui nous conduit à définir la notion d'*états équivalents*.

équivalents

Définition 1.53 *Soit $\mathcal{A} = (Q, A, E, i, F)$ un automate déterministe. Deux états p et q de Q sont dits* équivalents *si pour tout mot $u \in A^*$, $p \cdot u \in F$ si et seulement si $q \cdot u \in F$.*

1.6.8 Produit d'automates

Une construction simple permet, à partir de langages donnés par des automates, de construire un automate reconnaissant l'union des deux langages et un automate reconnaissant leur intersection.

Proposition 1.54 *Soient $\mathcal{A}_0 = (Q_0, A, E_0, I_0, F_0)$ et $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ deux automates. On pose*

- $Q = Q_0 \times Q_1$
- $E = \{((p_0, p_1), a, (q_0, q_1)) \mid (p_0, a, q_0) \in E_0 \text{ et } (p_1, a, q_1) \in E_1\}$

L'automate $\mathcal{A}_2 = (Q, A, E, I_0 \times I_1, F_0 \times F_1)$ reconnaît $L(\mathcal{A}_0) \cap L(\mathcal{A}_1)$. De plus, si $\mathcal{A}_0$ et $\mathcal{A}_1$ sont déterministes, alors $\mathcal{A}_2$ l'est aussi.

Si les automates $\mathcal{A}_0$ et $\mathcal{A}_1$ sont complets, alors l'automate

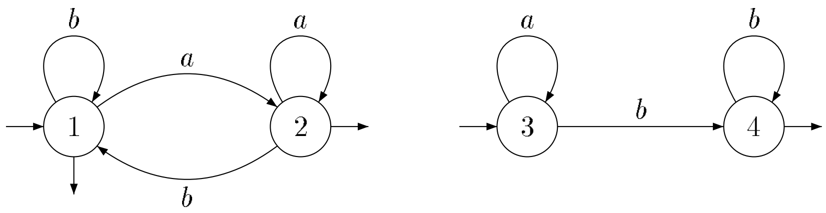
$$\mathcal{A}_3 = (Q, A, E, I_0 \times I_1, F_0 \times Q_1 \cup Q_0 \times F_1)$$

reconnaît $L(\mathcal{A}_0) \cup L(\mathcal{A}_1)$. De plus, si $\mathcal{A}_0$ et $\mathcal{A}_1$ sont déterministes, alors $\mathcal{A}_3$ l'est aussi.

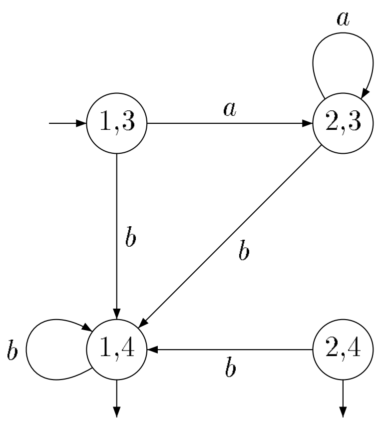
produit d'automates

L'automate $\mathcal{A}_2$ construit ci-dessus s'appelle *produit des automates* $\mathcal{A}_0$ et $\mathcal{A}_1$.

Exemple 1.55 Cherchons un automate déterministe qui reconnaît l'intersection des deux langages donnés par les automates ci-dessous.



On effectue le produit des deux automates et on obtient l'automate déterministe suivant.



∇

Dans la pratique, on ne calcule que la partie accessible du produit en commençant par les couples d'états initiaux.

Une conséquence importante de cette construction est le résultat suivant.

Corollaire 1.56 *L'ensemble des langages rationnels de A^* est clos par intersection.*

1.6.9 Complémentaire d'un langage rationnel

On vient de voir comment calculer l'intersection et l'union de deux langages donnés par des automates. Il est également facile de calculer le complémentaire d'un langage donné par un automate déterministe.

Proposition 1.57 *Soit $\mathcal{A} = (Q, A, E, i, F)$ un automate déterministe complet. L'automate $\mathcal{A}' = (Q, A, E, i, Q \setminus F)$ est déterministe complet et vérifie*

$$L(\mathcal{A}') = A^* \setminus L(\mathcal{A})$$

Corollaire 1.58 *L'ensemble des langages rationnels de A^* est stable par complémentaire.*

1.6.10 Graphe d'un automate

Un automate peut être considéré comme un graphe étiqueté dont on particularise certains états (pour plus de renseignements sur les graphes, notamment d'un point de vue algorithmique, se reporter, par exemple, à [CLR90]).

graphe sous-jacent

Définition 1.59 *Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate. On appelle* graphe sous-jacent *à $\mathcal{A}$ ou plus simplement* graphe de $\mathcal{A}$, *le graphe (Q, E)*

On peut alors définir des propriétés de l'automate relativement à son graphe.

composante fortement connexe

Définitions 1.60 *On appelle* composante fortement connexe *d'un automate $\mathcal{A} = (Q, A, E, I, F)$ tout couple du type (Q', E') où Q' est une composante fortement connexe du graphe de $\mathcal{A}$ et où*

$$E' = (Q' \times A \times Q') \cap E$$

Toute transition de E' est dite interne *à la composante fortement connexe. Toute transition qui n'est interne à aucune composante fortement connexe est dite* externe.

transition interne ou externe

La proposition suivante est très classique et est décrite, par exemple, dans [CLR90].

Proposition 1.61 *Soit $\mathcal{A}$ un automate fini à n états et m transitions. On peut calculer en temps $O(n + m)$ les composantes fortement connexes de $\mathcal{A}$.*

1.6.11 Langage vide

Soit $\mathcal{A}$ un automate fini, on peut facilement tester si $L(\mathcal{A}) = \emptyset$.

Proposition 1.62 *Soit $\mathcal{A}$ un automate fini à n états et m transitions. On peut tester en temps $O(n+m)$ si $L(\mathcal{A})$ est vide.*

Pour cela, il suffit de construire un arbre couvrant du graphe de $\mathcal{A}$ partant de chaque état initial et de regarder si l'on atteint un état final. Si c'est le cas, le langage est non vide, sinon le langage est vide. Cela revient à rechercher sides états finaux sont accessibles.

1.6.12 Inclusion de langages

En utilisant les propositions 1.54 et 1.57, on peut, étant donnés deux automates finis $\mathcal{A}_0$ et $\mathcal{A}_1$ tels que $\mathcal{A}_1$ soit déterministe, tester si $L(\mathcal{A}_0) \subseteq L(\mathcal{A}_1)$. Pour cela, il suffit de calculer un automate reconnaissant $L(\mathcal{A}_1)^c$ puis de calculer son intersection avec $L(\mathcal{A}_0)$. En effet, cette intersection est vide si et seulement si $L(\mathcal{A}_0) \subseteq L(\mathcal{A}_1)$. On peut donc effectuer ce test en temps polynomial.

Si les deux automates sont non déterministes, le problème est en général plus compliqué comme l'ont montré A. Aho, J. Hopcroft et J. Ullman [AHU74].

Proposition 1.63 *Soient $\mathcal{A}_0$ et $\mathcal{A}_1$ deux automates. Tester dans le cas général si $L(\mathcal{A}_0) \subseteq L(\mathcal{A}_1)$ est PSPACE-complet.*

Cela signifie que si $P \neq PSPACE$ (ce qui n'est pas prouvé mais très fortement conjecturé), alors on ne peut pas tester l'inclusion des langages rationnels donnés par des automates non déterministes en temps polynomial.

1.6.13 Mots reconnus par un automate

On teste facilement si un mot u est reconnu par un automate déterministe $\mathcal{A}$. Il suffit pour cela de suivre le chemin partant de l'état initial et d'étiquette u et de regarder si l'on arrive dans un état final. Mais si l'automate est non déterministe, il se peut que l'on ne sache pas soit par où commencer, soit quelles transitions emprunter. Dans ce cas, on construit un automate déterministe $\mathcal{A}'$ reconnaissant u, puis on teste si $L(\mathcal{A}') \cap L(\mathcal{A})$ est vide en effectuant le produit des deux automates.

Proposition 1.64 *Soit $\mathcal{A}$ un automate à n états et m transitions sur l'alphabet A et $u \in A^*$. On peut tester en temps $O((n+m)|u|)$ si $u \in L(\mathcal{A})$.*

1.6.14 Automates et expressions rationnelles

Par le théorème 1.36, on sait que tout langage exprimable par une expression rationnelle est reconnaissable par automate et réciproquement. Il existe de nombreux algorithmes, avec de nombreuses nuances, pour passer d'une représentation à l'autre. Nous ne donnons ici que quelques résultats en la matière.

Dans [HM00], C. Hagenah et A. Muscholl montrent comment calculer efficacement un automate non déterministe reconnaissant un langage donné par une expression rationnelle. Un algorithme similaire a été récemment développé par J.-M Champarnaud et D. Ziadi [CZ00].

Proposition 1.65 *Soit $\mathcal{E}$ une expression rationnelle de taille n. On peut calculer en temps $O(n\log^2(n))$ un automate non déterministe reconnaissant L et ayant $O(n)$ états et $O(n\log^2(n))$ transitions.*

Réciproquement, les choses ne fonctionnent pas aussi bien. A. Ehrenfeucht et P. Zeiger [EZ76] ont en effet montré que certains langages rationnels ne peuvent être représentés que par des expressions rationnelles de taille exponentielle en le nombre d'états de leur automate minimal. De même, il est montré dans [HU80] que l'automate minimal d'un langage peut posséder un nombre exponentiel d'états en la taille d'une expression rationnelle représentant le langage.

1.7 Parties rationnelles du groupe libre

1.7.1 Parties rationnelles

On vient de voir (théorème 1.36) qu'on pouvait décrire les langages rationnels du monoïde libre par des automates finis. Un résultat similaire a été montré par M. Benois [Ben69] pour le groupe libre.

Rappelons que π est la projection naturelle de $\tilde{A}^*$ dans $F(A)$.

Théorème 1.66 *Un sous-ensemble S du groupe libre est rationnel si et seulement si $D(S\pi^{-1})$ est un langage rationnel de $\tilde{A}^*$.*

On peut en déduire les corollaires suivants :

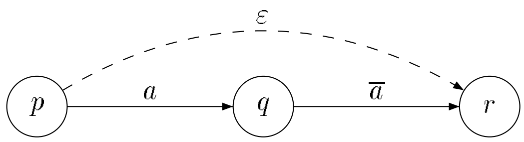
Corollaire 1.67 *Un sous-ensemble S du groupe libre est rationnel si et seulement si $D(S\pi^{-1})$ est reconnaissable par automate fini.*

Corollaire 1.68 *Si S est un sous-ensemble rationnel du groupe libre, alors $S \cap A^*$ est un langage rationnel du monoïde libre.*

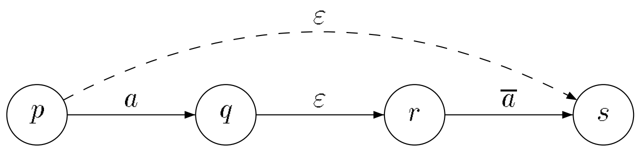
Soit L un langage rationnel de $\tilde{A}^*$. L'ensemble $D(L\pi\pi^{-1})$ des mots réduits équivalents aux mots de L peut se calculer en temps polynomial. Lorsque L est donné par un automate, décrivons intuitivement, par l'algorithme ci-dessous, comment calculer $D(L\pi\pi^{-1})$.

Algorithme 1.69

1. Chaque fois que l'on a dans l'automate reconnaissant L une configuration du type $p \cdot a = q$ et $q \cdot \overline{a} = r$, avec $a \in \tilde{A}$, on construit une ε-transition de p à r.



2. Chaque fois que l'on a une configuration du type $p \cdot a = q$ et $q \cdot \varepsilon = r$ et $r \cdot \overline{a} = s$, avec $a \in \tilde{A}$, on construit une ε-transition de p à s, puis on effectue la clôture transitive de toutes les ε-transitions. On itère ce processus jusqu'à ce qu'il n'y ait plus d'ε-transition à rajouter.



3. On élimine les ε-transitions.
4. On calcule l'intersection avec $D(\tilde{A}^*)$.

◇

Les étapes 1 et 2 construisent des ε-transitions entre des états joignables par un mot de $\tilde{A}^*$ dont l'image dans le groupe libre est 1 (en particulier si $p \cdot u\overline{u} = q$, alors on a construit une ε-transition entre p et q). A la fin de l'étape 3, on obtient un automate reconnaissant un langage L_0 contenant $L \cup D(L\pi\pi^{-1})$ et tel que $D(L_0) = D(L\pi\pi^{-1})$.

M. Benois et J. Sakarovitch ont montré dans [BS86] qu'en partant d'un automate sur $\tilde{A}$ à n états reconnaissant un langage L, on peut optimiser l'algorithme 1.69 ci-dessus pour qu'il s'exécute en temps $O(n^3)$ (la puissance 3 provient du fait que l'on est obligé d'effectuer une clôture transitive). On appellera `MotsRéduits` cet algorithme.

Il faut remarquer ici que, si le langage de départ est donné par un automate déterministe, il n'en est rien de l'automate obtenu à la fin de l'algorithme. De plus, si l'on se donne un automate $\mathcal{A}$ sur $\tilde{A}$ et que dans `MotsRéduits`$(\mathcal{A})$ on supprime toutes les transitions sur $\overline{A}$, on obtient un automate $\mathcal{A}_0$ sur A tel que $L(\mathcal{A}_0) = L(\mathcal{A})\pi \cap A^*$. Ce qui nous permet d'affirmer la proposition suivante.

Proposition 1.70 *Soit $\mathcal{A}$ un automate à n états sur $\tilde{A}$. On peut calculer en temps $O(n^3)$ un automate $\mathcal{A}_0$ sur A ayant n états et tel que*

$$L(\mathcal{A})\pi \cap A^* = L(\mathcal{A}_0)$$

On appelle `ProjectionA*` l'algorithme de la proposition 1.70.

De plus, M. Benois et J. Sakarovitch ont montré la proposition suivante qui nous sera utile par la suite.

Proposition 1.71 *Soit $\mathcal{A}$ un automate sur $\tilde{A}$ à n états. On peut calculer en temps $O(n^3)$ une matrice booléenne carrée de taille n telle que son coefficient d'indice (i, j) vaut 1 si et seulement s'il existe un mot $u \in 1\pi^{-1}$ qui est l'étiquette d'un chemin de l'état i à l'état j.*

On appelle `BenoisSakarovitch` l'algorithme de la proposition 1.71.

1.7.2 Sous-groupes rationnels

Tous les sous-groupes du groupe libre ne sont pas rationnels. S'il est aisé de voir que les sous-groupes finiment engendrés du groupe libre sont rationnels, la réciproque fut prouvée par A.W. Anissimov et F.D. Seifert [AS65]. Pour plus de détails sur l'algorithmique des sous-groupes rationnels du groupe libre, le lecteur est invité à se reporter à [Wei00]. Les résultats de cette section sont connus, sous d'autres formes, depuis le début du siècle. On doit à J. Stallings [Sta83] leur réécriture dans un vocabulaire de la théorie des graphes.

Théorème 1.72 *Un sous-groupe H du groupe libre est rationnel si et seulement s'il est finiment engendré.*

Nous allons décrire comment représenter les sous-groupes rationnels du groupe libre par des automates finis.

dual

Définition 1.73 *Un automate $\mathcal{A}$ sur l'alphabet $\tilde{A}$ est dit dual si, pour toute transition (p, a, q), le triplet $(q, \overline{a}, p)$ est aussi une transition.*

automate de sous-groupe

Définition 1.74 *Un automate de* sous-groupe *est un automate sur $\tilde{A}$, dual, déterministe, dont l'unique état final est aussi l'état initial.*

La dénomination “sous-groupe” se justifie par la proposition suivante.

Proposition 1.75 *Soit H un sous-groupe rationnel du groupe libre. Il existe un et un seul (à isomorphisme près) automate de sous-groupe, noté $\mathcal{A}_H$, ayant un nombre minimal d'états tel que $L(\mathcal{A}_H\pi) = H$.*

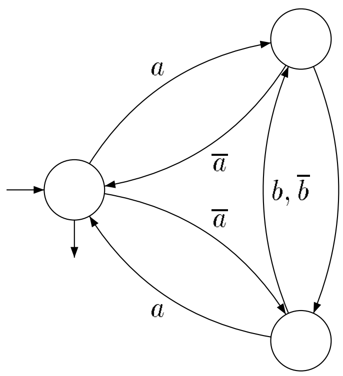
Réciproquement, on a

Proposition 1.76 *Tout automate de sous-groupe reconnaît un langage de $\tilde{A}^*$ dont l'image dans le groupe libre par π est un sous-groupe finiment engendré.*

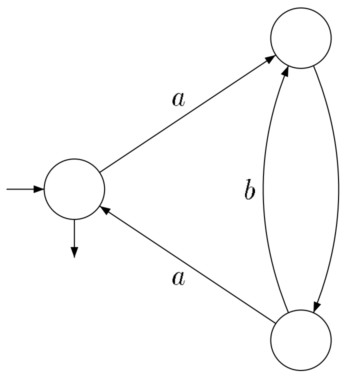
Pour plus de simplicité dans la représentation, lorsque l'on dessine un automate dual, on omet souvent de tracer les transitions étiquetées par des lettres de $\overline{A}$. On dit alors que l'on ne représente pas les transitions duales. D'ailleurs, si $\mathcal{A}$ est un automate de sous-groupe, on note par $\mathcal{A}^+$ l'automate obtenu à partir de $\mathcal{A}$ en enlevant toutes les transitions dont l'étiquette appartient à $\overline{A}$. transitions duales

Il existe un algorithme pour calculer l'automate de sous-groupe d'un sous-groupe rationnel du groupe libre. Avant de le décrire, considérons l'exemple suivant.

Exemple 1.77 Considérons le sous-groupe H de $F(\{a, b\})$ engendré par aba et $ab^2\overline{a}$. On peut montrer que son automate de sous-groupe $\mathcal{A}_H$ est le suivant :



Si on ne représente pas les transitions duales (c'est-à-dire étiquetées par les lettres $\overline{a}$ ou $\overline{b}$), on obtient l'automate $\mathcal{A}_H^+$ suivant :



$\triangledown$

Proposition 1.78 *Soit H un sous-groupe rationnel du groupe libre et $\mathcal{A}$ un automate sur l'alphabet $\tilde{A}$ à n états tel que $L(\mathcal{A})\pi = H$. On peut calculer en temps $O(n^2)$ un automate de sous-groupe $\mathcal{A}_0$ tel que $L(\mathcal{A}_0)\pi = H$. De plus, cet automate a moins de n états.*

On utilise pour cela l'algorithme `Réduction` suivant (qui s'applique à un automate $\mathcal{A}$ vérifiant $L(\mathcal{A})\pi = H$).

Algorithme 1.79

1. Construire l'automate $\mathcal{A}_0$ obtenu à partir de $\mathcal{A}$ en rajoutant toutes les transitions duales.
2. Fusionner en un seul état tous les états initiaux et finaux de $\mathcal{A}_0$.
3. Dans $\mathcal{A}_0$, tant qu'il y a une configuration du type $p \cdot a = q \cdot a$ ou du type $\{p, q\} \subseteq r \cdot a$ avec $a \in \tilde{A}$, fusionner les états p et q.
4. Retourner $\mathcal{A}_0$.

$\diamondsuit$

Dans l'algorithme précédent, l'étape 3 est itérée tant que l'on a des configurations d'un des deux types donnés. On peut montrer que l'ordre dans lequel on effectue ces fusions ne change pas le résultat final (à isomorphisme d'automate près).

Afin de prouver la complexité de l'algorithme `Réduction`, il faut donner quelques précisions sur la structure de données utilisée pour coder les automates. Généralement on utilise deux méthodes de codage. On peut coder un automate à l'aide de matrices, c'est-à-dire que l'on numérote les états de l'automate de 1 à n et que pour chaque lettre a on se donne une matrice booléenne dont le coefficient d'indice (i, j) vaut 1 si et seulement s'il y a une transition étiquetée par a entre i et j. Cette méthode a l'avantage de rendre les algorithmes assez simples mais est peu efficace lorsque l'on travaille exclusivement sur des automates déterministes (on code alors un nombre linéaire d'informations en un espace quadratique : les matrices contiennent de nombreux 0). L'autre méthode, héritée elle aussi du codage des graphes, utilise des listes d'adjacence. Pour chaque état p et pour chaque lettre a, on se donne la liste des états accessibles à partir de p en lisant a. Notons que pour un automate à n états, passer d'un codage à l'autre se fait en temps $O(n^2)$.

Dans le cas des automates déterministes complets, on obtient un codage très efficace en associant à chaque état un tableau de taille le cardinal de l'alphabet représentant les images de l'état par chacune des lettres.

Preuve. Pour l'algorithme ci-dessus on va utiliser une variante du codage par matrices. On suppose donc que l'on dispose d'un automate $\mathcal{A}$ à n états et m transitions sur l'alphabet A donné par $|A|$ matrices booléennes carrées de taille n. On note M_a la matrice associée à la lettre a. Analysons la complexité de l'algorithme étape par étape.

1. On ajoute à $\mathcal{A}$ toute ses transitions duales, c'est-à-dire que chaque fois que l'on a une transition (p, a, q), on ajoute la transition $(q, \overline{a}, p)$. La matrice booléenne associée à $\overline{a}$ est donc la transposée de la matrice associée à a. On peut ainsi construire les matrices des $\overline{a}$ en temps $O(n^2)$.

1. bis On va légèrement modifier la structure de donnée pour augmenter l'efficacité de l'algorithme. A partir de chaque matrice M_a, pour $a \in \tilde{A}$, on construit la matrice booléenne carrée N_a de taille $n + 1$ comme suit : le bloc de taille n en haut à gauche est M_a. Le coefficient d'indice $(n+1, j)$, pour $j \leq n$, vaut 1 si et seulement s'il y a au moins deux 1 dans la j-ième colonne de M_a. De même, le coefficient d'indice $(j, n + 1)$, pour $j \leq n$, vaut 1 si et seulement s'il y a au moins deux 1 dans la j-ième ligne de M_a. Le coefficient d'indice $(n+1, n+1)$, que nous n'utiliserons pas, vaut par défaut 0.

$$N_a = \left[\begin{array}{c|c} M_a & \\ \hline & 0 \end{array}\right]$$

Construire les matrices N_a se fait en parcourant une fois chaque ligne et chaque colonne des M_a, donc en temps $O(n^2)$. Avec cette structure de donnée, fusionner les états i et j s'effectue en temps $O(n)$. En effet, dans chaque matrice N_a, il suffit alors de mettre à 1 l'élément d'indice

(i,k) $(k \leq n)$ si et seulement si un des deux éléments d'indice (i,k) ou (j,k) vaut 1 (on effectue un "ou" logique des lignes i et j). On fait de même avec les colonnes i et j. On effectue ensuite une mise à jour des éléments d'indice $(i, n+1)$ et $(n+1, i)$. On supprime enfin la ligne j et la colonne j. Les nouvelles matrices ainsi obtenues (de taille n cette fois ci) codent l'automate où l'on a fusionné les états i et j. Pour ce faire, on effectue un travail en temps $O(n)$ dans chaque matrice N_a et donc le temps total est en $O(n)$ (on considère la taille de l'alphabet comme une constante).

2. On vient de voir que fusionner deux états se fait en temps $O(n)$ donc, comme on effectue au plus n fusions, cette étape peut se faire en temps $O(n^2)$.
3. Grâce à notre structure de données, trouver s'il y a deux états à fusionner s'effectue en temps $O(n)$. En effet, il suffit de parcourir les dernières lignes et colonnes des matrice N_a et de s'arrêter dès que l'on trouve un 1. Dans ce cas, on parcourt la colonne ou la ligne correspondante pour trouver les deux états à fusionner. Par conséquent, trouver s'il y a deux états à fusionner et les fusionner se fait en temps $O(n)$. Comme il y a au plus n fusions, cette étape s'effectue en temps $O(n^2)$.
4. Cette étape s'effectue en temps constant.

La complexité finale est donc de $O(n^2)$. □

La proposition suivante nous sera utile pour la suite.

Proposition 1.80 *Soit $\mathcal{A} = (Q, \tilde{A}, E, I, F)$ un automate émondé dual. Pour tout état $q \in Q$, on note $H_q = L(\mathcal{A}^q)\pi$. On a :*

- *H_q est un sous-groupe rationnel de $F(A)$.*
- *Si $p \in Q$ et $u \in \tilde{A}^*$ sont tels que $p \in q \cdot u$, alors $H_q = (u\pi)H_p(\overline{u}\pi)$.*
- *Pour tout état initial $i \in I$ et tout état final $f \in F$, on note u_{if} un mot tel que $f \in i \cdot u_{if}$. Alors*

$$L(\mathcal{A})\pi = \bigcup_{i\in I, f\in F} (u_{if}\pi)H_f = \bigcup_{i\in I, f\in F} H_i(u_{if}\pi)$$

En particulier, deux automates de sous-groupe ayant le même graphe reconnaissent des sous-groupes conjugués.

1.8 Langages rationnels et monoïdes finis

1.8.1 Langages reconnus par un monoïde fini

Les monoïdes finis permettent aussi de définir les langages rationnels. Ce résultat de M.O. Rabin et D. Scott [RS59] permit une approche algébrique pour l'étude des langages rationnels. Le premier résultat significatif de cette approche est certainement la caractérisation par des monoïdes finis des langages dits *sans-étoile* (langages pouvant s'exprimer à l'aide d'un nombre fini de symboles union, intersection, complémentaire et des ensembles finis) par M.-P. Schützenberger [Sch65]. Par la suite, d'autres classes de langages rationnels furent caractérisées par certaines classes de monoïdes finis [BS73, McN74, Sim90, PW97]. Le cadre théorique précis pour formuler ces résultats fut posé par S. Eilenberg [Eil74, Eil76] en définissant les notions de *variétés de langages* et de *variétés de monoïdes* (voir section 1.9).

Définition 1.81 *Soit M un monoïde fini et L un langage de A^*. On dit que M* reconnaît *L s'il existe un morphisme de monoïde φ de A^* dans M et une partie P de M telle que $L = P\varphi^{-1}$.*

Proposition 1.82 *Un langage L de A^* est reconnaissable par un automate fini si et seulement s'il est reconnaissable par un monoïde fini.*

On a donc égalité entre les classes des langages reconnaissables par un automate fini, par un monoïde fini et par une expression rationnelle.

Parmi tous les monoïdes finis qui reconnaissent un langage, il y en a un particulier (à isomorphisme près) que l'on peut associer au langage et que l'on appelle *monoïde syntaxique* du langage.

Proposition-Définition 1.83 *Soit L un langage rationnel. Soit $\sim_L$ la relation sur A^* définie par $u \sim_L v$ si et seulement pour tout couple de mots (w_1, w_2),* congruence syntaxique

$$w_1 u w_2 \in L \text{ si et seulement si } w_1 v w_2 \in L.$$

Cette relation est une congruence de monoïde d'indice fini appelée congruence syntaxique *de L. Le monoïde $A^*/\sim_L$ est fini et s'appelle* monoïde syntaxique *de L. On le note $M(L)$.* monoïde syntaxique

Proposition 1.84 *Soient L un langage rationnel, $P(L)$ l'ensemble des classes d'équivalence de $\sim_L$ qui contiennent un élément de L et φ_L la projection canonique de A^* dans $M(L)$. On a*

$$L = P(L)\varphi_L^{-1}$$

Le monoïde syntaxique d'un langage reconnaît donc ce langage.

Il existe une façon plus visuelle de construire un monoïde reconnaissant un langage rationnel et notamment le monoïde syntaxique. Si l'on considère un automate fini $\mathcal{A}$ déterministe complet, chaque mot induit une application de l'ensemble des états dans lui-même. L'ensemble des applications ainsi obtenues forme un monoïde fini pour le produit des applications ($\varphi\psi = \psi \circ \varphi$), appelé monoïde des transitions *monoïde des transitions* de $\mathcal{A}$, qui reconnaît $L(\mathcal{A})$ (la partie acceptant étant l'ensemble des applications qui envoient l'état initial sur un état final).

Proposition 1.85 *Le monoïde des transitions du complété de l'automate minimal d'un langage rationnel est son monoïde syntaxique.*

1.8.2 Langages à groupe

Nous définissons ici les *langages à groupe* qui seront étudiés plus en détail au chapitre 4.

langage à groupe

Définition 1.86 *Un langage rationnel est un* langage à groupe *si son monoïde syntaxique est un groupe.*

Les langages à groupe sont donc définis de façon algébrique. Cependant, on peut lire sur un automate minimal si le langage qu'il définit est à groupe.

Proposition 1.87 *Un langage de A^* est à groupe si et seulement si chaque lettre de A définit une permutation de l'ensemble des états de son automate minimal.*

On définit alors les *automates à groupe.*

automate à groupe

Définition 1.88 *Un automate sur l'alphabet A est* à groupe *si chaque lettre de A induit une permutation de l'ensemble de ses états.*

En particulier, un automate à groupe est complet. Cependant, il n'est pas nécessairement déterministe, car il peut posséder plusieurs états initiaux. En revanche, l'ensemble de ses transitions est déterministe.

On montre facilement la proposition suivante [Thi68].

Proposition 1.89 *Un langage est à groupe si et seulement s'il est reconnaissable par un automate à groupe, si et seulement si son automate minimal est à groupe.*

1.9 Sur la classification des langages rationnels

Lorsqu'une classe d'objets est souvent manipulée, il est utile de la classifier, c'est-à-dire de ranger ses objets, en fonction de leurs propriétés, dans différentes sous-classes, non nécessairement disjointes. Dans le cas des langages rationnels, l'expérience montra qu'une classification algébrique, c'est-à-dire relative aux propriétés des monoïdes syntaxiques, est pertinente. Il fut alors nécessaire de développer une classification des semigroupes finis qui préserve une cohérence au niveau des langages. Le formalisme précis et exact de cette classification est dû à S. Eilenberg [Eil74, Eil76] qui introduisit les concepts de variétés de langages et de monoïdes. On peut aussi citer les travaux antérieurs de M.-P Schützenberger [Sch65], J.A. Brzozowski et I. Simon [BS73], de I. Simon [Sim75] et aussi de R. McNaughton [McN74], qui ont grandement contribué à cette formalisation.

Le cadre de cette thèse n'est pas la théorie des variétés de langages. Cependant, nous nous intéresserons à des propriétés de certaines sous-classes de langages rationnels qui trouvent leur place dans cette théorie. Nous allons donc en définir les éléments de base afin d'éclairer notre travail. Nous nous contenterons de notions très simples, que nous ne discuterons pas et qui pourront être approfondies par la lecture de [Pin84, Eil74, Eil76] (pour les variétés) ou de [Arf87, Arf91, PW97] (pour les hiérarchies de langages).

1.9.1 Variétés de langages et de monoïdes finis

Commençons par définir les *variétés de monoïdes finis.*

Définitions 1.90 *Une* variété de monoïdes finis *est une classe de monoïdes finis stable par quotient, produit direct et passage au sous-monoïde. Si* $\mathbf{V}$ *est une variété de monoïdes et* A *un alphabet fini, on note* $A^*\mathbf{V}$ *l'ensemble des langages rationnels de* A^* *reconnaissables par un semigroupe de* $\mathbf{V}$.

variété de monoïdes

Les variétés de monoïdes finis sont aussi parfois appelées *pseudo-variétés de monoïdes* et sont souvent notées par des lettres majuscules grasses, telles que $\mathbf{V}$.

Définissons la notion de classe de langages rationnels.

Définition 1.91 *On appelle* classe de langages rationnels *toute application* $\mathcal{S}$*, qui à tout alphabet fini* A *associe un sous-ensemble* $A^*\mathcal{S}$ *des langages rationnels de* A^*.

classe de langages

On peut maintenant définir la notion de *variété de langages rationnels.*

Définition 1.92 *Une classe* $\underline{\mathcal{S}}$ *de langages rationnels est une* variété de langages *si elle vérifie les conditions suivantes :*

variété de langages

(i) Pour tout alphabet A*,* $A^*\mathcal{S}$ *est stable par les opérations booléennes.*

(ii) Si $L \in A^*\mathcal{S}$ *alors* $a^{-1}L$ *et* La^{-1} *sont aussi des éléments de* $A^*\mathcal{S}$.

(iii) Si φ *est un morphisme de monoïde libre de* A^* *dans* B^* *et si* $L \in B^*\mathcal{S}$*, alors* $L\varphi^{-1}$ *est un langage de* $A^*\mathcal{S}$.

Le lien entre les deux notions de variétés est donné par le théorème suivant [Eil74].

Théorème 1.93 *Une classe* $\mathcal{S}$ *de langages rationnels est une variété de langages si et seulement s'il existe une variété de monoïdes finis* $\mathbf{V}$ *telle que pour tout alphabet* A*,* $A^*\mathcal{S} = A^*\mathbf{V}$.

De très nombreux travaux ont été effectués dans ce domaine. Contentons nous de citer un résultat important [Pin95b] dans cette classification : en enrichissant la structure des monoïdes, on peut définir la notion de *variété de monoïdes finis ordonnés* qui peut être mise en correspondance avec les *variétés positives de langages rationnels*, c'est à dire les classes de langages rationnels vérifiant les conditions (*ii*) et (*iii*) de la définition 1.92 et fermées par union et intersection finies. On obtient ainsi une classification plus fine.

variété positive

Par abus de notation, si $\mathcal{S}$ est une classe de langages rationnels et qu'il n'y a pas d'ambiguïté sur l'alphabet A considéré, on dira qu'un langage L de A^* appartient à $\mathcal{S}$ si $L \in A^*\mathcal{S}$.

1.9.2 Langages sans-étoile

langages sans-étoile

La classe des langages sans-étoile joue un rôle important en théorie des automates. Ce sont les langages que l'on peut exprimer par une expression utilisant les lettres de l'alphabet et les symboles union, produit, intersection, complémentaire, $\emptyset$ et ε. C'est la première variété de langages concrètement étudiée [Sch65].

Pour définir la variété de monoïdes correspondante, définissons les monoïdes *apériodiques.*

apériodique

Définition 1.94 *Un monoïde fini est dit* apériodique *si et seulement si chacun de ses éléments x vérifie $x^\omega = x^\omega x$.*

Proposition 1.95 *Les monoïdes apériodiques forment une variété de monoïdes.*

Le résultat remarquable est le suivant [Sch65].

Théorème 1.96 *La classe des langages sans-étoile est une variété de langages dont la variété de monoïdes associée est la variété des monoïdes apériodiques.*

On ne peut pas parler des langages sans-étoile sans évoquer un des problèmes ouverts les plus importants en théorie des langages rationnels. On définit la hauteur d'étoile étendue d'un langage comme étant le nombre minimal d'étoiles imbriquées qu'il faut pour le représenter en utilisant les lettres de l'alphabet et les symboles union, produit, intersection, complémentaire, étoile et $\emptyset$. Les langages sans-étoile sont donc les langages de hauteur étendue 0. On sait qu'il existe des langages de hauteur étendue 1, c'est-à-dire qu'il existe des langages rationnels qui ne sont pas sans-étoile. On ne sait cependant pas s'il existe des langages de hauteur étendue 2.

1.9.3 Hiérarchies de concaténation

La notion de hiérarchie de concaténation a été introduite à l'origine pour mesurer la complexité des expressions rationnelles : plus un langage est de niveau faible dans la hiérarchie, plus il admet une représentation "simple" par une expression rationnelle. Il s'avère de plus que la notion de hiérarchie permet aussi d'ordonner algébriquement différentes variétés (positives) de langages rationnels. Avant de montrer comment on construit une hiérarchie de concaténation, définissons la notion de *clôture polynomiale.*

Définitions 1.97 *Soit A un alphabet et $\mathcal{L}$ un ensemble de langages rationnels de A^*. On définit la* clôture polynomiale *de $\mathcal{L}$, notée* Pol$\mathcal{L}$ *comme étant l'ensemble de tous les langages rationnels de A^* pouvant s'exprimer comme une union finie de langages du type*

clôture polynomiale

$$L_0 a_1 L_1 \cdots a_k L_k$$

où $L_i \in \mathcal{L}$ et $a_i \in A$. Par extension, si $\mathcal{S}$ est une variété de langages rationnels, on définit sa clôture polynomiale*, notée* Pol$\mathcal{S}$ *comme étant la classe de langages rationnels définie par*

$$A^*\mathrm{Pol}\mathcal{S} = \mathrm{Pol}(A^*\mathcal{S})$$

Une hiérarchie de concaténation est définie de demi-niveau en demi-niveau et est entièrement caractérisée par son niveau 0 (le niveau de départ), qui est une variété de langages. Les autres niveaux sont définis par récurrence en utilisant les règles suivantes :

hiérarchie

- Le niveau $n + 1/2$ est la clôture polynomiale du niveau n.
- Le niveau $n + 1$ est la clôture booléenne du niveau $n + 1/2$.

Les niveaux du type $n+1/2$ sont des variétés positives de langages ([Arf87, Arf91, PW97]) et les niveaux entiers sont des variétés de langages.

Nous aborderons dans cette thèse deux des trois hiérarchies de concaténation étudiées dans la littérature.

- La première hiérarchie résulte de travaux de H. Straubing [Str85] et D. Thérien [Thé81] et s'appelle la *hiérarchie de Straubing-Thérien.* Son niveau 0 est composé de l'ensemble vide et des langages du type A^*. Nous détaillerons plus cette hiérarchie au chapitre 5.
- La deuxième hiérarchie de langages est la *hiérarchie des langages à groupe*, qui est obtenue en prenant pour niveau 0 les langages à groupes. C'est la seule hiérarchie de langages non apériodiques étudiée dans la littérature. Elle sera détaillée au chapitre 4.

hiérarchie de Straubing-Thérien

hiérarchie des langages à groupe

Il est à noter que ces deux hiérarchies sont strictes [BK78], c'est à dire que, pour tout entier n, il existe des langages du niveau $n+1/2$ qui ne sont pas de niveau n et des langages du niveau $n+1$ qui ne sont pas de niveau $n + 1/2$.

La troisième hiérarchie (qui fut la première des hiérarchies considérées) est la hiérarchie *dot-depth* introduite par J.A. Brzozowski [Brz76]. Sa définition diffère légèrement des deux autres et ne repose pas sur la notion de variété de monoïde fini mais de *variété de semigroupes finis*, que nous ne définirons pas ici. Elle a fait l'objet de nombreuses études, notamment de la part de J.A. Brzozowski et R. Knast [BK78], qui montrèrent que c'est une hiérarchie stricte, de R. Knast [Kna83] qui s'intéressa aux propriétés du niveau 1 de cette hiérarchie et J. Stern qui montre dans [Ste85] comment décider efficacement le niveau 1. Terminons en citant les résultats récents de C. Glaßer et H. Schmitz [GS00] d'une part, et de J.-E. Pin et P. Weil [PW01b] d'autre part, qui montrent que le niveau 3/2 est décidable.

De même que l'on a défini la fermeture polynomiale, on peut définir la fermeture polynomiale non ambigüe (resp. déterministe)(resp. bidéterministe) en imposant aux produits utilisés d'être non ambigüs [PW97, Sch76, Pin81] (resp. déterministes [Pin81])(resp. bidéterministes [PT93, Bra97]). Pour chacune de ces constructions, on dispose de moyens algébriques pour décrire les variétés (éventuellement positives) de langages associés. Nous ne définirons pas davantage ces constructions ni ces définitions qui ne sont pas l'objet de cette thèse.

1.9.4 Variétés de groupes finis

Dans cette thèse, nous allons utiliser des variétés particulières de monoïdes finis : les *variétés de groupes finis*.

variété de groupes

Définition 1.98 *Une* variété de groupes finis *est une variété de monoïdes finis dont tous les éléments sont des groupes finis.*

Cette définition est justifiée par le fait que tout sous-monoïde d'un groupe fini est un sous-groupe de ce groupe et que tout morphisme de monoïde entre groupes est aussi un morphisme de groupe.

Les variétés de groupes finis sont souvent notées $\mathbf{H}$.

On peut, par exemple, considérer la variété $\mathbf{G}$ de tous les groupes finis, la variété $\mathbf{G}_{\mathrm{com}}$ de tous les groupes finis commutatifs : tout sous-groupe d'un groupe commutatif est commutatif, le produit de deux groupes commutatifs est aussi commutatif, de même que l'image par morphisme d'un groupe commutatif. D'autres variétés jouent un rôle important, comme la variété $\mathbf{G}_p$, où p est un nombre premier, qui est la classe de tous les p-groupes finis, c'est à dire tous les groupes dont le cardinal est une puissance de p ; ou encore les variétés $\mathbf{G}_{\mathrm{sol}}$ et $\mathbf{G}_{\mathrm{nil}}$ des groupes finis respectivement résolubles et nilpotents. Nous ne travaillerons pas sur ces deux dernières variétés que nous ne définirons donc pas plus ; pour plus de précisions, se reporter à [Rot95].

1.10 Décomposition unitaire-préfixielle

Nous allons décrire ici une décomposition particulière des langages rationnels due à S. Eilenberg [Eil74, chapitre IV].

1.10.1 Langages unitaires et langages préfixes

La décomposition qui va nous intéresser utilise des langages ayant de fortes propriétés, que nous développons ici.

Proposition-Définition 1.99 *Soit L un langage rationnel de A^*. Les assertions suivantes sont équivalentes :*

(i) L est reconnaissable par un automate déterministe ne possédant qu'un seul état final.

(ii) L'automate minimal de L ne possède qu'un seul état final.

(iii) Pour tout $u \in L$, $u^{-1}L = L^{-1}L$.

Dans ce cas, L est dit unitaire. unitaire

Pour illustrer cette définition, on peut rappeler la proposition suivante.

Proposition 1.100 *Tout langage est une union disjointe de langages unitaires.*

Preuve. Soit L un langage rationnel et $\mathcal{A} = (Q, A, E, i, F)$ son automate minimal. On a

$$L = \bigcup_{f \in F} L(\mathcal{A}^{i,f})$$

Par définition les $L(\mathcal{A}^{i,f})$ sont des langages unitaires. □

Un cas particulier des langages unitaires est obtenu lorsque l'on considère les langages reconnus par un automate déterministe possédant un unique état final confondu avec l'état initial. Ces langages sont alors des sous-monoïdes de A^*.

Proposition 1.101 *Soit L un langage rationnel de A^*. Les assertions suivantes sont équivalentes :*

(i) L est reconnaissable par un automate déterministe ne possédant qu'un seul état final qui est confondu avec l'état initial.

(ii) L'automate minimal de L ne possède qu'un seul état final qui est confondu avec l'état initial.

(iii) Pour tout $u \in L$, $u^{-1}L = L$.

(iv) L est unitaire et $\varepsilon \in A$.

(v) L est un monoïde unitaire.

Nous avons maintenant besoin de la notion de langage *préfixiel.*

Proposition-Définition 1.102 *Soit L un langage rationnel de A^*. Les assertions suivantes sont équivalentes :*

(i) L est reconnaissable par un automate déterministe ne possédant qu'un seul état final f et, de plus, $f \cdot A = \emptyset$.

(ii) L'automate minimal de L ne possède qu'un seul état final f et, de plus, $f \cdot A = \emptyset$.

(iii) Pour tout $u \in L$, $u^{-1}L = \{\varepsilon\}$.

(iv) $L^{-1}L = \{\varepsilon\}$.

préfixiel

Dans ce cas, L est dit préfixiel.

Définitions 1.103 *Un $(2k+1)$-uplet $(M_0, a_1, M_1, \cdots, a_k, M_k)$ où les M_i sont des monoïdes unitaires et où, pour tout $1 \leq j \leq k$,*

$$M_0 a_1 M_1 \ldots M_{j-1} a_j$$

monôme unitaire-préfixiel

est préfixiel, est appelé monôme unitaire-préfixiel. *Le langage $M_0 a_1 M_1 \ldots a_k M_k$ est appelé langage associé à $(M_0, a_1, M_1, \cdots, a_k, M_k)$*

1.10.2 Algorithme de décomposition unitaire-préfixielle

Rappelons que si L est un langage rationnel, $m(L)$ désigne le nombre d'états de son automate minimal.

Proposition 1.104 *Soit L un langage unitaire. On pose*

$$P = L \setminus \bigcup_{a \in A} LaA^*$$

ainsi que $M = P^{-1}L$ et, pour tout $a \in A$, $K_a = Pa^{-1}$. On a

$$L = \bigcup_{a \in A} K_a a M$$

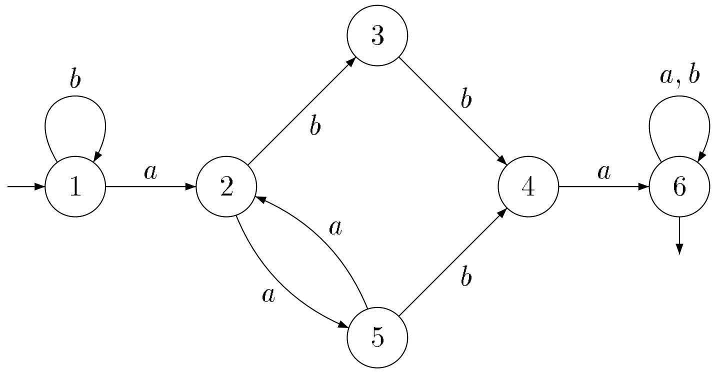
De plus, $m(K_a) < m(L)$, M est un monoïde unitaire et, pour tout $a \in A$, $K_a a M$ est préfixiel.

La décomposition ci-dessus s'obtient facilement à partir de l'automate minimal $\mathcal{A} = (Q, A, E, i, f)$ de L. En effet, M est le langage reconnu par l'automate (Q, A, E, f, f) et P le langage reconnu par

$$(Q, A, E \setminus \{(f, a, q) \mid a \in A, q \in Q\}, i, f)$$

c'est-à-dire l'automate déduit de $\mathcal{A}$ en supprimant toutes les transitions qui partent de f.

Exemple 1.105 Considérons le langage unitaire L donné par son automate minimal $\mathcal{A}$:

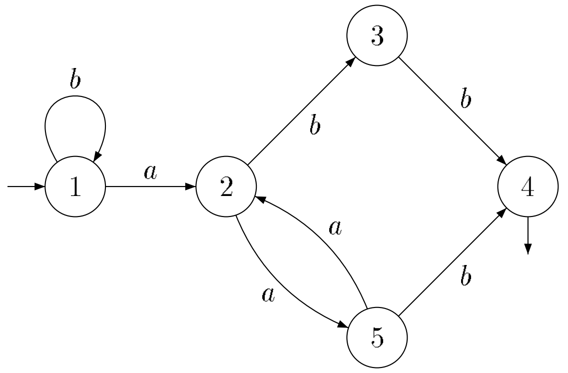


Le monoïde M est donc le langage reconnu par l'automate $\mathcal{A}$ dans lequel on prend pour état initial et final 6. Par conséquent, $M = A^*$.

Le langage P est reconnu par l'automate obtenu à partir de $\mathcal{A}$ en enlevant toutes les transitions qui partent de l'état 6, soit ici les transitions $(6, a, 6)$ et $(6, b, 6)$.

Le langage $K_b = Pb^{-1}$ est vide car aucun mot de P ne finit par la lettre b.

En revanche le langage K_a est reconnu par l'automate :



$\triangledown$

On déduit de la proposition 1.104 un algorithme de décomposition d'un automate déterministe émondé $\mathcal{A} = (Q, A, E, i, F)$. Il s'agit d'un algorithme Décompose qui applique récursivement la décomposition de la proposition 1.104.

Algorithme 1.106

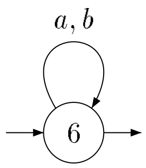
Si $L(\mathcal{A}) = \emptyset$ ne rien retourner et stopper l'algorithme.
Si $F = \{i\}$ et $|Q| = 1$ retourner $\mathcal{A}$.
Sinon, pour tout $f \in F$,

1. Calculer l'automate $\mathcal{A}_M$ obtenu en émondant (Q, A, E, f, f).
2. $E = E \setminus \{(f, a, q) \mid a \in A, q \in Q\}$.
3. Pour tout $a \in A$,
   (a) $F = \{p \mid (p, a, f) \in E\}$.
   (b) Emonder $\mathcal{A}$.
   (c) Retourner (`Décompose`$(\mathcal{A}), a, \mathcal{A}_M)$.

$\diamondsuit$

Exemple 1.107 Regardons ce que donne l'algorithme sur l'automate de l'exemple 1.105.

Le premier automate à calculer est l'automate que l'on obtient en émondant $(Q, A, E, 6, 6)$ (étape 1). Notons $\mathcal{A}_1$ cet automate :

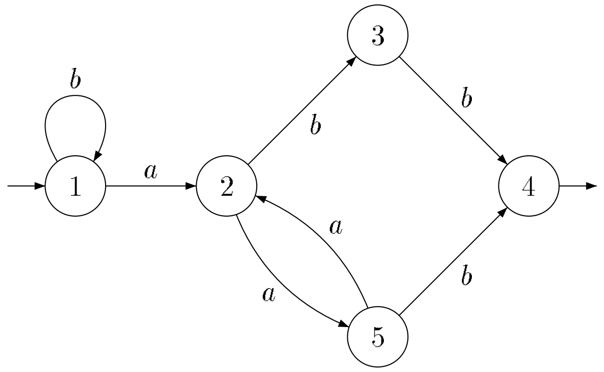


On enlève dans E les transitions qui partent de l'état 6 (étape 2). Il y a une bifurcation pour l'algorithme : il faut faire un calcul pour la lettre b et un pour la lettre a.

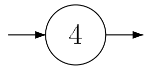
Pour la lettre b, on obtient à l'étape 3.(a) un ensemble d'états finaux vide. Par conséquent, cet automate reconnaît le vide et l'algorithme, pour cette branche, s'arrête sans rien retourner.

Pour la lettre a, on doit maintenant appliquer l'algorithme à l'automate

$\mathcal{A}_2$ suivant :

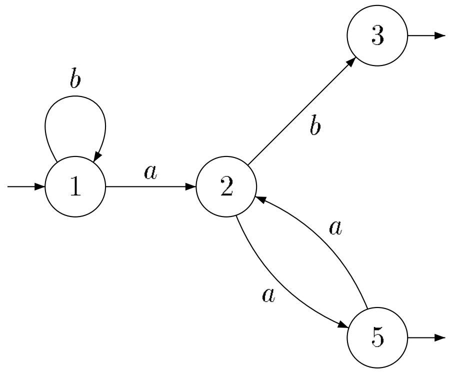


On calcule l'automate obtenu à partir de $\mathcal{A}_2$ en prenant comme état initial et final l'état 4. On obtient alors, après avoir émondé (étape 1), l'automate $\mathcal{A}_3$ suivant :

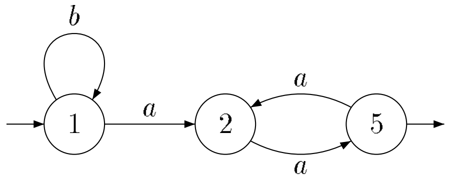


De même que ci-dessus, on ne doit maintenant examiner que le cas de la lettre b (car la lettre a conduit à un langage vide).

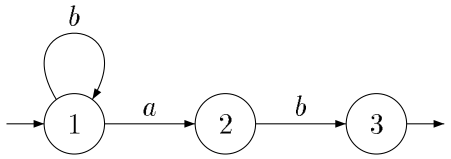
Il faut donc appliquer l'algorithme à l'automate $\mathcal{A}_4$ suivant :



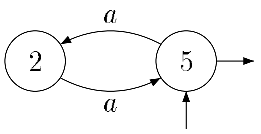
Comme il y a deux états finaux, il faut appliquer deux fois l'algorithme. D'abord à l'automate $\mathcal{A}_5$ suivant :



Et ensuite à l'automate $\mathcal{A}_6$ suivant :

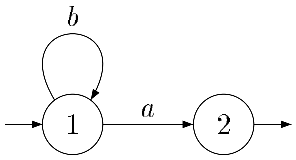


Commençons par appliquer l'algorithme à l'automate $\mathcal{A}_5$. On calcule alors l'automate $\mathcal{A}_7$ suivant (obtenu en prenant dans $\mathcal{A}_5$ pour état final l'état 5 et en émondant).

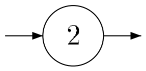


On reconsidère ensuite l'automate $\mathcal{A}_5$ dans lequel on supprime les transitions qui partent de l'état 5. Comme précédemment, le calcul ne se poursuit ici que pour la lettre a et on doit alors appliquer l'algorithme à l'automate $\mathcal{A}_8$

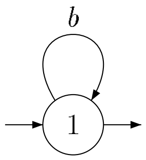
suivant :



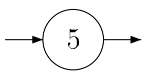
En prenant pour état final dans $\mathcal{A}_8$ l'état 2 et en émondant, on obtient l'automate $\mathcal{A}_9$ suivant :



Encore une fois le calcul ne se poursuit qu'avec la lettre a et il faut maintenant appliquer l'algorithme à l'automate $\mathcal{A}_{10}$ suivant :



Comme $\mathcal{A}_{10}$ ne possède qu'un seul état, le calcul s'arrête ici pour l'automate $\mathcal{A}_5$. Il faut maintenant appliquer l'algorithme à l'automate $\mathcal{A}_6$ pour finir. Le premier automate que l'on calcule alors est l'automate $\mathcal{A}_{12}$ suivant :



Ensuite, pour les même raisons que précédemment, il ne faut considérer que le calcul pour la lettre b et l'on retombe sur l'automate $\mathcal{A}_8$. Le calcul se termine alors de la même façon.

Il en résulte que l'algorithme retourne les deux 9-uplets suivants :

$$(\mathcal{A}_{10}, a, \mathcal{A}_9, a, \mathcal{A}_7, b, \mathcal{A}_3, a, \mathcal{A}_1)$$

et

$$(\mathcal{A}_{10}, a, \mathcal{A}_9, b, \mathcal{A}_{12}, b, \mathcal{A}_3, a, \mathcal{A}_1)$$

$\triangledown$

Proposition-Définition 1.108 *Soit $\mathcal{A}$ un automate déterministe. L'algorithme* Décompose*, appliqué à $\mathcal{A}$, retourne une liste de $(2k+1)$-uplets du type $(\mathcal{A}_0, a_1, \mathcal{A}_1, \cdots, a_k, \mathcal{A}_k)$ où les $\mathcal{A}_i$ sont des automates dont le graphe est un sous-graphe d'une composante fortement connexe de $\mathcal{A}$ et les a_i des lettres. Pour tout $(2k+1)$-uplet obtenu, le $(2k+1)$-uplet*

$$(L(\mathcal{A}_0), a_1, L(\mathcal{A}_1), \cdots, a_k, L(\mathcal{A}_k))$$

est un monôme unitaire préfixiel. De plus, $L(\mathcal{A})$ est l'union des langages associés à ces monômes unitaires préfixiels. La décomposition de $L(\mathcal{A})$ ainsi obtenue s'appelle décomposition unitaire préfixielle *de $\mathcal{A}$. Par extension, on appelle* décomposition unitaire préfixielle *d'un langage L la décomposition unitaire préfixielle de son automate minimal.*

décomposition unitaire-préfixielle

Chapitre 2

Topologie et parties rationnelles

Sommaire

2.1 Introduction et motivations 58
2.2 Topologies pro-H 59
2.2.1 Distances pro-**H** 59
2.2.2 Topologies pro-**H** 60
2.2.3 Sous-groupes rationnels ouverts-fermés 62
2.2.4 Topologies sur le monoïde libre 63
2.2.5 Propriétés des langages ouverts et des langages fermés 63
2.3 Calcul de la clôture d'un sous-groupe rationnel . 65
2.3.1 Généralités 65
2.3.2 Théorème de Hall 65
2.3.3 Calcul de la fermeture pro-p d'un sous-groupe rationnel 66
2.4 Calcul de la clôture d'une partie rationnelle . . . 70
2.4.1 Algorithme de Pin-Reutenauer 70
2.4.2 Topologie profinie et parties rationnelles 73
2.4.3 Clôture d'une partie rationnelle donnée par un automate 73
2.4.4 Calcul dans le cas profini 89
2.5 Calcul de la clôture d'un langage rationnel 90
2.5.1 Clôture d'un langage rationnel donné par un automate 90
2.5.2 Calcul dans le cas profini 91
2.5.3 Calcul dans le cas le pire 96
2.6 Propriétés topologiques des parties rationnelles . 100
2.6.1 Langages denses 100
2.6.2 Points d'accumulation 102
2.6.3 Fermeture d'une partie positive 108
2.6.4 Langages non rationnels et topologie profinie 111

2.7 Conclusion 112

2.1 Introduction et motivations

La topologie profinie, introduite pour le groupe libre par M. Hall [Hal50] en 1950, est définie algébriquement à l'aide des groupes finis. Dans son article fondateur, M. Hall prouve que les sous-groupes finiment engendrés (rationnels) du groupe libre sont fermés pour la topologie profinie, montrant déjà ainsi implicitement (la notion de partie rationnelle n'était pas encore connue) que des liens étroits unissent cette topologie et les parties rationnelles ; ce que des études postérieures ont confirmé.

En 1979, C. Reutenauer [Reu79, Reu81] montra que, d'une manière algébrique similaire, on peut construire la topologie profinie du monoïde libre et que, de plus, celle-ci est la trace de la topologie profinie du groupe libre sur le monoïde libre.

En ce qui concerne la théorie des langages formels, historiquement, la topologie profinie fut d'abord utilisée non pas pour l'étude des langages rationnels, mais pour l'étude des semigroupes finis. Une formulation topologique d'une conjecture de J. Rhodes, dite conjecture de type II [Rho87], relative à des problèmes de décidabilité de certaines classes de semigroupes finis, fut proposée par J.-E. Pin [Pin88] en ces termes : peut-on calculer la fermeture profinie d'une partie rationnelle du groupe libre? (pour plus de détails sur ce sujet, voir section 2.4.1 et chapitre 6). Un algorithme pour répondre à cette question fut conjecturé par J.-E. Pin et C. Reutenauer [PR91], initialisant ainsi l'étude topologique des langages rationnels. La conjecture de Rhodes fut prouvée, en utilisant la théorie des semigroupes, par C.J. Ash [Ash91], alors que la correction de l'algorithme fut démontrée deux ans plus tard par L. Ribes et P.A. Zalesskii [RZ93].

En 1987, J.-E. Pin montra les liens entre la topologie profinie et les langages reconnaissables par automates réversibles [Pin92] (voir chapitre 3), puis entre la hiérarchie des langages à groupe et la topologie profinie [Pin94] (voir chapitre 4).

Ces études, pour la topologie profinie, peuvent se généraliser à d'autres topologies définies d'une façon similaire : les topologies dites pro-**H**. Elles sont, elles aussi, liées à des problèmes de décision en théorie des semigroupes finis [HMPR91, Del98] (voir chapitre 6) ou avec des hiérarchies de concaténation [Ste00d].

Dans la section 2.2, nous définissons les topologies pro-**H** et esquissons leurs liens avec les parties rationnelles. Dans la section 2.4, nous abordons le problème initial du calcul de la fermeture pro-**H** d'un langage rationnel et montrons comment l'algorithme développé par J.-E. Pin et C. Reutenauer peut s'adapter pour fournir un algorithme efficace du calcul de la fermeture d'une partie rationnelle du groupe libre ou du monoïde libre donnée par un automate fini. Pour finir, dans la section 2.6, nous étudions les propriétés topologiques des langages rationnels en développant plus particulièrement les propriétés des langages fermés, denses et des points d'accumulation des langages rationnels.

2.2 Topologies pro-H

2.2.1 Distances pro-H

On va maintenant définir les *topologies pro-***H** qui permettent de comparer des éléments du groupe libre, relativement à une variété de groupes finis **H** dans le sens suivant : deux éléments x et y du groupe libre $F(A)$ sont proches pour la topologie pro-**H** s'ils ont des comportements similaires dans les groupes de **H**. Plus précisément, on va évaluer x et y dans tous les groupes de **H** qui sont A-engendrés et regarder si ces évaluations sont souvent égales. On introduit alors la notion suivante :

Définition 2.1 *On dit qu'un groupe G* sépare *deux éléments x et y du groupe libre $F(A)$ s'il existe un morphisme de groupe de $F(A)$ dans G par lequel x et y ont des images différentes.* sépare

Plus il y aura de groupes de **H** séparant x et y, plus on estimera qu'ils sont éloignés. Comme il y a une infinité de groupes séparant x et y, on ne peut pas se contenter de les compter. On introduit alors les deux quantités suivantes :

Définition 2.2 *Soit* **H** *une variété de groupes finis et x et y deux éléments de $F(A)$. On pose*

$$r_{\mathbf{H}}(x,y) = \min\{|G| \mid G \in \mathbf{H} \text{ et } G \text{ sépare } x \text{ et } y\}$$

et

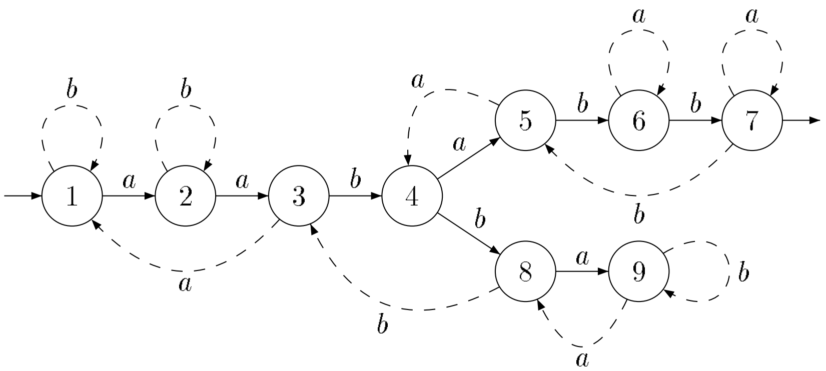
$$d_{\mathbf{H}}(x,y) = 2^{-r_{\mathbf{H}}(x,y)}$$

avec les conventions $\min \emptyset = +\infty$ *et* $2^{-\infty} = 0$.

Ainsi, plus $d_{\mathbf{H}}(x,y)$ est petit, plus le plus petit groupe de **H** séparant x et y est de cardinal important (et réciproquement). C'est donc ce réel $d_{\mathbf{H}}(x,y)$ qui permet d'évaluer, relativement à **H**, la proximité de x et y.

Selon la variété **H**, on ne peut pas forcément séparer deux éléments distincts x et y. Par exemple, ab et ba ne peuvent pas être séparés par un groupe commutatif, et donc $d_{\mathbf{G}_{\mathrm{com}}}(ab,ba) = 0$. En revanche, ab et ba peuvent être séparés par n'importe quel groupe non commutatif (en envoyant a et b sur deux éléments qui ne commutent pas), et, par conséquent, $d_{\mathbf{G}}(ab,ba) = 2^{-6}$, car le plus petit groupe non commutatif est de cardinal 6.

D'une façon générale, deux éléments x et y peuvent toujours être séparés par un groupe fini, comme le montre l'exemple suivant (qui peut facilement se généraliser), pour *aababb* et *aabba*.



On construit un automate déterministe (dans lequel les lettres induisent des fonctions injectives de l'ensemble des états dans lui-même) reconnaissant le mot *aababb* mais pas le mot *aabba* et on le complète afin que les lettres a et b définissent chacune une permutation de l'ensemble des états. On peut donc considérer le sous-groupe G des permutations d'un ensemble à 9 éléments engendré par a et b. Dans ce groupe, les permutations *aababb* et *aabba* sont distinctes car l'image de 1 par *aababb* est 7 et celle de 1 par *aabba* est 9. Il en résulte que G sépare *aababb* et *aabba*.

Cet exemple montre que deux éléments distincts x et y du groupe libre, représentés par des mots réduits de tailles respectives n et m, sont séparés par un groupe de cardinal au plus $(n+m)!$ et donc que $d_{\mathbf{G}}(x, y) \geq 2^{-(n+m)!}$. On peut obtenir une meilleure majoration [Rob96] ; cependant, on ne connaît pas de façon de calculer $d_{\mathbf{G}}(x, y)$ autre que d'énumérer les uns à la suite des autres tous les groupes finis. Ce n'est certes pas efficace mais cela ne représente pas un réel problème car, comme on va le voir, l'existence et les propriétés intrinsèques de la fonction $d_{\mathbf{H}}$ permettent d'obtenir des résultats sans que l'on ait à la calculer explicitement.

2.2.2 Topologies pro-H

La notion mathématique permettant de traiter les problèmes de proximité ou d'éloignement est la topologie. Grâce à la fonction $d_{\mathbf{H}}$, on peut définir une topologie sur le groupe libre dite topologie pro-**H**.

Proposition 2.3 *La fonction $d_{\mathbf{H}}$ définie une pseudo-distance sur le groupe* *libre $F(A)$, c'est-à-dire que :*

- *Quel que soit x dans $F(A)$, $d_{\mathbf{H}}(x,x) = 0$.*
- *Quels que soient x et y dans $F(A)$, $d_{\mathbf{H}}(x,y) = d_{\mathbf{H}}(y,x) \geq 0$.*
- *Quels que soient x, y et z dans $F(A)$, $d_{\mathbf{H}}(x,y) \leq d_{\mathbf{H}}(x,z) + d_{\mathbf{H}}(z,y)$.*

Remarquons que la fonction $d_{\mathbf{H}}$ est un distance utltramétrique, c'est-à-dire qu'elle vérifie, pour tous x, y et z dans $F(A)$,

$$d_{\mathbf{H}}(x,y) \leq \max(d_{\mathbf{H}}(x,z), d_{\mathbf{H}}(z,y))$$

On peut maintenant définir la topologie pro-**H**.

Définition 2.4 *On appelle* topologie pro-**H** *la topologie induite par $d_{\mathbf{H}}$.* topologie pro-**H**

Notons que dans le cas de la variété $\mathbf{G}_p$, on parle de topologie pro-p au lieu de topologie pro-$\mathbf{G}_p$. pro-p

De même dans le cas de la variété **G**, on parle de topologie profinie au lieu de topologie pro-**G**. profinie

Une propriété caractéristique [Hal50] de la topologie pro-**H** (qui est une autre façon de la définir) est la suivante :

Proposition 2.5 *La topologie pro-**H** est la topologie la moins fine qui rend continus tous les morphismes du groupe libre vers les groupes de **H**.*

L'ensemble des images réciproques de parties de groupes de **H** par un morphisme, qui ne sont autres que les parties du groupe libre reconnaissables par un groupe de **H**, forment donc une base d'ouverts de la topologie pro-**H**. De plus, la multiplication et l'inversion sont des applications continues pour la topologie pro-**H**, ce qui s'énonce :

Proposition 2.6 *Le groupe libre, muni de la topologie pro-**H**, est un groupe topologique.*

Notons que dans le cas présent on a des propriétés plus fortes : la multiplication est une application uniformément continue ainsi que les morphismes du groupe libre vers les groupes finis (pour la topologie pro-**H**).

La propriété de convergence suivante [Reu79] s'avère très utile pour l'étude des propriétés topologiques des parties rationnelles.

Proposition 2.7 *Soit x un élément du groupe libre $F(A)$. Pour toute topologie pro-**H**, la suite $(x^{n!})_{n\in\mathbb{N}}$ converge vers 1.*

Preuve. Soit x un élément de $F(A)$. Pour tout groupe G de **H** de cardinal inférieur à n et pour tout morphisme φ de $F(A)$ dans G, $(x\varphi)^{n!} = 1_G$ car $|G|$ divise $n!$. Il en résulte que $r_{\mathbf{H}}(x^{n!},1) \geq n$ et donc $d_{\mathbf{H}}(x^{n!},1) \leq 2^{-n}$. □

Un cas particulièrement intéressant est le cas où la fonction $d_{\mathbf{H}}$ définit une distance, c'est à dire le cas où deux éléments du groupe libre peuvent toujours être séparés. Une condition suffisante, mais assez générale, pour qu'il en soit ainsi, est le cas des variétés closes par extension.

variété close par extension

Définition 2.8 *Une variété* $\mathbf{H}$ *de groupe finis est dite* close par extension *si pour tout groupe fini* G *et tout sous-groupe normal* H *de* G *tels que* G/H *et* H *soient dans* $\mathbf{H}$*,* G *est dans* $\mathbf{H}$*.*

C'est le cas des variétés $\mathbf{G}$, $\mathbf{G}_p$ ou $\mathbf{G}_{\mathrm{sol}}$, mais pas de $\mathbf{G}_{\mathrm{nil}}$ ni de $\mathbf{G}_{\mathrm{com}}$. Le résultat [Hal50] qui nous intéresse est le suivant :

Proposition 2.9 *Soit* $\mathbf{H}$ *une variété close par extension. La fonction* $d_{\mathbf{H}}$ *est une distance.*

La proposition ci-dessus donne une condition suffisante pour que $d_{\mathbf{H}}$ soit une distance. Cette condition n'est pas nécessaire (par exemple $d_{\mathbf{G}_{\mathrm{nil}}}$ est une distance), mais si nous nous intéressons aux variétés closes par extension, c'est parce qu'elles définissent des topologies possédant de bonnes propriétés vis-à-vis des parties rationnelles.

2.2.3 Sous-groupes rationnels ouverts-fermés

Par ailleurs, les sous-groupes d'indice fini du groupe libre, qui ne sont autres que les sous-groupes rationnels du groupe libre dont l'automate de sous-groupe est à groupe, ont des propriétés topologiques remarquables.

Proposition 2.10 *Un sous-groupe* H *du groupe libre est d'indice fini si et seulement s'il est rationnel et si* $\mathcal{A}_H$ *est un automate à groupe. De plus, lorsque* H *est d'indice fini, le nombre d'états de* $\mathcal{A}_H$ *correspond à l'indice de* H *dans* $F(A)$*.*

Le résultat suivant est prouvé dans [Hal50].

Proposition 2.11 *Soit* $\mathbf{H}$ *une variété de groupes finis. Les sous-groupes rationnels d'indice fini* H *de* $F(A)$ *tels que* $M(L(\mathcal{A}_H)) \in \mathbf{H}$ *forment une base de voisinages ouverts-fermés de* 1 *pour la topologie pro-*$\mathbf{H}$*.*

Comme dans un groupe topologique les translations sont des homéomorphismes, on obtient une base de voisinage de n'importe quel point par translation d'une base de voisinages de l'unité.

Proposition 2.12 *Soit* $\mathbf{H}$ *une variété de groupes finis. Les parties de* $F(A)$ *du type* Hx*, où* $x \in F(A)$ *et* H *est un sous-groupe d'indice fini de* $F(A)$ *tel que* $M(L(\mathcal{A}_H)) \in \mathbf{H}$ *, forment une base d'ouverts-fermés pour la topologie pro-*$\mathbf{H}$*.*

2.2.4 Topologies sur le monoïde libre

Le monoïde libre étant un sous-monoïde du groupe libre, on peut considérer la trace de la topologie pro-**H** du groupe libre sur le monoïde libre et définir ainsi la topologie pro-**H** sur le monoïde libre.

Définition 2.13 *La topologie* pro-**H** de A^* *est la restriction de la topologie pro-***H** *de* $F(A)$ *à* A^*.

Le point important est que les propriétés algébriques des topologies pro-**H** du groupe libre sont préservées par passage au monoïde libre.

Proposition 2.14 *La multiplication sur* A^* *est uniformément continue pour la topologie pro-***H***.*

De plus,

Proposition 2.15 *La topologie pro-***H** *du monoïde libre est la topologie la moins fine qui rend continue tous les morphismes du monoïde libre vers les groupes de* **H**.

2.2.5 Propriétés des langages ouverts et des langages fermés

La proposition suivante montre de quelle manière la topologie pro-**H** de A^* est décrite par des langages à groupe, même lorsque **H** n'est pas close par extension.

Proposition 2.16 *Une base d'ouverts de la topologie pro-***H** *de* A^* *est constituée des langages rationnels à groupe dont le monoïde syntaxique appartient à* **H**.

Il en résulte que tout langage ouvert pour la topologie pro-**H** de A^* est une union (éventuellement infinie) de langages à groupe.

Les langages à groupe dont le monoïde syntaxique appartient à **H** jouent un rôle particulier, comme cela est montré dans [Pin94, Ste00d], qui se résume par le théorème suivant :

Théorème 2.17 *Soient* **H** *une variété de groupes finis fermée par extension et* L *un langage rationnel de* A^*.

(1) L *est à la fois ouvert et fermé pour la topologie pro-***H** *si et seulement si* L *est un langage à groupe dont le monoïde syntaxique est dans* **H**.

(2) Si L *est ouvert, alors* $\mathrm{Cl}^{\mathbf{H}}_{A^*}(L)$ *est à la fois ouvert et fermé.*

(3) Si K *est un rationnel ouvert disjoint de* L *et si* L *est ouvert, alors il existe un langage à groupe* R *dont le monoïde syntaxique est dans* **H** *tel que*

$$L \subseteq R \subseteq K^c$$

Ce théorème sera discuté de façon plus détaillée, pour la topologie profinie, au chapitre 4.

Proposition 2.18 *Les boules ouvertes, pour la topologie pro-$\mathbf{H}$ de A^*, sont des langages rationnels ouverts-fermés.*

PREUVE. Il suffit de remarquer que la boule ouverte de centre $u \in A^*$ et rayon r est l'ensemble des $v \in A^*$ tels que, pour tout groupe G de $\mathbf{H}$ de cardinal inférieur à $\log_2 r$ et pour tout morphisme φ de A^* dans G, $v \in u\varphi\varphi^{-1}$. Or, comme $u\varphi\varphi^{-1}$ est un langage à groupe dont le monoïde syntaxique est dans $\mathbf{H}$, $B(u, r)$ est une union finie de tels langages. Mais, comme une union finie de langages à groupe dont les monoïdes syntaxiques sont dans $\mathbf{H}$ est un langage à groupe dont le monoïde syntaxique est dans $\mathbf{H}$, toute boule ouverte est un rationnel ouvert-fermé. □

La proposition suivante est montrée dans [Pin91].

Proposition 2.19 *Soit $\mathbf{H}$ une variété de groupes finis. Soient K et L deux langages rationnels de A^*. Si L est ouvert pour la topologie pro-$\mathbf{H}$, alors $K^{-1}L$ et LK^{-1} le sont aussi.*

L'aspect algébrique de la topologie profinie sur A^* permet de donner une caractérisation syntaxique des langages rationnels ouverts et des langages rationnels fermés [Pin94].

Proposition 2.20 *Un langage rationnel L est fermé pour la topologie profinie du monoïde libre si et seulement s'il vérifie l'une des propriétés équivalentes suivantes :*

(1) Pour tous $u, v, w \in A^*$, si $uv^+w \subseteq L$, alors $uw \in L$.

(2) Pour tous $s, t \in M(L)$ et tout idempotent e de $M(L)$, si $set \in P(L)$, alors $st \in P(L)$.

Dualement, on a

Proposition 2.21 *Un langage rationnel L est ouvert pour la topologie profinie du monoïde libre si et seulement s'il vérifie l'une de propriétés équivalentes suivantes :*

(1) Pour tous $u, v, w \in A^*$, si $uw \in L$, alors $uv^+w \cap L \neq \emptyset$.

(2) Pour tous $s, t \in M(L)$ et tout idempotent e de $M(L)$, si $st \in P(L)$, alors $set \in P(L)$.

Dans un cadre plus général, des caractérisations syntaxiques des langages ouverts ou fermés existent pour d'autres topologies pro-$\mathbf{H}$. Cependant, elles nécessitent l'introduction de plusieurs notions de théorie des monoïdes finis que nous ne développerons pas ici. Pour de plus amples informations sur le sujet voir [Pin94, Ste00d].

Le graphe de l'automate minimal d'un langage fermé a une géométrie particulière, comme le montre la propriété suivante qui sera prouvée au chapitre 3.

Proposition 2.22 *Soient* $\mathbf{H}$ *une variété de groupes finis close par extension et* L *un langage rationnel de* A^* *fermé ou ouvert pour la topologie pro-*$\mathbf{H}$. *L'automate minimal* $\mathcal{A}$ *de* L *vérifie la propriété suivante : si* p, q, r *sont trois états d'une même composante fortement connexe de* $\mathcal{A}$ *tels que* $p \cdot a = q \cdot a = r$, *alors* $p = q$.

PREUVE. La topologie profinie étant plus fine que toutes les autres topologies pro-$\mathbf{H}$, il suffit de prouver le résultat lorsque $\mathbf{H} = \mathbf{G}$, ce que nous ferons au chapitre 3, proposition 3.19. □

La réciproque est fausse : l'automate minimal de $L = a^+$ vérifie la propriété précédente, mais ce n'est pas un langage fermé profini car $\varepsilon \in L$, d'après la proposition 2.7.

2.3 Calcul de la clôture d'un sous-groupe rationnel

2.3.1 Généralités

Dans un groupe topologique, la fermeture d'un sous-groupe est toujours un sous-groupe. Cependant, la fermeture d'un sous-groupe finiment engendré n'est pas nécessairement finiment engendrée ; c'est aussi le cas, en général, pour les topologies pro-$\mathbf{H}$: la fermeture d'un sous-groupe rationnel peut être un sous-groupe non rationnel. Cependant, dans le cas des variétés closes par extension, on a le résultat suivant [MSW98].

Proposition 2.23 *Soient* $\mathbf{H}$ *une variété de groupes finis close par extension et* H *un sous-groupe rationnel de* $F(A)$. *La fermeture de* H, *pour la topologie pro-*$\mathbf{H}$, *est un sous-groupe rationnel* K *de* $F(A)$. *De plus* $\mathcal{A}_K$ *possède au plus autant d'états que* $\mathcal{A}_H$.

En fait le résultat montré dans [RZ94] et retraduit dans [MSW98] est plus fort et dit que, sous les hypothèses ci-dessus, l'automate $\mathcal{A}_K$ est un quotient de $\mathcal{A}_H$ par une congruence sur ses états : on peut passer de $\mathcal{A}_H$ à $\mathcal{A}_K$ par des fusions d'états.

2.3.2 Théorème de Hall

Dans son article [Hal50], M. Hall montre une propriété des sous-groupes rationnels du groupe libre.

Théorème 2.24 *Tout sous-groupe rationnel du groupe libre est fermé pour la topologie profinie.*

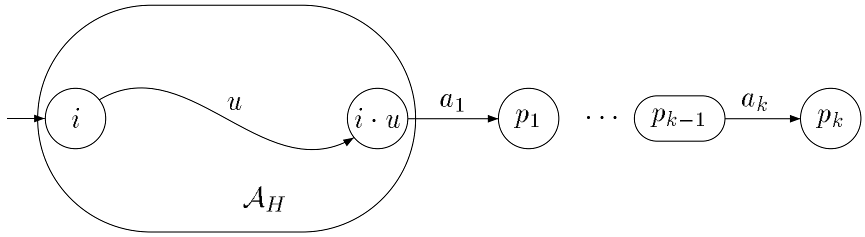
La preuve suivante se trouve dans [HL00] et a été très légèrement modifiée pour faire apparaître des automates et souligner le lien avec la rationalité.

Preuve. Soit H un sous-groupe rationnel (donc finiment engendré) de $F(A)$. D'après la proposition 2.11, les sous-groupes d'indice fini du groupe libre sont fermés pour la topologie profinie. Donc le sous-groupe H' défini par

$$H' = \bigcap_{H \subseteq K,\ K \text{d'indice fini}} K$$

est fermé et contient H. Montrons que $H' \subseteq H$, ce qui prouvera que H est fermé.

Soit $g \in F(A)$ tel que $g \notin H$. On va construire un sous-groupe normal contenant H et ne contenant pas g. L'automate de sous-groupe $\mathcal{A}_H = (Q, A, E, i, i)$ de H est déterministe, mais pas nécessairement complet. Soit v le mot réduit de $\tilde{A}^*$ tel que $v\pi = g$ et u le plus long préfixe de v que l'on peut lire dans $\mathcal{A}_H$. Soit $k = |v| - |u|$. On a $v = ua_1a_2 \ldots a_k$ On construit l'automate obtenu à partir de $\mathcal{A}_H$ en ajoutant les états $p_1, \cdots, p_k \notin Q$ et les transitions $(i \cdot u, a_1, p_1)$ et (p_i, a_{i+1}, p_{i+1}) pour tout $i \in \{1, \ldots, k-1\}$.
Ce qui peut s'illustrer par la figure ci-dessous.



Dans ce nouvel automate chaque lettre induit une fonction injective φ_a de l'ensemble des états $Q' = Q \cup \{p_1, \ldots, p_{|v|-|u|}\}$ dans lui-même. Ces fonctions peuvent être prolongées en des permutations $\hat{\varphi}_a$ de Q'. D'après la propriété universelle du groupe libre, il existe un morphisme de groupe ψ de $F(A)$ dans l'ensemble des permutations de Q' tel que pour tout $a \in A$, $a\psi = \hat{\varphi}_a$. Soit $K = \{u \mid i(u\psi) = i\}$. Par construction, $H \subseteq K$ et K est un sous-groupe de $F(A)$. De plus, $\ker \psi \subseteq K$. Or, $F(A)/\ker \psi$ est isomorphe à l'image de ψ qui est un groupe fini. Donc K est d'indice fini. Il en résulte que $g \notin H'$ et, par conséquent, $H' \subseteq H$. □

2.3.3 Calcul de la fermeture pro-p d'un sous-groupe rationnel

Dans [MSW98], S. Margolis, M. Sapir et P. Weil montrent comment adapter un résultat de P. Ribes et P.A. Zalesskii [RZ94] afin de calculer la fermeture pro-p d'un sous groupe rationnel du groupe libre donné par un automate de sous-groupe. Nous décrivons ici cet algorithme, sans toutefois en donner une justification précise qui serait longue et qui demanderait d'introduire de nombreuses notions. Nous invitons le lecteur intéressé à se référer, pour les justifications, à l'article original [MSW98] ou, pour une description détaillée, à [Wei00].

Soit H un sous-groupe rationnel du groupe libre $F(A)$. Afin de calculer sa fermeture pro-p, on utilise les arguments ci-dessous.

- Comme tout sous-groupe rationnel H' est un sous-groupe libre (i.e. il existe un ensemble fini B tel que H' soit isomorphe à $F(B)$), on peut considérer deux topologies pro-p sur H' : celle de $F(B)$ et celle induite par la topologie de $F(A)$. Elle coïncident si H' est fermé.
- On peut tester en temps polynomial si un sous-groupe du groupe libre est dense pour la topologie pro-p.
- Si H est dense dans H' et si H' et fermé, alors H' est la fermeture de H.

L'algorithme consiste alors à calculer une suite décroissante de sous-groupes rationnels fermés contenant H jusqu'à ce que H soit dense dans le dernier sous-groupe construit, H'.

Dans ce cas, la fermeture pro-p de H est exactement H'. Avant de décrire comment calculer H', commençons par quelques remarques et notations (dont les justifications peuvent être trouvées dans [MSW98]).

(1) Si $A = \{a_1, \ldots, a_k\}$ est un alphabet fini on note σ_A le morphisme de monoïde de $F(A)$ dans l'espace vectoriel $(\mathbb{Z}/p\mathbb{Z})^{|A|}$ tel que $(a_1\sigma_A, \ldots, a_k\sigma_A)$ soit la base canonique (on a, en particulier, $\overline{a_i}\sigma_A = -(a_i\sigma_A)$). Par exemple, $(ab\overline{a})\sigma_A = a\sigma_A + b\sigma_A - a\sigma_A = b\sigma_A$. Un sous-groupe rationnel H est dense dans $F(A)$, pour la topologie pro-p, si et seulement si $H\sigma = (\mathbb{Z}/p\mathbb{Z})^{|A|}$.

(2) Soit H un sous-groupe rationnel de $F(A)$. On rappelle que l'automate de sous-groupe ayant un nombre minimal d'états et reconnaissant H est noté $\mathcal{A}_H$. Pour tout état p de $\mathcal{A}_H$, on considère un mot u_p tel que l'image de l'état initial par u_p soit p. On définit alors la relation $\equiv_A$ sur les états de $\mathcal{A}_H$, par
$$p \equiv_A q \quad \text{ssi} \quad (u_p\overline{u_q})\pi\sigma \in H\sigma$$
On peut montrer que $\equiv_A$ est une relation d'équivalence qui ne dépend pas du choix des u_p. L'automate obtenu en fusionnant dans $\mathcal{A}_H$ les états équivalents reconnaît un sous-groupe rationnel fermé K du groupe libre, qui contient H et pour lequel les deux topologies pro-p coïncident. On note $\mathcal{A}_H/\equiv_A$ cet automate. De plus, si H n'est pas dense dans $F(A)$, alors K est strictement inclus dans $F(A)$.

De la remarque (1) ci-dessus on déduit un algorithme, que l'on appelle `Teste-pdense`, pour tester si un sous-groupe rationnel H de $F(A)$, donné par l'automate $\mathcal{A}_H$, est dense dans pour la topologie pro-p.

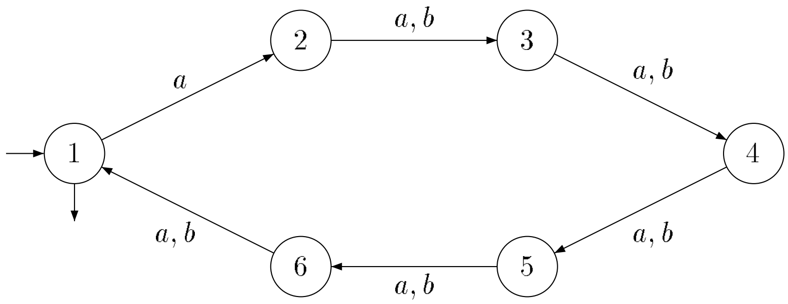
Algorithme 2.25

1. Calculer $h_1, \ldots, h_j$ des éléments de H tels que
$$\langle h_1, \ldots, h_j \rangle = H$$
2. Calculer le rang r de la famille de vecteurs $\{h_1\sigma_A, \ldots, h_j\sigma_A\}$.
3. Si $r = |A|$ alors H est dense, sinon H n'est pas dense.

$\diamondsuit$

L'étape 1 de l'algorithme peut se faire classiquement en calculant un arbre couvrant de l'automate $\mathcal{A}_H$ (pour une description détaillée de cette méthode, voir, par exemple, [MSW98]).

Exemple 2.26 Considérons l'automate $\mathcal{A}_H^+$ suivant.



L'automate dual qui lui est associé reconnaît le sous-groupe H engendré par $a\overline{b}$, $a^2\overline{b}\overline{a}$, $a^3\overline{b}\overline{a}^2$, $a^4\overline{b}\overline{a}^3$, $a^5\overline{b}\overline{a}^4$ et a^6. On calcule l'image par σ_A de chacun de ces éléments:

- $(a\overline{b})\sigma_A = (1, -1)$.
- $(a^2\overline{b}\overline{a})\sigma_A = (1, -1)$.
- $(a^3\overline{b}\overline{a}^2)\sigma_A = (1, -1)$.
- $(a^4\overline{b}\overline{a}^3)\sigma_A = (1, -1)$.
- $(a^5\overline{b}\overline{a}^4)\sigma_A = (1, -1)$.
- $(a^6)\sigma_A = (6, 0)$.

On doit donc calculer le rang r de la famille de vecteurs $\{(1, -1), (6, 0)\}$.

Si $p \geq 5$, alors $r = 2$ et H est dense dans $F(A)$. En revanche, si $p = 2$ ou $p = 3$, alors $r = 1$ et H n'est pas dense.

$\triangledown$

On est maintenant en mesure de décrire un algorithme qui calcule la fermeture pro-p d'un sous-groupe rationnel H de $F(A)$ donné par l'automate de sous-groupe $\mathcal{A}_H$. Pour cela on va effectuer des changements d'alphabets et tester si H est dense dans différents groupes libres le contenant.

On appelle FermetureSousgroupeGp cet algorithme.

Algorithme 2.27

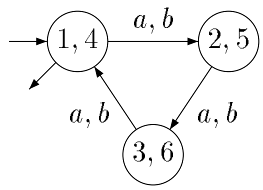
1. $K = F(A)$.
2. $B = A$.
2. Tant que H n'est pas dense dans $F(B)$ faire
   (a) Calculer $\mathcal{A}' = \mathcal{A}_H / \equiv_B$.
   (b) Calculer B tel que $F(B) = L(\mathcal{A}_H / \equiv_B)\pi$.
   (c) Traduire B dans l'alphabet A.
3. Retourner $\mathcal{A}'$.

$\diamondsuit$

Exemple 2.28 Reprenons l'exemple ci-dessus et cherchons la fermeture de H pour la topologie pro-3. Comme on l'a vu, H n'est pas dense dans $F(A)$, avec $A = \{a, b\}$. On calcule donc l'automate $\mathcal{A}_H / \equiv_A$. Pour tout état i de $\mathcal{A}_H$, on peut prendre $u_i = a^{i-1}$. On a alors, si $i \geq j$,

$$(u_i \overline{u_j})\sigma_A = (i - j, 0)$$

Par conséquent, les états i et j sont équivalents si et seulement si $i - j$ est un multiple de 3. On obtient ainsi trois classes d'équivalence : $\{1, 4\}$, $\{2, 5\}$ et $\{3, 6\}$. L'automate $\mathcal{A}_K^+$ associé à K est donc :



Le sous-groupe K est donc engendré par $x = b\overline{a}$, $y = ab\overline{a}^2$, $z = a^2b$ et $t = a^3$. On calcule maintenant H dans cette nouvelle base :

- $a\overline{b} = \overline{x}$,

- $a^2\overline{b}\overline{a} = \overline{y}$,
- $a^3\overline{b}\overline{a}^2 = t\overline{z}$,
- $a^4\overline{b}\overline{a}^3 = t\overline{x}\overline{t}$,
- $a^5\overline{b}\overline{a}^4 = 4\overline{y}\overline{t}$,
- $a^6 = t^2$.

Donc, en posant $B = \{x, y, z, t\}$, on doit maintenant tester si H est dense dans $F(B) = K$. Pour cela on calcule les images par σ_B des générateurs de H. On obtient :

- $(\overline{x})\sigma_b = (-1, 0, 0, 0)$,
- $(\overline{y})\sigma_b = (0, -1, 0, 0)$,
- $(t\overline{z})\sigma_b = (0, 0, -1, 1)$,
- $(t\overline{x}\overline{t})\sigma_b = (-1, 0, 0, 0)$,
- $(4\overline{y}\overline{t})\sigma_b = (0, -1, 0, 0)$,
- $(t^2)\sigma_b = (0, 0, 0, 2)$.

Cette famille de vecteurs est bien de rang 4, donc H est dense dans K. Il en résulte que K est la fermeture pro-3 de H.

$\triangledown$

Soulignons enfin qu'il est prouvé dans [MSW98] que si H est un sous-groupe rationnel du groupe libre dont l'automate de sous-groupe $\mathcal{A}_H$ possède n états et dont la fermeture pro-p est K, alors $\mathcal{A}_K$ possède au plus n états et est calculable en temps $O(n^5)$ (voir proposition 2.23).

2.4 Calcul de la clôture d'une partie rationnelle du groupe libre

Historiquement, le calcul de la clôture profinie d'un langage rationnel a joué un rôle important dans la tentative de résoudre la conjecture de Type II de Rhodes [Rho87] (prouvée par C.J. Ash [Ash91]). A cet effet, J.-E Pin et C. Reutenauer [PR91] ont tout d'abord conjecturé un algorithme pour calculer la fermeture d'un langage donné par une expression rationnelle. La correction de cet algorithme, que nous décrivons dans la partie 2.4.1, fut prouvée par P. Ribes et P.A. Zalesskii [RZ93, RZ94] (voir théorème 2.29).

2.4.1 Algorithme de Pin-Reutenauer

La clef de voûte de l'étude des propriétés topologiques des parties rationnelles est due à P. Ribes et P.A. Zalesskii [RZ93, RZ94]. Une preuve combinatoire de ce résultat est donnée par B. Herwig et D. Lascar dans [HL00]. Récemment, B. Steinberg [Ste00c] a retranscrit une preuve de C.J. Ash [Ash91] de la conjecture de Rhodes afin d'obtenir une preuve utilisant la théorie des automates et des semigroupes.

Théorème 2.29 *Soit* $\mathbf{H}$ *une variété de groupes finis close par extension et* H_1, $H_2, \ldots,$ H_n *un nombre fini de sous-groupes rationnels du groupe libre fermés pour la topologie pro-*$\mathbf{H}$*. Le sous-ensemble* $H_1 H_2 \cdots H_n$ *du groupe libre est aussi fermé pour la topologie pro-*$\mathbf{H}$*.*

Le théorème 2.29 permet de construire un algorithme [PR91] (décrit ci-après), calculant la fermeture pro-$\mathbf{H}$, dans le cas où $\mathbf{H}$ est close par extension, d'une partie rationnelle du groupe libre donnée par une expression rationnelle.

On note

- $\mathrm{Cl}_F^{\mathbf{H}}(S)$ la fermeture pro-$\mathbf{H}$ dans le groupe libre d'un sous-ensemble S du groupe libre.
- $\mathrm{Cl}_{A^*}^{\mathbf{H}}(S)$ la fermeture pro-$\mathbf{H}$ dans le monoïde libre d'un sous-ensemble S du monoïde libre.
- $\mathrm{Cl}_F(S)$ la fermeture profinie dans le groupe libre d'un sous-ensemble S du groupe libre.
- $\mathrm{Cl}_{A^*}(S)$ la fermeture profinie dans le monoïde libre d'un sous-ensemble S du monoïde libre.

L'algorithme suivant fonctionne récursivement sur une expression rationnelle d'une partie rationnelle de $F(A)$.

Algorithme 2.30

Pour éviter des lourdeurs d'écriture, la notation Cl désigne ici $\mathrm{Cl}_F^{\mathbf{H}}$.

Soit $\mathbf{H}$ une variété de groupes finis close par extension.

(1) Si S est fini, alors $\mathrm{Cl}(S) = S$.

(2) Si S_1 et S_2 sont des parties rationnelles de $F(A)$, alors

$$\mathrm{Cl}(S_1 \cup S_2) = \mathrm{Cl}(S_1) \cup \mathrm{Cl}(S_2)$$

(3) Si S est une partie rationnelle de $F(A)$ alors

$$\mathrm{Cl}(S^*) = \mathrm{Cl}(\langle S \rangle)$$

(4) Si S_1 et S_2 sont des parties rationnelles de $F(A)$, alors

$$\mathrm{Cl}(S_1 S_2) = \mathrm{Cl}(S_1)\mathrm{Cl}(S_2)$$

$\diamondsuit$

Il faut remarquer que, dans cet algorithme, il faut savoir calculer la fermeture d'un sous-groupe rationnel du groupe libre. Pour la topologie profinie, tout sous-groupe rationnel étant fermé (théorème 2.24), l'étape (3) devient plus simple. Pour la topologie pro-p, on sait calculer la fermeture d'un sous-groupe rationnel (voir section 2.3.3). On peut aussi adapter cet algorithme pour la topologie pro-$\mathbf{G}_{\rm com}$, même si la variété $\mathbf{G}_{\rm com}$ n'est pas close par extension [Del98].

Nous ne donnons qu'une idée de la preuve de la correction de l'algorithme 2.30, que l'on peut trouver dans [PR91].

Preuve. Dans un espace métrique, un ensemble fini est toujours fermé et la fermeture d'une union finie est toujours l'union des fermetures.

Soient S une partie rationnelle de $F(A)$, et $u \in S$. Comme $u^* \subseteq S^*$, la suite $u^{n!-1}$ converge vers vers $\overline{u}$ (voir proposition 2.7). On montre alors assez facilement que

$$\mathrm{Cl}(S^*) = \mathrm{Cl}(\langle S \rangle)$$

Le fait que si S_1 et S_2 sont des parties rationnelles de $F(A)$, alors

$$\mathrm{Cl}(S_1S_2) = \mathrm{Cl}(S_1)\mathrm{Cl}(S_2)$$

est une conséquence du corollaire 2.32. □

Rappelons que d'après la définition 2.13 de la topologie pro-$\mathbf{H}$ de A^*, on a le résultat suivant.

Proposition 2.31 *Soit $S \subseteq A^*$, $\mathrm{Cl}^{\mathbf{H}}_{A^*}(S) = \mathrm{Cl}^{\mathbf{H}}_{F}(S) \cap A^*$.*

Ce qui permet de calculer la fermeture d'un langage rationnel de A^* en calculant sa clôture dans $F(A)$ et en prenant sa trace sur A^*.

On peut déduire de l'algorithme 2.30 les deux corollaires importants suivants.

Corollaire 2.32 *Soit* $\mathbf{H}$ *une variété de groupes finis close par extension. Une partie rationnelle du groupe libre est fermée pour la topologie pro-*$\mathbf{H}$ *si et seulement si elle peut s'exprimer comme une union finie d'ensembles du type $gH_1H_2\dots H_n$ où $g \in F(A)$ et où les H_i sont des sous-groupes rationnels fermés du groupe libre.*

Corollaire 2.33 *Soit* $\mathbf{H}$ *une variété de groupes finis close par extension. La fermeture, pour la topologie pro-*$\mathbf{H}$ *du groupe [resp. monoïde] libre, d'une partie [resp. langage] rationnelle du groupe [resp. monoïde] libre est une partie [resp. langage] rationnelle du groupe [resp. monoïde] libre.*

2.4.2 Topologie profinie et parties rationnelles

Une des topologies pro-**H** est plus particulièrement liée aux langages rationnels : il s'agit de la topologie profinie associée à la variété **G** de tous les groupes finis.

Les parties rationnelles du groupe libre sont définies comme étant la classe des sous-ensembles du groupe libre contenant les ensembles finis, close par union et par les opérations algébriques produit et étoile (monoïde engendré). Or, dans le groupe libre, il paraît intéressant d'utiliser, comme opérations algébriques, le produit et, non plus le monoïde engendré, mais le sous-groupe engendré. On obtient alors la sous-famille des parties rationnelles suivante :

Définition 2.34 *On note $\mathcal{F}$ la plus petite famille de sous-ensembles de $F(A)$ contenant les ensembles finis et fermée par : union, produit et sous-groupe engendré.*

Le lien étroit que possèdent les parties rationnelles avec la topologie profinie vient du résultat suivant [PR91].

Proposition 2.35 *Un sous-ensemble rationnel S du groupe libre est fermé pour la topologie profinie si et seulement s'il appartient à $\mathcal{F}$.*

On voit que l'on a une topologie qui, d'une part, a des propriétés algébriques très fortes (notamment vis à vis des groupes), et qui, d'autre part, est intimement liée aux parties rationnelles du groupe libre. Or, d'après un résultat de M. Benois [Ben69] (voir théorème 1.66), la trace d'une partie rationnelle du groupe libre sur le monoïde libre est un langage rationnel : on peut donc, d'une certaine manière, redescendre du groupe libre vers le monoïde libre (pour la rationnalité). Nous allons voir que l'on peut aussi redescendre les propriétés topologiques du groupe libre vers le monoïde libre.

2.4.3 Clôture d'une partie rationnelle donnée par un automate

Une partie rationnelle du groupe libre peut aussi être donnée par un automate (cf corollaire 1.67). On est donc amené à se demander comment calculer la fermeture (pour la topologie pro-**H**) d'une partie rationnelle donnée par un automate. Une méthode consiste à calculer une expression rationnelle de l'ensemble et à appliquer ensuite l'algorithme 2.30. Cependant, le calcul d'une expression rationnelle peut être exponentiel en espace mémoire [EZ76] et donc en temps. Nous allons montrer ici que cette étape n'est pas nécessaire et que l'algorithme 2.30 peut être appliqué directement sur un automate, ce qui permet ainsi de calculer la fermeture d'une partie rationnelle (donnée par un automate) en temps polynomial (à condition que l'on sache calculer en temps polynomial la fermeture d'un sous-groupe rationnel). Les algorithmes développés ici l'ont été aussi, de façon indépendante et par une approche différente, par B. Steinberg [Ste00a]. Dans toute la partie 2.4.3, **H** est une variété de groupes finis close par extension.

Nous allons donc étudier comment traduire l'algorithme 2.30 sur un automate fini. Dans l'algorithme 2.30, le point sensible est le point (3). Commençons donc par traiter le cas du calcul du sous-groupe engendré, dans un cadre un peu plus général.

Proposition 2.36 *Soient $\mathcal{A} = (Q, \tilde{A}, E, I, F)$ un automate émondé et*

$$E_0 = E \cup \{(q, \overline{a}, p) \mid (p, a, q) \in E\} \cup \{(p, \varepsilon, q) \mid p, q \in F \cup I\}$$

L'automate $\mathcal{A}_0 = (Q, \tilde{A}, E_0, I \cup F, F \cup I)$ satisfait

$$\langle L(\mathcal{A})\pi \rangle = L(\mathcal{A}_0)\pi$$

Preuve. Montrons d'abord que $L(\mathcal{A}_0)$ est un sous-monoïde de $\tilde{A}^*$ stable par l'opération $\bar{\cdot}$. Soient u et v dans $L(\mathcal{A})$ et m_u et m_v deux chemins réussis dans $\mathcal{A}_0$ d'étiquettes respectives u et v. Soient enfin p l'état terminal de m_u et q l'état d'origine de m_v. D'après la définition de E_0, $m_u(p, \varepsilon, q)m_v$ est un chemin réussi de $\mathcal{A}_0$, donc $uv \in L(\mathcal{A}_0)$. De plus, par définition de E_0, ε appartient à $L(\mathcal{A}_0)$, qui est donc un sous-monoïde de $\tilde{A}^*$. De plus, si

$$m_u = (p_0, a_0, p_1), \cdots, (p_n, a_n, p_{n+1}) \quad (\text{avec } p_{n+1} = p)$$

alors, d'après la définition de $\mathcal{A}_0$, le chemin

$$m'_u = (p_{n+1}, \overline{a}, p_n), \cdots, (p_1, \overline{a_0}, p_0)$$

est aussi un chemin réussi dans $\mathcal{A}_0$. Par conséquent, $\overline{u} \in L(\mathcal{A}_0)$. Il en résulte que $L(\mathcal{A}_0)\pi$ est un sous-groupe de $F(A)$.

Réciproquement, soit $x \in L(\mathcal{A}_0)\pi$. Il existe un chemin réussi m dans $\mathcal{A}_0$ (utilisant éventuellement des ε-transitions) tel que $[m]\pi = x$. Comme l'ensemble des ε-transitions de $\mathcal{A}_0$ forme, par construction, un graphe transitif, on peut considérer que dans m on n'emprunte pas deux ε-transitions consécutivement. Ce chemin peut être décomposé en une suite de chemins m_i de $\mathcal{A}_0$ n'utilisant pas d'ε-transition :

$$m = m_0, (p_0, \varepsilon, q_1), m_1, (p_1, \varepsilon, q_2), \cdots, (p_{\ell-1}, \varepsilon, q_\ell), m_\ell$$

où p_i est l'état terminal de m_i et q_i son origine. Soit $x_i = [m_i]\pi$. Comme π est un morphisme, $x = x_0 \cdots x_\ell$.

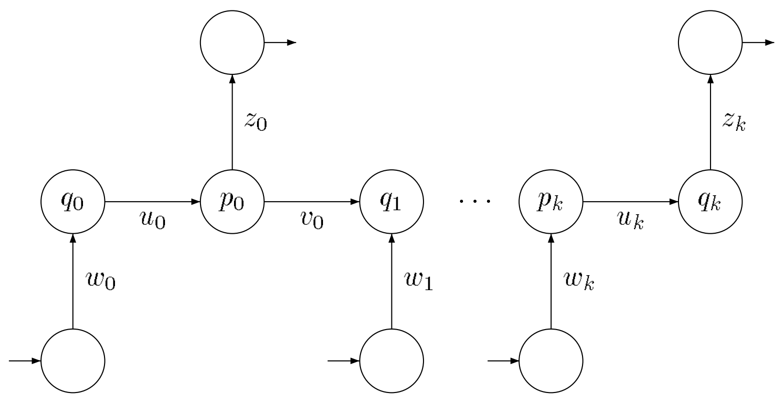
Prouvons maintenant que $x_i \in \langle L(\mathcal{A})\pi \rangle$. En effet, m_i peut être décomposé en une suite de chemins r_j, s_j utilisant des transitions respectivement dans E et $E_0 \setminus E$ et telle que :

$$m_i = r_0, s_0, r_1, s_1, \cdots, r_k, s_k \tag{2.1}$$

Notons que si la première transition de m_i est dans $E_0 \setminus E$, alors r_0 est vide et que, de même, si la dernière transition de m_i est dans E_0, alors s_k est vide. Soient u_j l'étiquette du chemin r_j, p_j son état d'origine et q_j son état d'arrivée. D'après (2.1), l'origine de s_j est q_j et sa fin est p_{j+1} ; on note v_j l'étiquette de s_j.
Comme $\mathcal{A}$ est émondé, pour chaque p_j il existe un mot $w_j \in \tilde{A}^*$ tel que $q_j \in I \cdot w_j$ (dans $\mathcal{A}$) et, de même, pour tout q_j, il existe un mot $z_j \in \tilde{A}^*$ tel que $p_j \cdot z_j \in F$ (dans $\mathcal{A}$). Il en résulte que

$$w_j u_j z_j \in L(\mathcal{A}) \text{ et } w_{i+1}\overline{v_i}z_i \in L(\mathcal{A})$$

ce qui peut être illustré par la figure suivante.



On pose

$$t = \left(w_0u_0z_0\overline{(w_1\overline{v_0}z_0)}w_1u_1z_1\overline{(w_2\overline{v_1}z_1)}\cdots w_ku_kz_k\right)\pi$$

Par construction $t \in \langle L(\mathcal{A})\pi\rangle$. En observant que

$$w_ju_jz_j\overline{(w_{j+1}\overline{v_j}z_j)} = w_ju_jz_j\overline{z_j}v_j\overline{w_{j+1}}$$

on obtient

$$t = (w_0u_0z_0\overline{z_0}v_0\overline{w_1}w_1u_1z_1\overline{z_1}v_1\overline{w_2}\cdots w_ku_kz_k)\pi$$

On en déduit que

$$t = (w_0u_1v_1 ... u_{k-1}v_{k-1}u_kz_k)\pi = (w_0\pi)x_i(z_k\pi)$$

Si $q_0 \in I$, on peut choisir $w_0 = 1$ et si $q_0 \in F$, alors $w_0 \in L(\mathcal{A})$. On peut donc prendre w_0 tel que $w_0\pi \in \langle L(\mathcal{A})\pi\rangle$. Pour la même raison, on peut choisir z_k tel que $z_k\pi \in \langle L(\mathcal{A})\pi\rangle$. Par conséquent, $x_j \in \langle L(\mathcal{A})\pi\rangle$.

Il en résulte alors que $x \in \langle L(\mathcal{A})\pi\rangle$. $\square$

Il en découle que l'on peut facilement calculer le sous-groupe engendré par un sous-ensemble rationnel de $F(A)$.

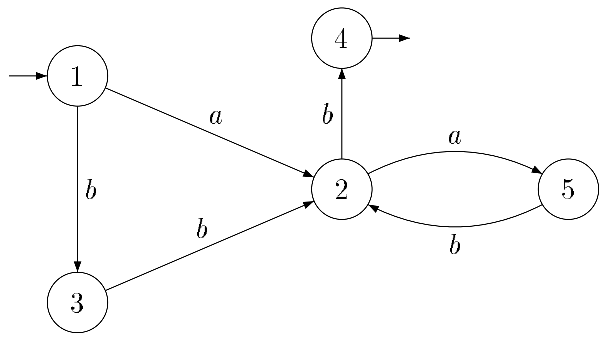
Corollaire 2.37 *Soit $\mathcal{A}$ un automate émondé à n états sur l'alphabet $\tilde{A}$. On peut calculer en temps $O(n^2)$ un automate de sous-groupe $\mathcal{A}_0$ tel que $\langle L(\mathcal{A})\pi \rangle = L(\mathcal{A}_0)\pi$.*

Preuve. En effet, d'après la proposition 2.36, il suffit de fusionner les états initiaux et finaux de $\mathcal{A}$, de le rendre dual et de lui appliquer l'algorithme 1.79.

Pour la complexité, fusionner les états initiaux et finaux nécessite un temps $O(n^2)$. De plus, si l'on considère que l'automate a m transitions, alors le rendre dual se fait en temps $O(m) = O(n^2)$ et, pour appliquer l'algorithme 1.79, on a besoin d'un temps en $O(n^2)$. D'où une complexité totale en $O(n^2)$. □

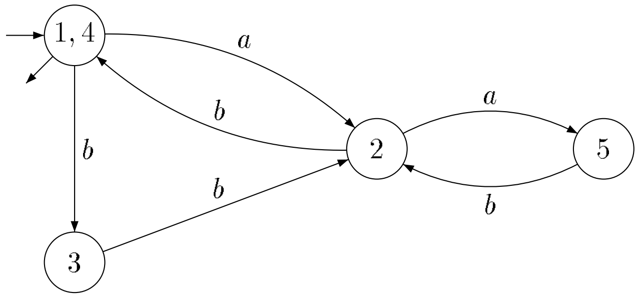
On appelle `GroupeEngendré` l'algorithme du corollaire 2.37.

Exemple 2.38 Considérons l'automate $\mathcal{A}$ suivant :

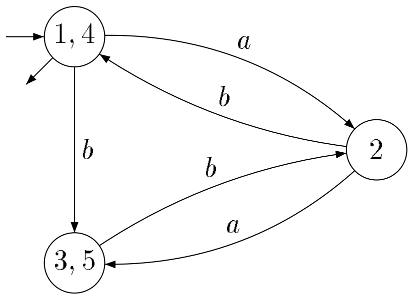


La première étape consiste à fusionner les états initiaux et finaux (ici 1 et 4), et à rendre l'automate dual (on ne représente pas les transitions duales).

On obtient l'automate :



On applique maintenant la réduction de l'algorithme 1.79 : on fusionne les états 3 et 5 et on obtient l'automate :



∇

Pour appliquer l'algorithme ci-dessus, l'automate n'a pas besoin d'être déterministe. On en déduit donc, à l'aide de la proposition 1.65, le résultat suivant :

Corollaire 2.39 *Soit $\mathcal{E}$ une expression rationnelle de taille n représentant un langage L de A^*. On peut calculer en temps $O(n^2)$ l'automate de sous-groupe $\mathcal{A}_{\langle L \rangle}$.*

Dans le cas où l'automate donné est fortement connexe, on obtient le résultat suivant qui nous sera utile par la suite.

Corollaire 2.40 *Soient $\mathcal{A} = (Q, \tilde{A}, E, I, F)$ un automate fortement connexe sur $\tilde{A}$, i et j deux éléments de Q et $w \in \tilde{A}^*$ tel que $j \in i \cdot w$. Alors*

$$\langle L(\mathcal{A}^i)\pi \rangle = w\pi \langle L(\mathcal{A}^j)\pi \rangle (\overline{w}\pi)$$

Preuve. Le automates $(Q, \tilde{A}, E_0, i, i)$ et $(Q, \tilde{A}, E_0, j, j)$ sont fortement connexes et possèdent un état final : ils sont donc émondés. Donc, d'après la proposition 2.36,

$$\langle L(\mathcal{A}^i)\pi \rangle = L((Q, \tilde{A}, E_0, i, i))\pi \quad \text{et} \quad \langle L(\mathcal{A}^j)\pi \rangle = L((Q, \tilde{A}, E_0, j, j))\pi$$

où

$$E_0 = E \cup \{(q, \overline{a}, p) \mid (p, a, q) \in E\}$$

Par conséquent, d'après la proposition 1.80,

$$\langle L(\mathcal{A}^i)\pi \rangle = (w\pi) \langle L(\mathcal{A}^j)\pi \rangle (\overline{w}\pi)$$

$\square$

Nous allons maintenant voir que si l'on sait calculer la fermeture pro-**H** d'un sous-groupe rationnel du groupe libre, alors on peut déduire de la proposition 2.36 un algorithme de calcul de la fermeture pro-**H** d'un langage rationnel donné par un automate dont le graphe est fortement connexe.

Plus précisément, on suppose que l'on dispose d'un algorithme que l'on appelle `FermetureGroupeH` et qui, étant donné un sous-groupe rationnel H du groupe libre donné par l'automate $\mathcal{A}_H$, calcule l'automate de sous-groupe de $\mathrm{Cl}_F^{\mathbf{H}}\,(H)$. On note f une fonction de $\mathbb{N}$ dans $\mathbb{R}$ qui borne le temps de calcul de `FermetureGroupeH`: si $\mathcal{A}_H$ possède n états, alors cet algorithme calcule $\mathcal{A}_{\mathrm{Cl}_F^{\mathbf{H}}\,(H)}$ en temps au plus $O(n\mathsf{f})$. De façon naturelle, on suppose que f est une fonction croissante et que pour tous n_1, n_2, $(n_1+n_2)\mathsf{f} \geq n_1\mathsf{f} + n_2\mathsf{f}$. Remarquons que, comme on l'a vu à la section 2.3.3, pour la topologie pro-p, on peut prendre $n\mathsf{f} = n^5$.

Rappelons enfin que, d'après la proposition 2.23, $\mathcal{A}_{\mathrm{Cl}_F^{\mathbf{H}}\,(H)}$ possède moins d'états que $\mathcal{A}_H$.

Avant d'aborder l'algorithme, on a besoin du lemme suivant.

Lemme 2.41 *Soit* **H** *une variété de groupes finis close par extension. Soit $\mathcal{A}$ un automate fortement connexe sur $\tilde{A}$ ne possédant qu'un seul état initial i et qu'un seul état final f. Soit enfin u et v tels que $f \in i \cdot u$ et $i \in f \cdot v$. On a*

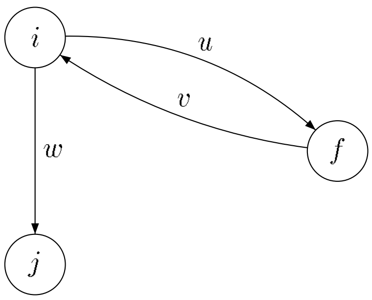
$$\mathrm{Cl}_F^{\mathbf{H}}\,(L(\mathcal{A})\pi) = \mathrm{Cl}_F^{\mathbf{H}}\,(\langle L(\mathcal{A}^i)\pi\rangle)(u\pi) = \mathrm{Cl}_F^{\mathbf{H}}\,(\langle L(\mathcal{A}^i)\pi\rangle)(\overline{v}\pi)$$

De plus, pour tout état j et tout mot w tel que $j \in i \cdot w$, on a

$$\mathrm{Cl}_F^{\mathbf{H}}\,(L(\mathcal{A}^{j,f})\pi = (w\pi)\mathrm{Cl}_F^{\mathbf{H}}\,(\langle L(\mathcal{A}^i)\pi\rangle)(\overline{w}u\pi)$$

Preuve. Pour éviter des lourdeurs, on écrit ici Cl à la place de $\mathrm{Cl}_F^{\mathbf{H}}$.

La figure suivante illustre la situation.



Si $u' \in L(\mathcal{A}^i)$, alors $u'u \in L(\mathcal{A})$; par conséquent,

$$L(\mathcal{A}^i)u \subseteq L(\mathcal{A})$$

Il en résulte que, si $x = u\pi$,

$$\mathrm{Cl}(L(\mathcal{A}^i)\pi)x \subseteq \mathrm{Cl}(L(\mathcal{A})\pi)$$

Soit maintenant $y = v\pi$. On a de même

$$\mathrm{Cl}(L(\mathcal{A})\pi)y \subseteq \mathrm{Cl}(L(\mathcal{A}^i)\pi)$$

Par conséquent,

$$\mathrm{Cl}(L(\mathcal{A}^i)\pi)x \subseteq \mathrm{Cl}(L(\mathcal{A})\pi) \subseteq \mathrm{Cl}(L(\mathcal{A}^i)\pi)\overline{y} \tag{2.2}$$

De plus, $\mathcal{A}^i$ ne possède qu'un état final qui est confondu avec l'état initial donc $L(\mathcal{A}^i)^* = L(\mathcal{A}^i)$ et, par conséquent, d'après l'algorithme 2.30,

$$\mathrm{Cl}(L(\mathcal{A}^i)\pi) = \mathrm{Cl}(\langle L(\mathcal{A}^i)\rangle\pi)$$

Il en résulte que (2.2) devient

$$\mathrm{Cl}(\langle L(\mathcal{A}^i)\pi\rangle)x \subseteq \mathrm{Cl}(L(\mathcal{A})\pi) \subseteq \mathrm{Cl}(\langle L(\mathcal{A}^i)\rangle\pi)\overline{y} \tag{2.3}$$

Or $\langle L(\mathcal{A}^i)\pi\rangle x = \langle L(\mathcal{A}^i)\pi\rangle xy\overline{y}$. Et comme $xy \in L(\mathcal{A}^i)\pi$, on a $\langle L(\mathcal{A}^i)\pi\rangle xy = \langle L(\mathcal{A}^i)\pi\rangle$. Il en résulte que

$$\langle L(\mathcal{A}^i)\pi\rangle x = \langle L(\mathcal{A}^i)\pi\rangle\overline{y}$$

Donc, (2.3) nous donne

$$\mathrm{Cl}(L(\mathcal{A})\pi) = \mathrm{Cl}(\langle L(\mathcal{A}^i)\pi\rangle)(u\pi) = \mathrm{Cl}(\langle L(\mathcal{A}^i)\pi\rangle)(\overline{v}\pi)$$

Montrons maintenant l'autre égalité. Le résultat précédent, appliqué à $\mathcal{A}^{j,f}$, nous donne

$$\mathrm{Cl}(L(\mathcal{A}^{j,f})\pi) = \mathrm{Cl}(\langle L(\mathcal{A}^j)\pi\rangle)(\overline{vw}\pi)$$

Par conséquent, en utilisant le corollaire 2.40, on obtient :

$$\mathrm{Cl}(\langle L(\mathcal{A}^j)\pi\rangle) = \overline{w}\pi\mathrm{Cl}(\langle L(\mathcal{A}^i)\pi\rangle)(w\pi)$$

Comme, $(w\overline{vw}\pi) = \overline{v}\pi$, on a donc

$$\mathrm{Cl}(L(\mathcal{A}^{j,f})\pi = (w\pi)\mathrm{Cl}(\langle L(\mathcal{A}^i)\pi\rangle)(\overline{w}u_f\pi)$$

□

On en déduit le résultat suivant.

Proposition 2.42 *Soit $\mathcal{A} = (Q, \tilde{A}, E, I, F)$ un automate fortement connexe à n états. On peut calculer en temps $O(n\mathsf{f} + n^2)$ un automate $\mathcal{A}_0$ sur $\tilde{A}$ tel que $L(\mathcal{A}_0)\pi = \mathrm{Cl}_F^{\mathbf{H}}\,(L(\mathcal{A}))\pi$.*

On utilise pour cela l'algorithme FermetureConnexeH suivant :

Algorithme 2.43

1. Soit i_0 un état initial fixé.
2. Pour chaque état p qui est initial ou final, calculer u_p tel que $i_0 \cdot u_p = p$.
3. Soit $\mathcal{A}_1 =$ GroupeEngendré$((Q, \tilde{A}, E, i_0, i_0))$.
4. Soit $\mathcal{A}_2 = (Q_2, \tilde{A}, E_2, i_2, i_2) =$ FermetureGroupeH$(\mathcal{A}_1)$.
5. Soit $I_2 = \emptyset$.
6. Pour chaque $i \in I$, faire $I_2 = I_2 \cup \{i_2 \cdot_{\mathcal{A}_2} u_i\}$.
7. Soit $F_2 = \emptyset$.
8. Pour chaque $f \in F$, faire $F_2 = F_2 \cup \{i_2 \cdot_{\mathcal{A}_2} u_f\}$.
9. Retourner $\mathcal{A}_3 = (Q_2, \tilde{A}, E_2, I_2, F_2)$.

$\diamondsuit$

Preuve. Commençons par évaluer la complexité de cet algorithme en supposant que l'automate a m transitions.

1. Cette étape peut être effectuée en temps $O(1)$.
2. On peut réaliser cette étape en calculant un arbre couvrant de $\mathcal{A}$. Chaque fois que l'on rencontre un état initial ou final, on s'arrête et on écrit le mot correspondant. Calculer un arbre couvrant peut se faire en temps $O(m+n)$. On s'arrête au plus n fois et l'on écrit alors un mot de taille au plus n. On peut donc le faire en temps $O(n^2)$.
3. Cette étape peut être effectuée en temps $O(n^2)$ d'après la proposition 1.78 et le corollaire 2.37.
4. Cette étape nécessite un temps $O(n'\mathsf{f})$ où n' est le nombre d'états de $\mathcal{A}_1$. Mais, d'après la proposition 1.78 et le corollaire 2.37, $n' \leq n$. Comme f est une fonction croissante, cette étape peut être réalisée en temps $O(n\mathsf{f})$.

5-8. Comme $\mathcal{A}_2$ est déterministe (c'est un automate de sous-groupe), ces étapes peuvent être réalisées en un temps total $O(n^2)$.

9. Cette étape se fait en temps proportionnel à la taille du résultat, soit $O(n\mathsf{f})$.

On obtient donc un complexité totale en $O(n\mathsf{f} + n^2)$

Prouvons maintenant la correction de l'algorithme, c'est-à-dire que

$$\mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A})\pi) = L(\mathcal{A}_3)\pi$$

En effet, d'après le lemme 2.41,

$$\mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A}^{i_0})\pi) = L(\mathcal{A}_2)\pi \tag{2.4}$$

De plus,

$$L(\mathcal{A}_3) = \bigcup_{i\in I_2, f\in F_2} L(\mathcal{A}_2^{i,f})$$

D'où, d'après la proposition 1.80 et en posant $x_p = u_p\pi$,

$$L(\mathcal{A}_3)\pi = \bigcup_{i\in I_2, f\in F_2} \overline{x_i} L(\mathcal{A}_2)\pi x_f \tag{2.5}$$

Or

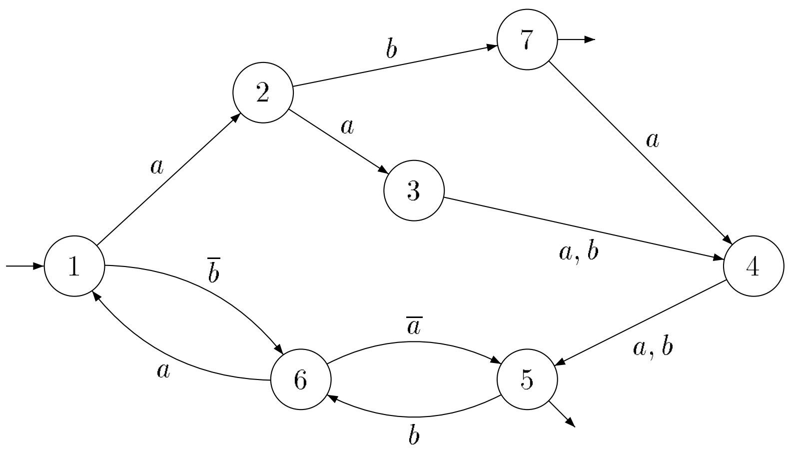
$$L(\mathcal{A}) = \bigcup_{i\in I, f\in F} L(\mathcal{A}^{i,f})$$

On a donc

$$\begin{aligned}
\mathrm{Cl}_F^{\mathbf{H}}\,(L(\mathcal{A})\pi) &= \bigcup_{i\in I, f\in F} \mathrm{Cl}_F^{\mathbf{H}}\,(L(\mathcal{A}^{i,f}\pi)) \\
&= \bigcup_{i\in I, f\in F} \left(\mathrm{Cl}_F^{\mathbf{H}}\,(\langle L(\mathcal{A}^{i}\pi\rangle)\right)\overline{x_i}x_f && \text{par le lemme 2.41} \\
&= \bigcup_{i\in I, f\in F} \overline{x_i}\left(\mathrm{Cl}_F^{\mathbf{H}}\,(\langle L(\mathcal{A}^{i_0}\pi\rangle)\right)x_i\overline{x_i}x_f && \text{par le lemme 2.41} \\
&= \bigcup_{i\in I_2, f\in F_2} \overline{x_i}\left(L(\mathcal{A}_2)\pi\right)x_f && \text{d'après (2.4)} \\
&= L(\mathcal{A}_3)\pi && \text{d'après (2.5)}
\end{aligned}$$

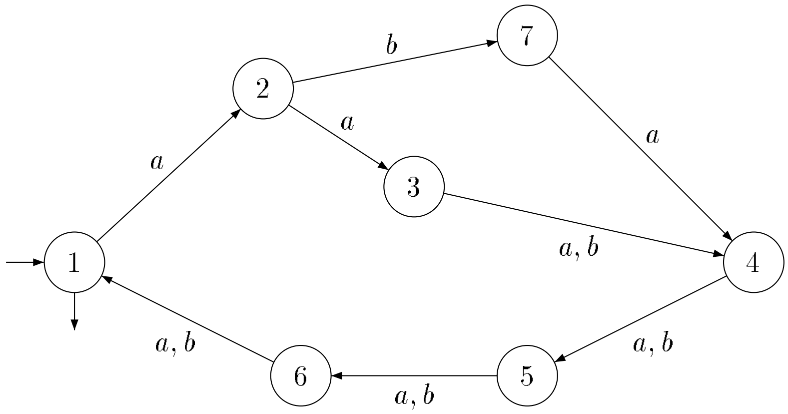
□

Exemple 2.44 Considérons l'automate $\mathcal{A}$ fortement connexe suivant :

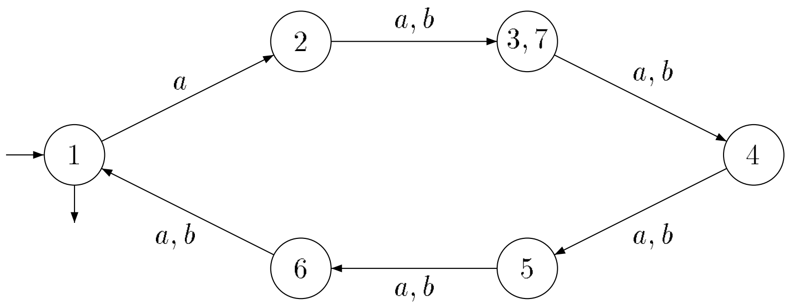


Cherchons la fermeture, pour la topologie pro-3, de $L(\mathcal{A})\pi$. Il n'y a qu'un seul état initial 1 ($u_1 = \varepsilon$) et deux états finaux 7 ($u_7 = ab$) et 5 ($u_5 = \overline{b}\overline{a}$).

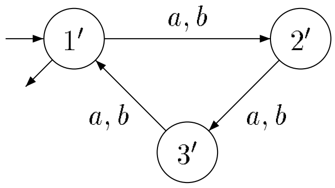
On rend maintenant l'automate dual. On obtient alors l'automate suivant (on ne représente pas les transitions duales) :



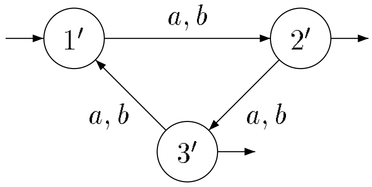
On applique alors l'algorithme 1.79 de réduction et on obtient :



Il est montré dans [MSW98] que la fermeture pro-3 de ce sous-groupe est donnée par l'automate suivant (voir aussi l'exemple 2.28) :



Pour obtenir les états finaux, on calcule dans ce nouvel automate $1' \cdot u_7 = 3'$ et $1' \cdot u_5 = 2'$. La fermeture pro-3 de $L(\mathcal{A})\pi$ est donc reconnue par l'automate dual suivant (on ne représente pas les transitions duales) :



∇

Si l'on regarde attentivement l'algorithme 2.30, on voit que pour calculer la fermeture d'une partie rationnelle donnée par une expression rationnelle, les seules modifications apportées à celle-ci ont lieu lorsque l'on rencontre une étoile. Or, dans une partie rationnelle donnée par un automate fini, les étoiles sont codées dans les boucles, c'est-à-dire dans les composantes fortement connexes de l'automate. Or, on vient de voir comment calculer la fermeture d'un sous-ensemble donné par un automate fortement connexe. On va maintenant montrer qu'il suffit, en quelque sorte, d'appliquer l'algorithme 2.43 `FermetureConnexeH` sur chaque composante fortement connexe d'un automate afin de calculer la fermeture du sous-ensemble qu'il reconnaît.

Théorème 2.45 *Soit $\mathcal{A}$ un automate sur $\tilde{A}$ à n états. On peut calculer en temps $O(n\mathsf{f}+n^2)$ un automate $\mathcal{A}_0$ ayant au plus n états sur $\tilde{A}$ tel que $L(\mathcal{A}_0)\pi = \mathrm{Cl}_F^{\mathbf{H}}\,(L(\mathcal{A})\pi)$.*

On utilise pour cela l'algorithme `FermetureH` décrit ci-dessous. Soit $\mathcal{A}$ un automate dont on note $G_k = (Q_k, E_k)$, avec $1 \leq k \leq \ell$, les composantes fortement connexes.

Algorithme 2.46

1. Calculer $G_1, \cdots G_\ell$ les composantes fortement connexes de $\mathcal{A}$.
2. Pour tout $k \in \{1, \ldots, \ell\}$, choisir $g_k \in Q_k$.
3. Calculer l'ensemble T des transitions externes de $\mathcal{A}$.
4. Pour chaque $1 \leq k \leq \ell$,
   (a) Pour chaque état p de G_k qui est initial ou final, ou qui est état d'origine ou final d'une transition de T, calculer u_p tel que $p \in g_k \cdot u_p$.

(b) Soit $\mathcal{A}'_k$ = <u>`FermetureConnexeH`</u>$((Q_k, \tilde{A}, E_k, g_k, g_k))$. On pose alors $\mathcal{A}'_k = (Q'_k, \tilde{A}, E'_k, q'_k, F'_k)$.

(c) Pour chaque élément p de Q_k qui est initial ou final, ou qui est l'origine ou la fin d'une transition de T, calculer l'état p' de $\mathcal{A}'_k$ tel que $p' = q'_k \cdot_{\mathcal{A}'_k} u_p$.

5. Soit $Q' = \cup_{1 \leq k \leq \ell} Q'_k$.
6. Soit $I' = \{i' \mid i \in I\}$ et $F' = \{f' \mid f \in F\}$.
7. Soit $T' = \{(p', a, q') \mid (p, a, q) \in T\}$.
8. Soit $E' = (\cup_{1 \leq k \leq \ell} E'_k) \cup T'$.
9. Retourner $\mathcal{A}' = (Q', \tilde{A}, E', I', F')$.

$\diamondsuit$

Preuve. Analysons d'abord la complexité de l'algorithme. On suppose que $\mathcal{A}$ possède m transitions. On note n_k le cardinal de Q_k et m_k celui de E_k.

1. Cette étape peut être effectuée en temps $O(m+n) = O(n^2)$.
2. Comme il y a au plus n composantes fortement connexes, cette étape se réalise en temps $O(n)$.
3. Il suffit ici de tester pour chaque transition si ses deux extrémités sont dans la même composante fortement connexe. On peut donc le faire en temps $O(n+m) = O(n^2)$.
4. Cette étape est répétée pour chaque composante fortement connexe.
   (a) Pour un certain nombre d'états $s \in Q_k$, on doit calculer un mot u_s tel que $g \cdot u_s = s$. Il suffit de calculer un arbre couvrant de G_k ayant sa racine en g_k et on s'arrête à chaque fois que l'on rencontre un tel état s. Calculer un arbre couvrant se fait en temps $O(m_k + n_k)$. On s'arrête au plus n_k fois en écrivant à chaque fois un mot de taille au plus n_k. D'où un temps, pour cette étape, en $O(m_k + n_k + n_k^2) = O(n_k^2)$.
   (b) D'après la proposition 2.42, cette étape s'effectue en temps $O(n_k^2 + n_k \mathsf{f})$.
   (c) D'après la proposition 2.42, l'automate $\mathcal{A}'_k$ est un automate de sous-groupe ; il est donc déterministe. On peut donc calculer $g'_k \cdot_{\mathcal{A}'_k} u$ en temps $O(|u|)$. Or, ici, $|u| \leq n_k$ et comme on ne fait qu'au plus n_k calculs de ce type, cette étape peut se réaliser en temps $O(n_k^2)$.
5. Comme on a déjà calculé les Q'_k, on peut calculer leur union en temps $O(n)$.
6. Cette étape s'effectue en temps $O(|I| + |F|) = O(n)$.
7. On peut calculer T' en temps $O(|T|) = O(m)$.
8. De même que pour 6., on peut réaliser cette étape en temps $O(n)$.

9. Cette étape se fait en temps proportionnel à la taille du résultat.
On peut donc calculer FermetureH($\mathcal{A}$) en temps

$$O(m+n) + \sum_{k=1}^{\ell} \left(O(n_k \mathsf{f} + n_k^2)\right)$$

Or

$$\sum_{k=1}^{\ell} \left(O(n_k \mathsf{f} + n_k^2)\right) = O(n^2) + \sum_{k=1}^{\ell} O(n_k \mathsf{f})$$

Donc, comme $\sum_{k=1}^{\ell} O(n_k \mathsf{f}) = O(n\mathsf{f})$, on peut calculer FermetureH($\mathcal{A}$) en temps $O(n^2 + n\mathsf{f})$.

Montrons maintenant que l'automate $\mathcal{A}'$ obtenu à la fin de l'algorithme vérifie :

$$L(\mathcal{A}')\pi = \mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A})\pi)$$

Soit $\mathcal{A}_k = (Q_k, \tilde{A}, E_k, g_k, g_k)$. Le langage $L(\mathcal{A})$ peut être décomposé en l'union des langages de $\tilde{A}^*$ de la forme :

$$L_0 a_0 L_1 a_1 \cdots a_r L_r$$

avec $L_j = L(\mathcal{A}_{k_j}^{p_j, q_j})$ où p_j et q_j appartiennent à Q_{k_j} et où $(q_j, a_j, p_{j+1}) \in T$, $p_0 \in I$, $q_r \in F$. Or, d'après l'algorithme 2.30, la fermeture de $L(\mathcal{A})\pi$ est obtenue en prenant l'union des

$$\mathrm{Cl}_F^{\mathbf{H}}\ ((L_0\pi)a_0(L_1\pi)a_1 \cdots a_r(L_r\pi))$$

qui est égale à

$$\mathrm{Cl}_F^{\mathbf{H}}\ (L_0\pi)a_0\mathrm{Cl}_F^{\mathbf{H}}\ (L_1\pi)a_1 \cdots a_r\mathrm{Cl}_F^{\mathbf{H}}\ (L_r\pi)$$

Mais, d'après la proposition 2.42, on a

$$\mathrm{Cl}_F^{\mathbf{H}}\ (L_j\pi) = \mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A}_{k_j}^{p_j,q_j})\pi) = \mathrm{Cl}_F^{\mathbf{H}}\ (\overline{u_{p_j}}(L(\mathcal{A}_{k_j})\pi)u_{q_j})$$

où u_{p_j} et u_{k_j} sont les mots calculés à l'étape 4.(a). Or on a

$$\mathrm{Cl}_F^{\mathbf{H}}\ ((\overline{u_{p_j}})(L(\mathcal{A}_{k_j})\pi)u_{q_j}) = \overline{u_{p_j}}(\mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A}'_{k_j})\pi))u_{q_j} = \overline{u_{p_j}}(L(\mathcal{A}'_{k_j})\pi)u_{q_j}$$

Donc

$$\mathrm{Cl}_F^{\mathbf{H}}\ ((\overline{u_{p_j}})(L(\mathcal{A}_{k_j})\pi)u_{q_j}) = L(\mathcal{A}_{k_j}^{p'_j,q'_j})\pi$$

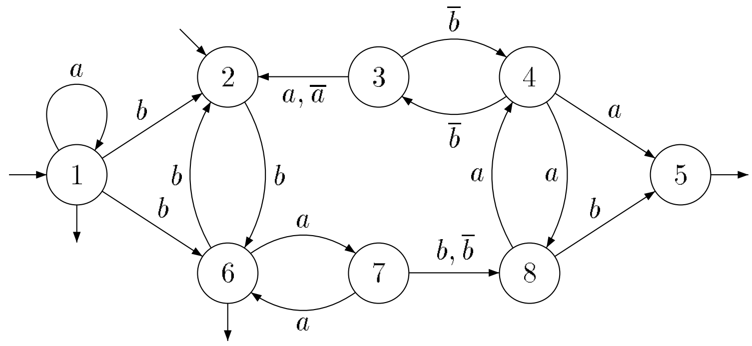
Par conséquent, $\mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A})\pi)$ est exactement l'union des langages

$$(L'_0\pi)a_1(L'_1\pi)a_2 \cdots a_r(L'_r\pi)$$

où $L'_j = L(\mathcal{A}_{k_j}^{p_j,q_j{}'})$, $(q'_j, a_j, p'_{j+1}) \in T'$, $p'_0 \in I'$, $q'_r \in F'$, ce qui achève la preuve de la correction de l'algorithme.

D'après la proposition 2.23 l'automate obtenu a au plus n états car chaque composante fortement connexe a un nombre d'états qui diminue au cours de l'algorithme. □

Exemple 2.47 Cherchons par exemple la fermeture pour la topologie pro-2 du sous-ensemble $L(\mathcal{A})\pi$ du groupe libre défini par l'automate suivant :

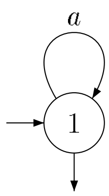


On commence par calculer les composantes fortement connexes de $\mathcal{A}$. On en a trois : $G_1 = \{1\}$, $G_2 = \{2,3,4,6,7,8\}$ et $G_3 = \{5\}$.

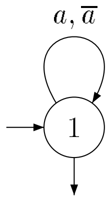
On choisit un représentant par composante, par exemple $1, 2$ et 5.

L'ensemble des transitions externes est $\{(1,b,2), (1,b,6), (4,a,5), (8,b,5)\}$.

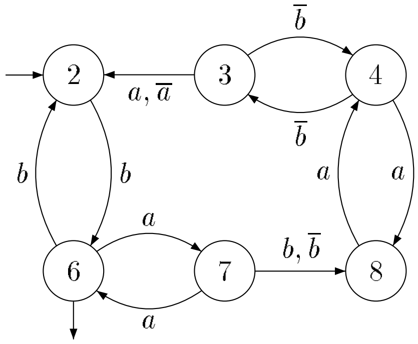
Traitons maintenant chaque composante fortement connexe selon l'étape 4 de l'algorithme. La première composante fortement connexe ne contient qu'un seul état qui est à la fois initial et final. On prend donc $u_1 = \varepsilon$. Il faut maintenant appliquer l'algorithme 2.43 FermetureConnexeH à l'automate :



Or la fermeture pro-2 de a^* est $\langle a \rangle$. On obtient donc l'automate de sous-groupe ayant un nombre minimal d'états reconnaissant $\langle a \rangle$ qui est le suivant :

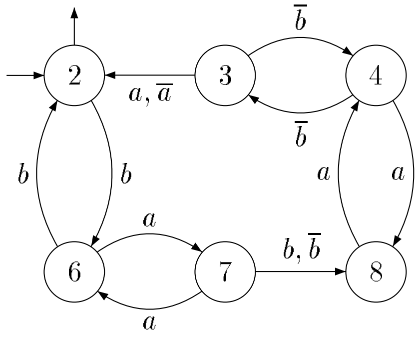


Occupons nous du cas de la seconde composante fortement connexe :

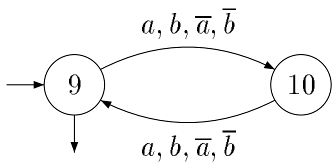


L'état 2 est initial ($u_2 = \varepsilon$), l'état 6 est final $u_6 = b$, les états 2 et 6 sont aussi extrémités finales de transitions de T et les états 4 ($u_4 = ba\overline{b}a$) et 8 ($u_8 = ba\overline{b}$) sont extrémités initiales de transitions de T.

Il faut maintenant appliquer l'algorithme 2.43 `FermetureConnexeH` à l'automate :



Ce qui nous donne, d'après [MSW98, exemple 3.8],

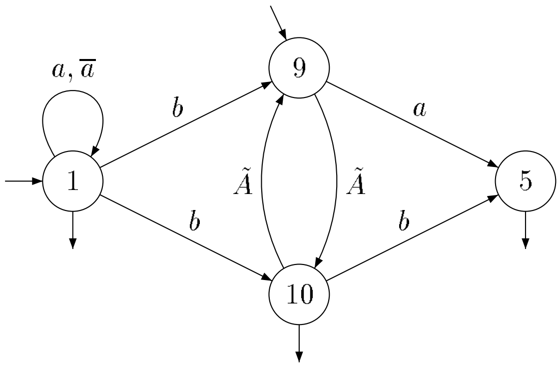


On a alors, en reprenant les notations de l'algorithme,

$$2' = 4' = 9 \quad \text{et} \quad 6' = 8' = 10$$

La troisième composante fortement connexe ne contient aussi qu'un seul état : 5 ($u_5 = \varepsilon$). On doit, pour appliquer l'étape 4 de l'algorithme, calculer l'automate de sous-groupe ayant un nombre minimal d'états reconnaissant $\{\varepsilon\}$ (car G_3 reconnaît $\{\varepsilon\}$). Comme $\{\varepsilon\}$ est un fermé, G_3 est inchangé par application de l'algorithme.

L'automate $\mathcal{A}'$ ci-dessous vérifie alors $L(\mathcal{A}')\pi = \mathrm{Cl}^F_{\mathbf{G}_2}(L(\mathcal{A})\pi)$.



$\triangledown$

En utilisant les résultats de la section 2.3.3 on obtient le résultat suivant.

Corollaire 2.48 *Soit $\mathcal{A}$ un automate sur $\tilde{A}$ à n états. On peut calculer en temps $O(n^5)$ un automate $\mathcal{A}_0$ ayant moins de n états tel que $L(\mathcal{A}_0)\pi$ soit la fermeture pro-p de $L(\mathcal{A})\pi$.*

2.4.4 Calcul dans le cas profini

Dans le cas de la topologie profinie, tous les sous-groupes rationnels du groupe libre sont fermés (voir théorème 2.24), ce qui simplifie grandement les algorithmes.

Proposition 2.49 *Soit $\mathcal{A}$ un automate sur $\tilde{A}$ à n états et m transitions. On peut calculer en temps $O(n+m)$ un automate $\mathcal{A}_0$ à n états tel que*

$$L(\mathcal{A}_0)\pi = \mathrm{Cl}_F(L(\mathcal{A})\pi)$$

Pour cela on utilise l'algorithme `FermetureG` suivant qui s'applique à un automate fini $(Q, \tilde{A}, E, I, F)$.

Algorithme 2.50

1. Calculer les composantes fortement connexes de $\mathcal{A}$.
2. Calculer l'ensemble T des transitions externes.
3. Calculer $E' = \{(p, a, q) \mid (p, \overline{a}, q) \in E \setminus T\}$.
4. Retourner $(Q, \tilde{A}, E \cup E', I, F)$.

$\diamondsuit$

Preuve. Toutes les étapes de cet algorithme se font en temps $O(m+n)$. Sa correction découle du théorème 2.45. $\square$

2.5 Calcul de la clôture d'un langage rationnel du monoïde libre

2.5.1 Clôture d'un langage rationnel donné par un automate

L'algorithme précédent s'adapte très bien, grâce à la proposition 2.31 et au corollaire 1.67, au calcul de la fermeture dans le monoïde libre d'un langage rationnel du monoïde libre, dans le cas bien sûr où la variété $\mathbf{H}$ est close par extension.

Proposition 2.51 *Soient $\mathcal{A}$ un automate à n états sur l'alphabet A et $\mathbf{H}$ une variété de groupes finis close par extension. On peut calculer en temps $O(n\mathsf{f} + n^3)$ un automate $\mathcal{A}_0$ ayant au plus $n\mathsf{f}$ états sur l'alphabet A reconnaissant la fermeture de $L(\mathcal{A})$ pour la topologie pro-$\mathbf{H}$ de A^*.*

Soit $\mathcal{A}$ un automate à n états. On utilise l'algorithme `FermetureHA*` suivant.

Algorithme 2.52
Retourner `ProjectionA*`(`FermetureH`($\mathcal{A}$)).

$\diamondsuit$

PREUVE. La correction de l'algorithme est une conséquence directe de la proposition 2.31 et du corollaire 1.67.

Pour la complexité, calculer `FermetureH`($\mathcal{A}$) se fait en temps $O(n\mathsf{f})$ et l'automate obtenu a au plus n états, d'après le théorème 2.45. Par conséquent, d'après le corollaire 1.67, appliquer `ProjectionA*` à l'automate obtenu se fait en temps $O(n^3)$. D'où une complexité totale en $O(n\mathsf{f} + n^3)$. $\square$

Corollaire 2.53 *Soient $\mathcal{A}$ un automate déterministe à n états sur l'alphabet A, $\mathbf{H}$ une variété de groupes finis close par extension et $u \in A^*$. On peut tester en temps $O((n^3 + n\mathsf{f}(|u| + 1))$ si u appartient à l'intérieur de $L(\mathcal{A})$.*

PREUVE. Soient $\mathcal{A} = (Q, A, E, i, F)$ un automate déterministe à n états sur l'alphabet A, $\mathbf{H}$ une variété de groupes finis close par extension et $u \in A^*$. Le mot u appartient à l'intérieur de $L(\mathcal{A})$ si et seulement s'il n'appartient pas à la fermeture pro-$\mathbf{H}$ du complémentaire de $L(\mathcal{A})$. Or $L(\mathcal{A})^c$ est reconnu par l'automate $\mathcal{A}' = (Q, A, E, i, Q \setminus F)$. Soit $\mathcal{A}_0$ l'automate obtenu en calculant `FermetureHA*`($\mathcal{A}'$). On peut le calculer en temps $O(n^3 + n\mathsf{f})$ et il a au plus n états. Par conséquent, tester si $u \in L(\mathcal{A}_0)$ se fait en temps $O(n\mathsf{f}|u|)$. $\square$

Il faut remarquer ici que l'on n'a pas calculé d'automate reconnaissant l'intérieur de $L(\mathcal{A})$ mais un automate non déterministe reconnaissant son complémentaire.

Il est très facile de calculer le complémentaire d'un langage donné par un automate fini déterministe, contrairement au cas où il est donné par une expression rationnelle. Le corollaire 2.53 ci-dessus est un exemple d'utilisation de cette propriété. Cependant, cette supériorité algorithmique des automates sur les expressions rationnelles s'exprime plus pleinement par le théorème suivant.

Théorème 2.54 *Soient $\mathcal{A}$ un automate déterministe à n états et* $\mathbf{H}$ *une variété de groupes finis close par extension. On peut tester en temps $O(n\mathsf{f}+n^3)$ si $L(\mathcal{A})$ est fermé pour la topologie pro-*$\mathbf{H}$.

On utilise pour cela l'algorithme <u>`TesteFerméHA*`</u> suivant.

Algorithme 2.55

1. $\mathcal{A}'$ =<u>`FermetureHA*`</u>$((\mathcal{A}))$.
2. Tester si $L(\mathcal{A}') \subseteq L(\mathcal{A})$.

$\diamond$

Preuve. L'étape 1 se fait en temps $O(n\mathsf{f} + n^3)$ d'après la proposition 2.51. Comme l'automate $\mathcal{A}'$ a au plus n états et $O(n^2)$ transitions, l'étape 2 se fait en temps $O(n^3)$. Par conséquent l'algorithme se termine en temps $O(n\mathsf{f}+n^3)$.

La correction de l'algorithme découle du fait qu'un langage est fermé si et seulement s'il est inclus dans sa fermeture. $\square$

2.5.2 Calcul dans le cas profini

Comme dans le groupe libre, le calcul de la fermeture profinie d'un langage rationnel du monoïde libre est plus simple que dans le cas général : on peut adapter l'algorithme 2.52 afin de calculer la fermeture profinie d'un langage rationnel de A^*.

Théorème 2.56 *Soit $\mathcal{A}$ un automate à n états sur l'alphabet A. On peut calculer en temps $O(n^3)$ un automate $\mathcal{A}_0$ sur A tel que*

(1) $L(\mathcal{A}_0) = \mathrm{Cl}_{A^*}(L(\mathcal{A}))$.

(2) $\mathcal{A}_0$ a au plus n états.

(3) Si p, q et r appartiennent à la même composante fortement connexe et s'il existe $a \in A$ tel que $p \cdot a = q \cdot a = r$, alors $p = q$.

(4) Si p, q et r appartiennent à la même composante fortement connexe et s'il existe $a \in A$ tel que $\{q, r\} \subseteq p \cdot a$, alors $q = r$.

On utilise pour cela l'algorithme `FermetureGA*` suivant s'appliquant à un automate $\mathcal{A}$ et qui fait appel à l'algorithme 2.50 `FermetureG` ainsi qu'à l'algorithme `ProjectionA*` décrit à la proposition 1.70.

Algorithme 2.57

1. Soit $\mathcal{A}_1 =$`FermetureG`$(\mathcal{A})$.
2. Soit $\mathcal{A}_0 =$`ProjectionA*`$(\mathcal{A}_1)$.
3. Tant qu'il y a des états p, q, r appartenant à la même composante fortement connexe de $\mathcal{A}_0$ et une lettre $a \in A$ telle que $p \cdot a = q \cdot a = r$ ou telle que $\{p, q\} \in r \cdot a$, fusionner les états p et q.
4. Retourner $\mathcal{A}_0$.

$\diamondsuit$

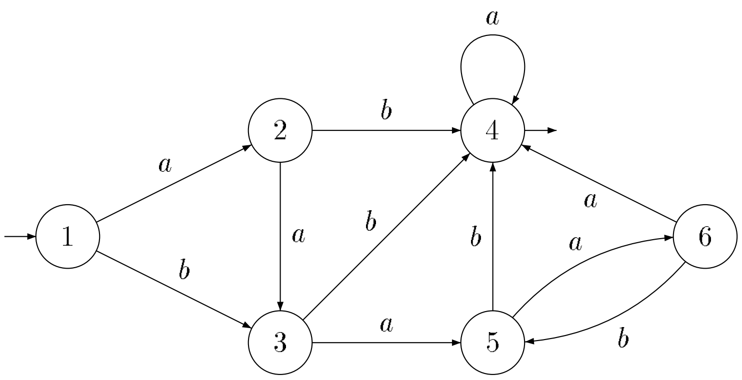
Preuve. Il est à noter que dans l'algorithme ci-dessus, on itère l'étape 3 tant qu'il y a des configurations d'un des deux types, sachant qu'une fusion peut en générer de nouvelles. Mais comme à chaque fusion le nombre d'états de $\mathcal{A}_0$ diminue, on est sûr que l'algorithme s'arrête.

A la fin de l'étape 2, l'automate $\mathcal{A}_0$ reconnaît, d'après la proposition 2.49, le langage $\mathrm{Cl}_{A^*}(L(\mathcal{A}))$. Supposons que l'on ait alors p, q, r appartenant à la même composante fortement connexe de $\mathcal{A}_0$ tels qu'il existe une lettre $a \in A$ telle que $p \cdot a = q \cdot a = r$. Dans ce cas là, d'après l'algorithme 2.50 `FermetureG`, on a aussi $\{p, q\} \subseteq r \cdot \overline{a}$. On peut donc aller de p à q en lisant $a\overline{a}$. Par conséquent, on peut rajouter une transition (p, ε, q) sans modifier l'image du langage dans le groupe libre. De même on peut passer de q à p en lisant $a\overline{a}$ et donc ajouter une transition (q, ε, p). Il en résulte que l'on peut fusionner les états p et q. On montre de même que fusionner deux états lorsque l'on a l'autre configuration ne modifie pas le langage reconnu. Par conséquent, par récurrence, il s'en suit que l'automate $\mathcal{A}_0$ retourné à l'étape 4 reconnaît exactement $\mathrm{Cl}_{A^*}(L(\mathcal{A}))$.

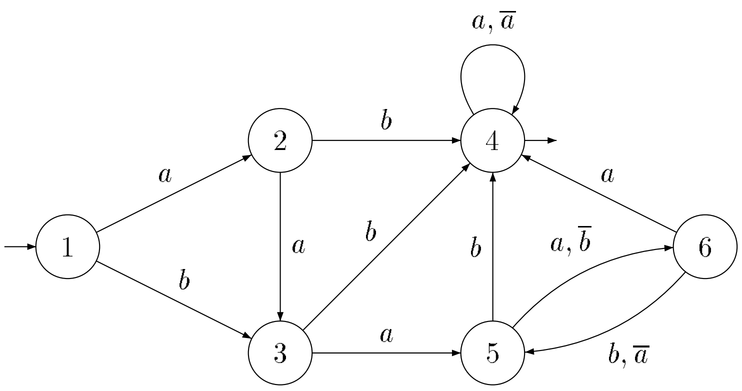
De plus, on vient de montrer que $\mathcal{A}_0$ a au plus n états.

Chaque étape de cet algorithme pouvant se réaliser en temps $O(n^3)$, la complexité totale est en $O(n^3)$. $\square$

Exemple 2.58 Calculons la fermeture profinie du langage donnée par l'automate suivant :

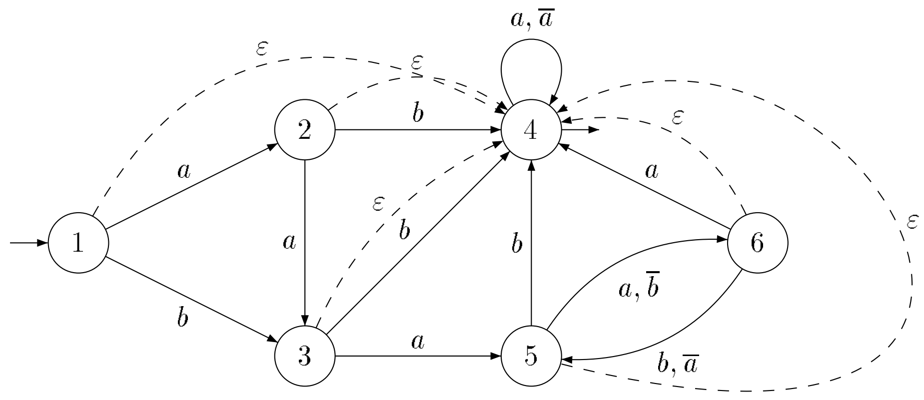


On cherche les composantes fortement connexes de l'automate. Il y en a 5 : $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$ et $\{5, 6\}$. Pour chaque transition interne à une composante fortement connexe, on construit la transition duale. On obtient l'automate :

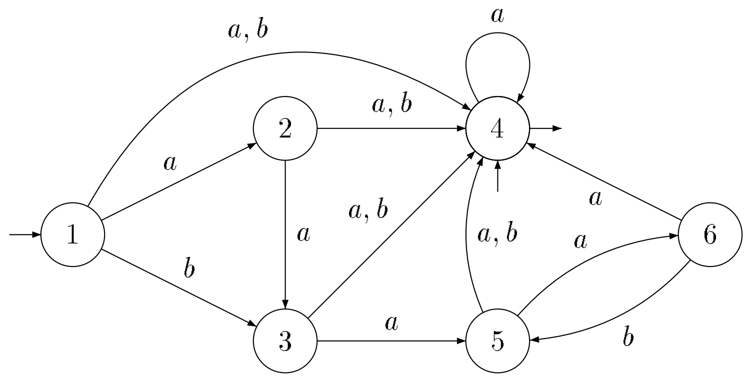


Chaque fois que l'on a $p \cdot u = q$ avec $u\pi = 1$, on construit la transition (p, ε, q). On vérifie qu'ici on construit seulement cinq ε-transitions de cette façon :

-$(6, \varepsilon, 4)$ car $6 \cdot a\overline{a} = 4$.
-$(5, \varepsilon, 4)$ car $5 \cdot a^2\overline{a^2} = 4$.
-$(3, \varepsilon, 4)$ car $3 \cdot a^3\overline{a^3} = 4$.
-$(2, \varepsilon, 4)$ car $2 \cdot a^4\overline{a^4} = 4$.
-$(1, \varepsilon, 4)$ car $2 \cdot a^5\overline{a^5} = 4$.



Ce qui, après élimination des ε-transitions, nous donne :



$\triangledown$

Finalement, on peut aussi obtenir un nouvel algorithme pour tester si un langage rationnel donné par un automate déterministe est fermé pour la topologie profinie de A^*. L'intérêt de cet algorithme sera discuté au chapitre 4.

Théorème 2.59 *Soit $\mathcal{A}$ un automate déterministe à n états sur l'alphabet A. On peut tester en temps $O(n^3)$ si $L(\mathcal{A})$ est fermé pour la topologie profinie de A^*.*

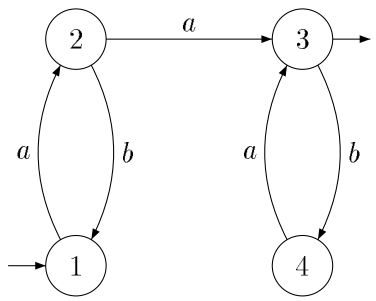
On utilise pour cela l'algorithme TesteFerméGA* suivant, qui est une traduction directe de TesteFerméHA* et dont la justification en est identique. Cet algorithme s'applique sur un automate déterministe $\mathcal{A}$.

Algorithme 2.60

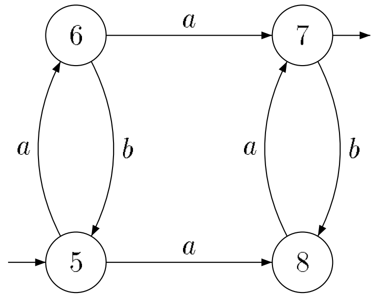
1. Calculer $\mathcal{A}_0$ tel que $L(\mathcal{A}_0) = \mathrm{Cl}_{A^*}(L(\mathcal{A}))$
2. Tester si $L(\mathcal{A}) \subseteq L(\mathcal{A}_0)$.

$\diamondsuit$

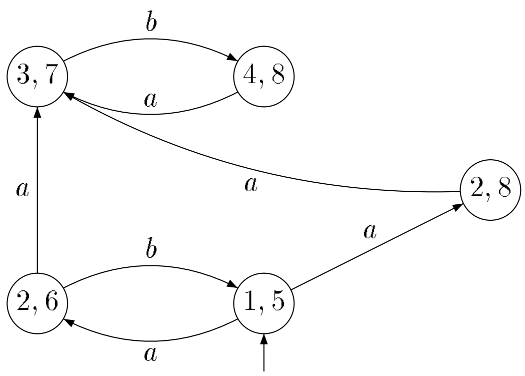
Exemple 2.61 Considérons le langage de A^* donné par l'automate $\mathcal{A}$:



On lui applique l'algorithme 2.57 FermetureGA* et on obtient, après simplification des ε-transitions (ici il n'y en a qu'une entre les états 2 et 4), l'automate $\mathcal{A}_0$ suivant :



On calcule le produit des automates $\mathcal{A}$ et $\mathcal{A}_0$ pour obtenir un automate reconnaissant $L(\mathcal{A})^c \cap L(\mathcal{A}_0)$ et on obtient l'automate suivant (en ne représentant que la partie accessible) :



Cet automate reconnaissant le langage vide (il n'y a pas d'état final), il en résulte que $L(\mathcal{A})$ est fermé pour la topologie profinie.

$\triangledown$

2.5.3 Calcul dans le cas le pire

L'algorithme FermetureGA* calcule la fermeture d'un langage rationnel donné par un automate fini et retourne un automate qui est, en général, non déterministe (même lorsque l'automate de départ est déterministe). Donc, si l'on souhaite obtenir un automate déterministe reconnaissant la fermeture profinie d'un langage, on peut appliquer cet algorithme, puis déterminiser l'automate obtenu. Cependant, il est possible que la déterminisation demande un temps de calcul exponentiel. On peut alors se demander si, par un autre moyen, on pourrait calculer directement un automate déterministe reconnaissant la fermeture d'un langage en temps polynomial. Nous allons montrer ici que cela est impossible, en montrant qu'il existe une suite de langages K_n vérifiant :

(1) l'automate minimal de K_n possède $3n$ états,

(2) l'automate minimal de la fermeture profinie de K_n possède $\Omega(4^n/\sqrt{n})$ états.

Pour montrer ce résultat, nous avons besoin du lemme suivant.

Lemme 2.62 *Soient a et b les deux permutations de $Q_n = \{1, \dots, 2n\}$ (avec $n \geq 2$) suivantes :*

$$a = (2, 3, \cdots, 2n, 1) \quad b = (1, 2)$$

Soient aussi $E_n = \{(i, a, ia) \mid i \in Q\} \cup \{(j, a, ja) \mid i \in Q_n\}$ *et l'automate*

$$\mathcal{A}_n = (Q_n, \{a, b\}, E_n, \{1, \ldots, n\}, 2n)$$

L'automate minimal de $L(\mathcal{A}_n)$ *possède* $(2n)!/(n!)^2$ *états.*

Preuve. Commençons par remarquer que a et b engendrent tout le groupe des permutations de $\{1, \ldots, 2n\}$. On étend alors la notation σ_u à tout mot de $\{a, b\}^*$ en posant $\sigma_{ua} = \sigma_u a = a \circ \sigma_u$ et $\sigma_{ub} = \sigma_u b = b \circ \sigma_u$. La remarque ci-dessus permet de dire que pour toute permutation σ de Q, il existe un mot u tel que $\sigma = \sigma_u$.

Considérons l'automate $\mathcal{A}'_n$ sur l'alphabet $\{a, b\}$ que l'on obtient en appliquant l'algorithme classique de déterminisation.

- L'ensemble des états de $\mathcal{A}'_n$ est constitué de toutes les images de $\{1, \ldots, n\}$ par toutes les permutations de Q. On obtient ainsi $(2n)!/(n!)^2$ états, chacun étant une partie de Q à n éléments.
- On a une transition entre deux parties R et S étiquetée par a [resp. par b], si et seulement si $a(R) = S$ [resp. $b(R) = S$].
- L'état initial de $\mathcal{A}'_n$ est $\{1, \ldots, n\}$.
- Les états finaux de $\mathcal{A}'_n$ sont toutes les parties contenant $2n$.

Par construction, l'automate $\mathcal{A}'_n$ reconnaît $L(\mathcal{A}_n)$.

Montrons maintenant que l'automate $\mathcal{A}'_n$ est minimal. Considérons pour cela deux états distincts $R = \{a_1, \cdots, a_n\}$ et $S = \{b_1, \cdots, b_n\}$ de $\mathcal{A}'_n$ et prouvons que R et S ne sont pas équivalents (voir définition 1.53). Pour cela considérons les cas suivants, selon que R ou S sont finaux ou initiaux.

- Si R est final et S n'est pas final, alors $R \cdot \varepsilon$ est final mais pas $S \cdot \varepsilon$. Donc R et S ne sont pas équivalents. Le même raisonnement est valable si S est final mais pas R.
- Si R et S sont tous deux finaux ou tous deux non finaux, alors il existe $a_k \in R$ tel que $a_k \neq 2n$ et $a_k \notin S$. Considérons un mot u tel que $\sigma_u = (a_k, 2n)$. Alors l'état $R \cdot u$ est final car il contient $2n$. En revanche $S \cdot u$ ne contient pas $2n$ car $a_k \notin S$.

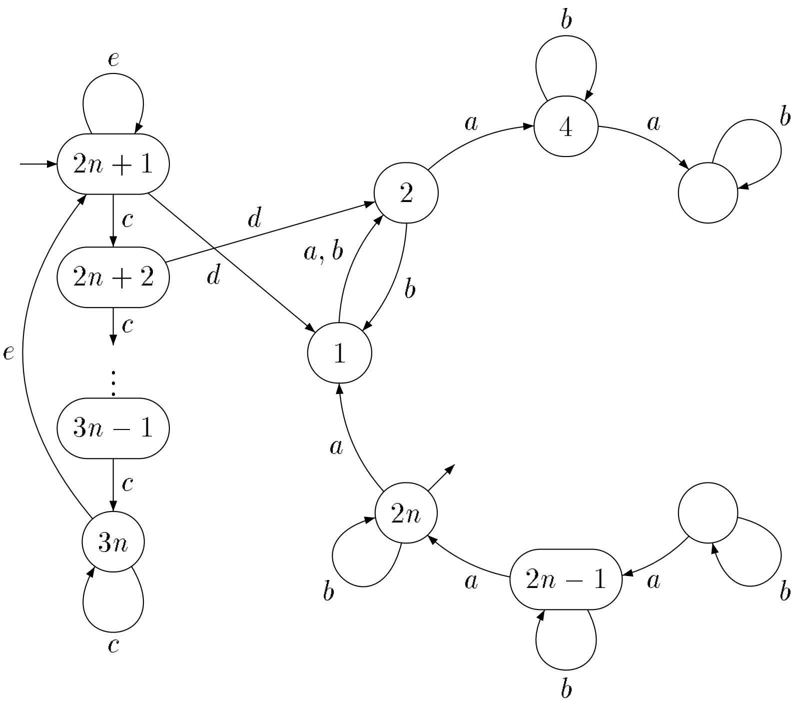
On a ainsi montré que l'automate minimal de $L(\mathcal{A}_n)$ possède $(2n)!/(n!)^2$ états. □

Avec les notations du lemme précédent, on considère l'automate $\mathcal{B}_n$ suivant.

- L'ensemble des états de $\mathcal{B}_n$ est $\{1, \ldots, 3n\}$.
- L'alphabet de $\mathcal{B}_n$ est $\{a, b, c, d, e\}$.

- Les transitions de $\mathcal{B}_n$ sont tous les triplets suivants:
  - $(2n+k, c, 2n+k+1)$ pour $1 \leq k \leq n-1$
  - $(2n+k, d, k)$ pour $1 \leq k \leq n-1$
  - (k, a, ka) et (k, b, kb) pour $1 \leq k \leq 2n$.
  - $(3n, c, 3n)$, $(2n+1, e, 2n+1)$ et $(3n, e, 2n+1)$.
- L'unique état initial de $\mathcal{B}_n$ est $2n+1$.
- L'unique état final de $\mathcal{B}_n$ est $2n$.

Graphiquement on obtient :



Proposition 2.63 *Avec les notations ci-dessus,* $m(L(\mathcal{B}_n)) = 3n$ *et,*

$$m(\mathrm{Cl}_{A^*}(L(\mathcal{B}_n))) = \Omega(4^n/\sqrt{n})$$

Preuve. Montrons que l'automate $\mathcal{B}_n$ est minimal. Considérons deux états distincts k_1 et k_2 de cet automate.

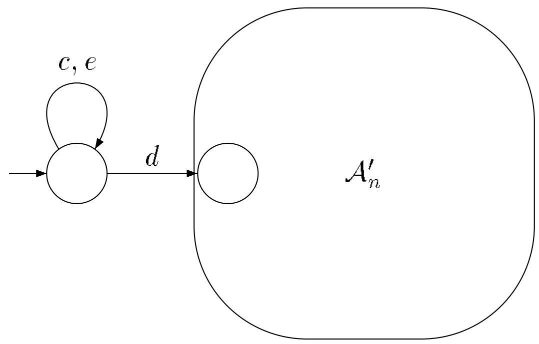
- Si $1 \leq k_1 \leq 2n$ et $1 \leq k_2 \leq 2n$, alors $k_1 \cdot a^{2n-k_1}$ est final alors que $k_2 \cdot a^{2n-k_1}$ ne l'est pas. Les deux états ne sont donc pas équivalents.

- Si $1 \le k_1 \le 2n$ et $2n+1 \le k_2 \le 3n$, alors $k_1 \cdot a^{2n-k_1}$ est final alors que $k_2 \cdot a^{2n-k_1}$ ne l'est pas.
- Si $1 \le k_2 \le 2n$ et $2n+1 \le k_1 \le 3n$, on conclut de façon identique au cas précédent.
- Si $2n+1 \le k_2 \le 3n$ et $2n+1 \le k_1 \le 3n$, alors $k_1 \cdot da^{4n-k_1}$ est final alors que $k_2 \cdot da^{4n-k_1}$ ne l'est pas. Les deux états ne sont donc pas équivalents.

Il en résulte que $m(L(\mathcal{B}_n)) = 3n$.

Si l'on applique l'algorithme 2.57 FermetureGA* à $\mathcal{B}_n$, on fusionne exactement tous les états entre $2n+1$ et $3n$. En effet, après avoir ajouté dans les composantes fortement connexes de $\mathcal{B}_n$ toutes les transitions duales, on doit ajouter des ε-transitions entre deux états si et seulement si on peut les relier par un chemin qui a pour étiquette un mot de $1\pi^{-1}$. Comme aucune transition étiquetée par d n'est interne à une composante fortement connexe, il n'y a aucune transition étiquetée par $\overline{d}$. Il en résulte que toutes les ε-transitions ajoutées sont internes aux composantes fortement connexes. Or, la composante fortement connexe $\{1, \dots, 2n\}$, munie des transitions qui lui sont internes, forme un automate à groupe : on n'y ajoute donc aucune ε-transition. En revanche, dans la composante fortement connexe $\{2n+1, \dots, 3n\}$, on peut passer de l'état k_1 à l'état k_2 en lisant $c^{3n-k_1}(\overline{c})^{n-k_1+k_2}e\overline{e}c^{k_2-2n}$, dont l'image par π vaut 1. On peut donc fusionner tous les états de la composante fortement connexe $\{2n+1, \dots, 3n\}$.

La fermeture profinie de $L(\mathcal{B}_n)$ est donc le langage $\{c, e\}^* dL(\mathcal{A}_n)$, qui est reconnu par l'automate suivant :



En utilisant le lemme 2.62, on conclut qu'il est minimal. Il en résulte que

$$m(\mathrm{Cl}_{A^*}(L(\mathcal{B}_n))) = 1 + \frac{(2n)!}{(n!)^2}$$

D'où, en utilisant la formule de Stirling,

$$m(\mathrm{Cl}_{A^*}(L(\mathcal{B}_n))) = \Omega(4^n/\sqrt{n})$$

□

On vient de montrer que, dans le cas le pire, le calcul d'un automate déterministe reconnaissant la fermeture d'un langage donné nécessite un espace mémoire et un temps de calcul exponentiels.

2.6 Propriétés topologiques des parties rationnelles

Cette partie est consacrée aux propriétés topologiques des parties rationnelles. On a déjà vu, dans la partie 2.4.2, qu'un lien étroit unissait la topologie profinie et les parties rationnelles de $F(A)$ et de A^*. Nous développons ici ces aspects en nous intéressant plus particulièrement aux aspects topologiques des langages rationnels de A^*.

2.6.1 Langages denses

En adaptant un résultat de P. Ribes et P.A. Zalesskii [RZ94] et en utilisant des arguments de densité (entre autres), S. Margolis, M. Sapir et P. Weil [MSW98] ont développé un algorithme en temps polynomial pour calculer la fermeture pro-p d'un sous-groupe du groupe libre (voir section 2.3.3). Dans cette section, nous étudions quelques propriétés des langages rationnels denses.

Proposition 2.64 *Soient L et R deux langages non vides de A^* et* $\mathbf{H}$ *une variété de groupes finis. Le langage LA^*R est dense pour la topologie pro-*$\mathbf{H}$*.*

Preuve. Soient L et R deux langages rationnels non vide de A^*. Soient $u \in L$ et $v \in R$. Pour tout mot $w \in A^*$ et tout entier k, le mot $uu^kwv^kv \in LA^*R$. Or, d'après la proposition 2.7,

$$\lim_{n\to+\infty} uu^{n!-1}wv^{n!-1}v = w$$

et, par conséquent, LA^*R est dense pour la topologie pro-$\mathbf{H}$. □

La réciproque n'est pas vraie. En effet, si on applique l'algorithme 2.57 au langage

$$K = [(b \cup ab)(ab)^*b]^*$$

on voit qu'il est dense pour la topologie profinie. Or, comme aucun mot de K ne contient le facteur aa, il en résulte que K ne contient aucun langage du type LA^*R avec L et R non vides.

Proposition 2.65 *Soient* $\mathbf{H}$ *une variété de groupes finis close par extension et* L *un langage rationnel fermé et d'intérieur vide pour la topologie pro-*$\mathbf{H}$*. L'automate minimal de* L *n'est pas complet.*

Preuve. Supposons que l'automate minimal $\mathcal{A}$ de L soit complet. On dira que deux composantes fortement connexes S_1 et S_2 de $\mathcal{A}$ vérifient la relation $S_1 \leq S_2$ si et seulement s'il existe un chemin allant d'un état de S_1 à un état de S_2. Soit $S = (Q_S, E_S)$ une composante fortement connexe de $\mathcal{A}$ maximale pour cet ordre. Comme $\mathcal{A}$ est complet et S est maximale, le graphe de S est complet. Donc, d'après la proposition 2.22, chaque lettre de A induit une application bijective de l'ensemble des états de S dans lui-même. Comme $\mathcal{A}$ est émondé et que S est maximale, Q_S contient au moins un état final q. Soit, enfin, $u \in A^*$ tel que $i \cdot u = q$ où i est l'état initial de $\mathcal{A}$. On pose $\mathcal{A}_0 = (Q_S, A, E_S, q, q)$, qui est un automate à groupe. De plus, $uL(\mathcal{A}_0) \subseteq L$ et $L(\mathcal{A}_0)$ est non vide.

Or, en posant $\mathcal{A}_1 = (Q_S, A, E_S, r, q)$, où r est l'unique état de Q_S vérifiant $r \cdot u = q$, et en appliquant l'algorithme 2.52 `FermetureHA*` à $uL(\mathcal{A}_0)$, on remarque que

$$\mathrm{Cl}^{\mathbf{H}}_{A^*}(L(\mathcal{A}_1)) \subseteq \mathrm{Cl}^{\mathbf{H}}_{A^*}(uL(\mathcal{A}_0)) \subseteq L$$

car L est fermé. Mais, d'après la proposition 4.5, $K = \mathrm{Cl}^{\mathbf{H}}_{A^*}(L(\mathcal{A}_1))$ est un langage à groupe dont le monoïde syntaxique est dans $\mathbf{H}$. Donc K est aussi ouvert (par la proposition 2.17). Il en résulte que L contient un ouvert non vide, ce qui contredit le fait que L^c est dense. □

Corollaire 2.66 *Soit* $\mathbf{H}$ *une variété de groupes finis close par extension et* L *un langage rationnel ouvert dense pour la topologie pro-*$\mathbf{H}$*. Il existe un mot* $u \in A^*$ *tel que* $uA^* \subseteq L$*.*

Preuve. Soit $\mathbf{H}$ une variété de groupes finis close par extension et L un langage rationnel ouvert dense pour la topologie pro-$\mathbf{H}$. D'après la proposition 2.65, l'automate minimal $\mathcal{A} = (Q, A, E, i, F)$ de L^c n'est pas complet. Complétons $\mathcal{A}$. Pour cela, on pose

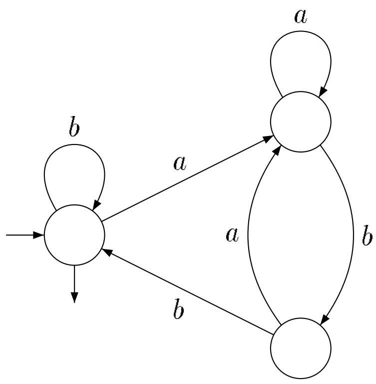
- $Q' = Q \cup \{p\}$ tel que $p \notin Q$.
- $E' = E \cup \{(q, a, p) \mid q \in Q' \text{ et } q \cdot a = \emptyset\}$.

L'automate $\mathcal{A}' = (Q', A, E', i, F)$ est complet déterministe et reconnaît L^c. Par conséquent, l'automate $(Q', A, E', i, Q' \setminus F)$ accepte L. En particulier, comme $p \notin Q$, $p \notin F$. Par conséquent, le langage reconnu par (Q', a, E', i, p) est non vide. Pour tout mot u de ce langage on a donc

$$uA^* \subseteq L$$

$\square$

Dans la proposition ci-dessus, l'hypothèse ouvert est importante. Considérons par exemple le langage L donné par l'automate ci-dessous :



Par une application directe de l'algorithme FermetureGA*, on voit que L est dense pour la topologie profinie (de même que son complémentaire, ce qui prouve que L n'est pas fermé). Or aucun mot de L ne finit par un a, donc L ne contient aucun langage du type uA^*.

De plus, dans le corollaire 2.66, l'hypothèse de rationnalité est elle aussi importante. En effet, si l'on considère le sous-langage $L = 1\pi^{-1}$ de $\tilde{A}^*$, on va montrer, à la proposition 2.82, que son complémentaire est un ouvert dense. Or, pour tout mot u, $u\overline{u} \in u\tilde{A}^* \cap L$, donc L^c ne contient aucun langage du type $u\tilde{A}^*$.

2.6.2 Points d'accumulation

Il est montré dans [BCP89] que tout mot de A^* est limite, pour la topologie pro-2, d'une sous-suite convergente de la suite de Thue-Morse. Nous étudions dans cette partie l'ensemble des points d'accumulation d'un langage rationnel de A^*.

Pour toute variété de groupe fini $\mathbf{H}$, les langages à groupes dont le monoïde syntaxique appartient à $\mathbf{H}$ forment une base d'ouverts pour la topologie pro-$\mathbf{H}$. Par conséquent, si l'on se donne un langage L (pas nécessairement rationnel) et un mot u, alors u est un point d'accumulation de L si et seulement si pour tout langage à groupe K dont le monoïde syntaxique appartient à $\mathbf{H}$ et contenant u, le langage $K \cap L$ n'est pas un langage fini.

On peut maintenant énoncer le résultat principal.

Théorème 2.67 *Soit* $\mathbf{H}$ *une variété de groupes finis close par extension et* L *un langage rationnel de* A^*. *L'ensemble des points d'accumulation de* L *pour la topologie pro-*$\mathbf{H}$ *est un langage rationnel.*

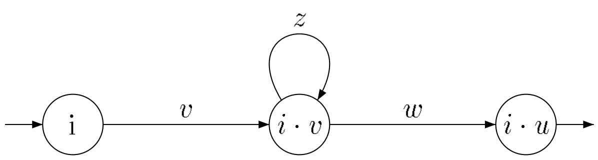
PREUVE. On note R^d l'ensemble dérivé, pour la topologie pro-$\mathbf{H}$, (c'est-à-dire l'ensemble des points d'accumulation) de tout langage R.

Si L est fini, alors $L^d = \emptyset$ est rationnel.

Supposons maintenant que L n'est pas fini. Soit $\mathcal{A} = (Q, A, E, i, F)$ un automate déterministe reconnaissant L. On appelle Q_f le sous-ensemble de Q constitué des états de $\mathcal{A}$ qui ne sont pas sur une boucle. Autrement dit, $q \in Q_f$ si et seulement s'il n'y a pas de chemin dans $\mathcal{A}$ passant deux fois par q. Soit L_f le langage des mots qui sont l'étiquette d'un chemin de $\mathcal{A}$ qui n'utilise que des états de Q_f. Il est clair que L_f est fini. On définit maintenant le langage L_∞ de tous les mots qui sont l'étiquette d'un chemin réussi dans $\mathcal{A}$ utilisant au moins un état hors de Q_f. Comme $\mathcal{A}$ est déterministe :

$$L_\infty = L \setminus L_f$$

Prouvons que $L^d = \mathrm{Cl}^{\mathbf{H}}_{A^*}(L_\infty)$. Soient $u \in L_\infty$ et K un langage à groupe contenant u et tel que $M(K) \in \mathbf{H}$. Le mot u n'appartient pas à L_f, donc il existe v et w dans A^* tels que $u = vw$ et $i \cdot v \notin Q_f$. Par conséquent, il existe un mot z non vide tel que $i \cdot vz = i \cdot v$. Ce qui s'illustre par la figure suivante :



Il en résulte que, pour tout $k \geq 0$,

$$vz^{k!}w \in L_\infty$$

Par conséquent, en utilisant la proposition 2.7, u est la limite d'une suite infini d'éléments de L. Donc $u \in L^d$, ce qui prouve que $L_\infty \subseteq L^d$ et, comme L^d est un fermé pro-$\mathbf{H}$,

$$\mathrm{Cl}^{\mathbf{H}}_{A^*}(L_\infty) \subseteq L^d$$

Passons à l'inclusion opposée. Comme $L_\infty \cup L_f = L$, il en découle que

$$L^d_\infty \cup L^d_f = L^d$$

Or L_f est fini, donc $L_f^d = \emptyset$. Par conséquent,

$$L^d = L_\infty^d \subseteq \mathrm{Cl}_{A^*}^{\mathbf{H}}(L_\infty)$$

Par conséquent $L^d = \mathrm{Cl}_{A^*}^{\mathbf{H}}(L_\infty)$. Or L_∞ est rationnel, donc sa fermeture pro-**H** aussi (car **H** est fermée par extension) ; L^d est donc un langage rationnel. □

Par ailleurs une partie non rationnelle peut avoir un ensemble de points d'accumulation rationnel. En effet, si $u \in A^+$, alors le sous-ensemble $\{u^{k!} \mid k \in \mathbb{N}\}$ n'est pas rationnel et pour toute topologie pro-**H**, son ensemble dérivé est réduit au mot vide (proposition 2.7).

La preuve du théorème 2.67 est constructive, ce qui permet de déduire un algorithme de calcul de l'ensemble dérivé d'un langage rationnel, à condition de savoir calculer la fermeture d'un sous-groupe rationnel (voir page 78).

Corollaire 2.68 *Soit* **H** *une variété close par extension. Soit L un langage rationnel donné par un automate déterministe à n états. On suppose que l'on dispose d'un algorithme qui, étant donné un sous-groupe H de $F(A)$ donné par son automate $\mathcal{A}_H$ à n états, calcule l'automate de sous-groupe $\mathcal{A}_K$ de sa fermeture K (pour la topologie pro-*__H__*). On peut calculer un automate reconnaissant l'ensemble dérivé de L pour la topologie pro-*__H__* en temps $O(n^2\mathsf{f}+n^6)$.*

On utilise pour cela l'algorithme DériveH suivant qui s'applique à un automate déterministe $\mathcal{A} = (Q, A, E, i, F)$ à n états.

Algorithme 2.69

1. Calculer les composantes fortement connexes de $\mathcal{A}$.
2. Calculer le sous-ensemble Q_f des états q qui sont seuls dans leur composante fortement connexe et tels qu'il n'existe pas de transition du type (q, a, q).
3. Calculer l'automate $\mathcal{A}_f = (Q_1, A, E_1, i_1, F_1)$ obtenu en en émondant l'automate $(Q_f, A, E \cap Q_f \times A \times Q_f, i \cap Q_f, F \cap Q_f)$ et en le rendant complet.
4. Calculer l'automate $\mathcal{A}_0$ obtenu en faisant le produit de $\mathcal{A}$ par $(Q_1, , A, E_1, i_1, Q_1 \setminus F_1)$.
5. Calculer l'automate $\mathcal{A}_\infty$ obtenu en minimisant $\mathcal{A}_0$.
6. Retourner `FermetureHA*`$(\mathcal{A}_\infty)$.

◊

PREUVE. La correction de cet algorithme découle de la preuve du théorème 2.67 : par construction l'automate $\mathcal{A}_f$ reconnaît L_f et les automate $\mathcal{A}_0$ et $\mathcal{A}_\infty$ reconnaissent L_∞. L'algorithme retourne bien $\mathrm{Cl}^{\mathbf{H}}_{A^*}(L_\infty) = L^d$.

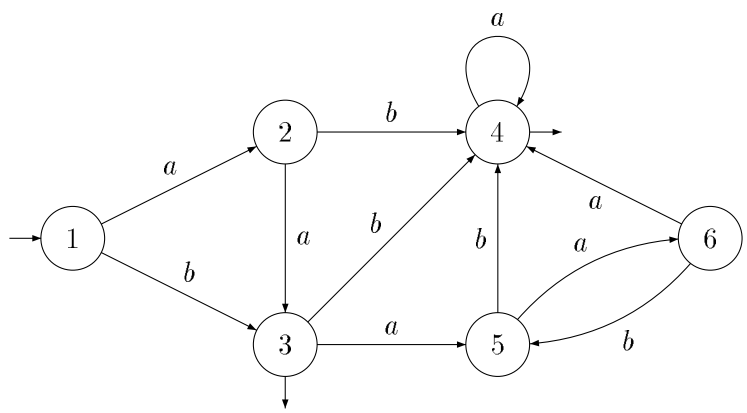
1. Cette étape peut être réalisée en temps $O(n)$.
2. Comme $\mathcal{A}$ est déterministe, il a au plus $n|A|$ transitions. Il faut donc ici un temps en $O(n)$.
3. Cette étape peut être réalisée en temps $O(n)$. De plus l'automate $\mathcal{A}_f$ est déterministe et a au plus $n+1$ états.
4. Cette étape s'exécute en temps $O(n^2)$.
5. Cette étape demande un temps $O(n^2 \ln n)$ (voir section 1.51).
6. On peut faire cette étape en temps $O(n^2\mathsf{f} + n^6)$, d'après la proposition 2.51.

□

Il faut remarquer qu'il n'est pas nécessaire d'exécuter l'étape 5. La faire n'améliore pas la complexité dans le pire des cas. Cependant, l'étape limitante en temps étant l'étape 6, il est avantageux d'essayer d'obtenir un automate ayant un nombre minimal d'états.

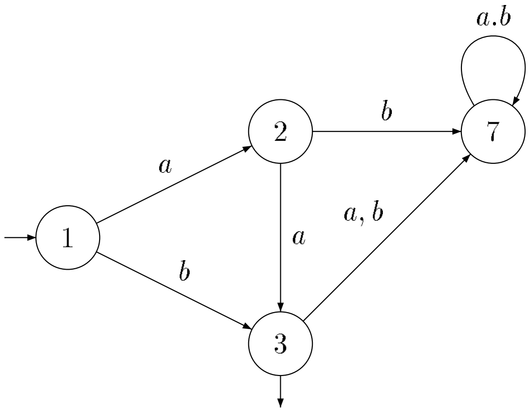
On peut remarquer aussi que dans le cas profini la complexité dans le pire des cas est en $O(n^6)$.

Exemple 2.70 Calculons l'ensemble dérivé, pour la topologie profinie, du langage rationnel L donné par l'automate $\mathcal{A}$ suivant :

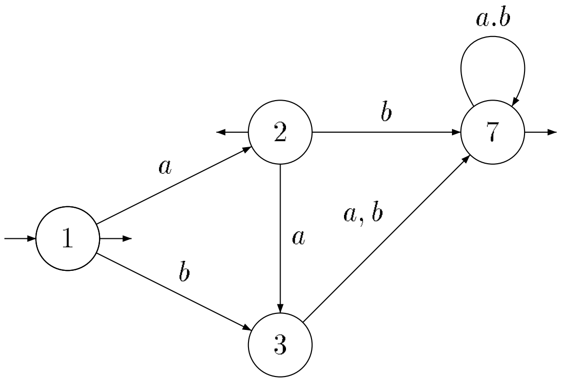


Les états 1,2 et 3 ne sont pas sur une boucle.

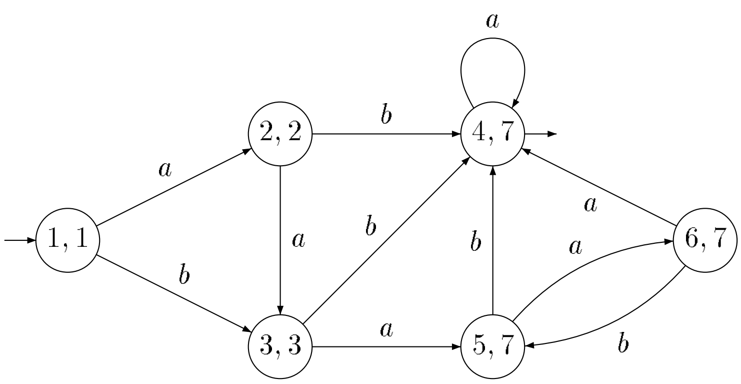
L'automate $\mathcal{A}_f$ est donc le suivant :



On doit donc effectuer le produit de $\mathcal{A}$ par l'automate suivant :

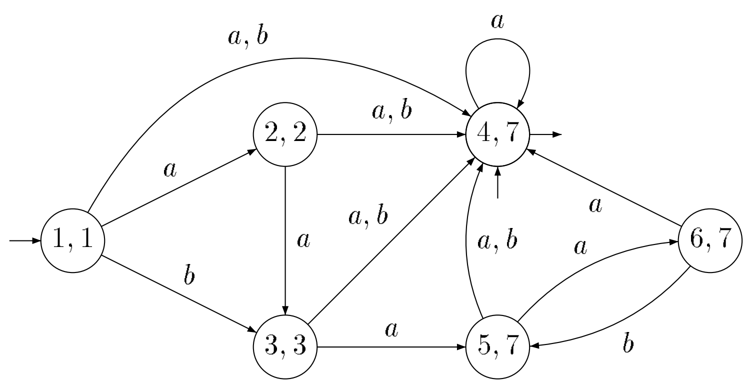


On obtient alors l'automate suivant $\mathcal{A}_0$ suivant :



L'automate $\mathcal{A}_0$ étant minimal, on a $\mathcal{A}_\infty = \mathcal{A}_0$.

La fermeture profinie de L_∞ est donnée par l'automate non déterministe $\mathcal{A}'$ suivant (voir exemple 2.58) :



L'automate ci-dessus reconnaît donc L^d.

∇

On peut déduire quelques propriétés intéressantes des points isolés d'un langage rationnel, découlant de la preuve du théorème 2.67.

Corollaire 2.71 *Soient* **H** *une variété de groupes finis close par extension et L un langage rationnel de A^*. L'ensemble des points isolés de L pour la topologie pro-***H** *est fini. De plus, si L est infini, alors son ensemble dérivé pour la topologie pro-***H** *est aussi infini.*

PREUVE. En effet, en prouvant le théorème 2.67, on a montré que l'ensemble des points isolés de L est un sous-ensemble de L_f ; il est donc fini. On a aussi prouvé que si L est infini, alors L_∞ l'est aussi et, comme $L_\infty \subseteq L^d$, L^d est infini. $\square$

On définit une *suite rationnelle* comme étant une suite (u_n) de mots de A^* telle que $\{u_n \mid n \geq 0\}$ soit rationnel. suite rationnelle

Corollaire 2.72 *Soient* **H** *une variété de groupes finis close par extension et (u_n) une suite rationnelle de A^*. Si (u_n) a un nombre fini de points d'accumulation, alors (u_n) prend un nombre fini de valeurs.*

PREUVE. D'après le corollaire 2.71, si (u_n) est une suite rationnelle qui prend un nombre fini de valeurs, alors l'ensemble $\{u_n \mid n \geq 0\}$ a un nombre fini de points d'accumulation et, par conséquent, est fini. $\square$

Rappelons que le monoïde libre, muni de la distance pro-$\mathbf{H}$, n'est pas complet [Pin91].

Corollaire 2.73 *Soient* $\mathbf{H}$ *une variété de groupes finis close par extension et* (u_n) *une suite rationnelle de Cauchy pour la topologie pro-*$\mathbf{H}$*. La suite* (u_n) *est ultimement constante.*

PREUVE. Comme une suite de Cauchy a au plus un point d'accumulation, si elle est de plus rationnelle, par le corollaire 2.72, elle prend un nombre fini de valeurs. De plus, comme chaque valeur répétée infiniment souvent est un point d'accumulation de la suite, cela lui impose d'être ultimement constante. □

2.6.3 Fermeture d'une partie positive

Comme on l'a déjà mentionné, la conjecture de type II de Rhodes fut prouvée par C.J. Ash [Ash91] en utilisant la théorie des semigroupes finis. La version topologique du résultat, proposée par J.-E. Pin [Pin88], fut prouvée par P. Ribes et P.A. Zalesskii [RZ93]. Récemment, B. Herwig et D. Lascar ont donné une nouvelle preuve [HL00] du résultat de P. Ribes et P.A. Zalesskii. Au delà de la conjecture de type II, J. Almeida et M. Delgado [AD99, AD00] ont montré de quelle façon ces trois résultats étaient liés et, dans cette étude, ils ont montré l'importance (entre autres) du problème suivant : certains sous-ensembles rationnels du groupe libre fermés pour la topologie profinie ne peuvent pas être obtenus comme la fermeture profinie d'un langage rationnel de A^*. Nous abordons ici ce problème.

Proposition 2.74 *Soit* S *un sous-ensemble rationnel du groupe libre. Il existe un langage* R *de* A^* *tel que* $\mathrm{Cl}_F(R) = S$ *si et seulement si* $\mathrm{Cl}_F(S \cap A^*) = S$*.*

PREUVE. Si $\mathrm{Cl}_F(S \cap A^*) = S$, alors $R = S \cap A^*$ est rationnel et d'après le théorème 1.66, $\mathrm{Cl}_F(R) = S$.

Réciproquement, s'il existe un langage rationnel R de A^* tel que $\mathrm{Cl}_F(R) = S$, alors $R \subseteq S \cap A^*$. De plus,

$$S = \mathrm{Cl}_F(R) \subseteq \mathrm{Cl}_F(S \cap A^*) \subseteq \mathrm{Cl}_F(S) = S$$

Par conséquent, $\mathrm{Cl}_F(S \cap A^*) = L$. □

On en déduit le résultat suivant.

Proposition 2.75 *Soit* S *une partie rationnelle du groupe libre donnée par un automate déterministe* $\mathcal{A}$ *à* n *états reconnaissant* $D(S\pi^{-1})$*. On peut tester en temps* $O(n^3)$ *si* S *est la fermeture profinie d'un langage rationnel de* A^**.*

L'algorithme suivant s'applique à un automate déterministe complet $\mathcal{A} = (Q, \tilde{A}, E, i, F)$ reconnaissant les mots réduits d'une partie rationnelle S.

Algorithme 2.76

1. Calculer l'automate $\mathcal{A}_0$ obtenu en supprimant dans $\mathcal{A}$ les transitions dont l'étiquette appartient à $\overline{A}$.
2. $\mathcal{A}_1 =$ `MotsRéduits`(`FermetureG`($\mathcal{A}_0$)).
3. Calculer l'automate $\mathcal{A}_2$ obtenu en faisant le produit de $\mathcal{A}_1$ par $(Q, \tilde{A}, E, i, Q \setminus F)$.
4. Tester si $L(\mathcal{A}_2)$ est vide.

$\diamondsuit$

PREUVE. La correction de cet algorithme découle de la proposition 2.74: l'automate $\mathcal{A}_0$ reconnaît $S \cap A^*$ et, par conséquent, l'automate $\mathcal{A}_1$ reconnaît la fermeture profinie de $S \cap A^*$. De plus, comme $D(S^c\pi^{-1}) = D(S\pi^{-1})^c$, l'automate $(Q, \tilde{A}, E, i, Q \setminus F)$ reconnaît S. Il en résulte que $L(\mathcal{A}_2)$ est vide si et seulement si $S \subseteq \mathrm{Cl}_F(S \cap A^*)$. Or, on a toujours $\mathrm{Cl}_F(S \cap A^*) \subseteq \mathrm{Cl}_F(S) = S$. Par conséquent, $L(\mathcal{A}_2)$ est vide si et seulement si $S \subseteq \mathrm{Cl}_F(S \cap A^*)$.

Analysons la complexité dans le cas le pire.

1. Cette étape s'effectue en temps $O(n)$.
2. D'après le théorème 2.56, on peut réaliser cette étape en temps $O(n^3)$. L'automate $\mathcal{A}_1$ obtenu n'est pas nécessairement déterministe mais a au plus n états.
3. Cette étape s'effectue en temps $O(n)$.
4. Cette étape se fait en temps $O(n^2)$.
5. Par le calcul d'un arbre couvrant, cette étape se fait en temps $O(n^2)$.

On obtient donc une complexité totale, dans le cas le pire, en $O(n^3)$. $\square$

Etudions maintenant le cas particulier où l'ensemble S est un sous-groupe rationnel du groupe libre. Rappelons que si H est un sous-groupe rationnel du groupe libre, on note $\mathcal{A}_H$ l'automate de sous-groupe ayant un nombre minimal d'états reconnaissant H. De plus, $\mathcal{A}_H^+$ désigne l'automate obtenu à partir de $\mathcal{A}_H$ en supprimant toutes les transitions dont l'étiquette appartient à $\overline{A}$ (voir section 1.7.2)

Proposition 2.77 *Un sous groupe rationnel H du groupe libre est la fermeture profinie d'un langage rationnel de A^* si et seulement si l'automate $\mathcal{A}_H^+$ est fortement connexe.*

Preuve. Soit H un sous-groupe rationnel du groupe libre. D'après l'algorithme 2.50 FermetureG, calculer la fermeture profinie d'une partie donnée par un automate fortement connexe consiste exactement à ajouter les transitions duales. Donc, si $\mathcal{A}_H^+$ est fortement connexe, alors $\mathrm{Cl}_F(L(\mathcal{A}_H^+)) = H$.

Réciproquement, supposons que $\mathcal{A}_H^+$ ne soit pas fortement connexe. Comme $\mathcal{A}_H^+$ a un unique état final qui est aussi l'état initial, il n'est pas émondé. D'où, en émondant $\mathcal{A}_H^+$, on obtient un automate $\mathcal{A}_0$ qui a strictement moins d'états que $\mathcal{A}_H^+$. En ajoutant à $\mathcal{A}_0$ ses transitions duales, on obtient un automate de sous-groupe $\mathcal{A}_1$ tel que

$$L(\mathcal{A}_1)\pi = \mathrm{Cl}_F(L(\mathcal{A}_0)) = \mathrm{Cl}_F(L(\mathcal{A}_H^+))$$

Comme $\mathcal{A}_1$ possède strictement moins d'états que $\mathcal{A}_H$ et comme $\mathcal{A}_H$ est l'automate de sous-groupe reconnaissant H ayant le moins d'états,

$$\mathrm{Cl}_F(L(\mathcal{A}_H^+)) \neq H$$

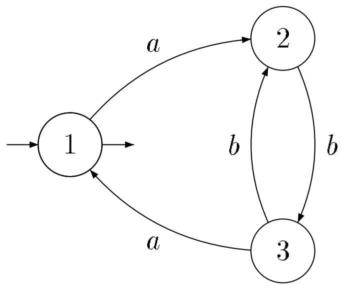
Or $L(\mathcal{A}_H^+) = H \cap A^*$. Donc, par la proposition 2.74, H n'est pas la fermeture profinie d'un langage rationnel de A^*. □

Notons que les sous-groupes H tels que $\mathcal{A}_H^+$ est fortement connexe ont d'autres propriétés intéressantes, comme cela est montré dans [MW00] et [Ste00b].

Une conséquence immédiate de ce résultat est le corollaire suivant.

Corollaire 2.78 *Soit H un sous-groupe rationnel du groupe libre donné par son automate de sous-groupe à n états. On peut tester en temps $O(n)$ si H est la fermeture profinie d'un langage de A^*.*

Exemple 2.79 Par exemple, le sous-groupe H engendré par $ab^2\overline{a}$ et aba est la fermeture profinie d'un langage rationnel de A^*. En effet, l'automate $\mathcal{A}_H^+$ est fortement connexe.



Automate $\mathcal{A}_H^+$

Le sous-groupe H est la fermeture profinie de $L(\mathcal{A}_H^+) = [ab(b^2)^*a]^*$.

▽

2.6.4 Langages non rationnels et topologie profinie

On vient de voir que les langages rationnels ont de fortes propriétés topologiques, dont l'origine se trouve dans le théorème de Hall et le théorème de Ribes et Zalesskii. Cependant, ce fort lien entre les langages rationnels et la topologie (notamment profinie) a ses limites, comme nous le verrons à la section 4.2.2 où nous donnons un exemple de langage non rationnel à la fois ouvert et fermé. Il ne faut pas s'étonner de cette situation : la définition des topologies pro-**H** utilise des arguments "non rationnels" et il est donc normal qu'il y ait une limite à la relation entre parties rationnelles et topologie pro-**H**.

Si l'on regarde les langages non rationnels les plus classiques, on peut voir qu'ils ont aussi des propriétés topologiques.

Proposition 2.80 *L'ensemble $\{a^n b^n \mid n \in \mathbb{N}\}$ est fermé pour toute topologie pro-***H** *et ne possède aucun point isolé.*

Preuve. On pose $K = \{a^n b^n \mid n \in \mathbb{N}\}$. Aucun point de K n'est isolé car la limite de $a^{n+k!}b^{n+k!}$, lorsque k tend vers $+\infty$, est $a^n b^n$. Donc tout point $a^n b^n$ de K est limite d'une suite d'éléments distincts de K.

Montrons que K est fermé. En effet, a^*b^* est un langage fermé (par application directe de l'algorithme `TesteFerméGA*`). On note par R_n le langage des mots u sur $\{a, b\}$ contenant autant de a que de b modulo n. Les langages R_n sont des langages à groupe. De plus, le langage R de A^* des mots contenant autant de a que de b est fermé : c'est l'intersection de tous les R_n, pour $n \geq 1$. Par conséquent, comme $K = R \cap a^*b^*$, K est fermé pour la topologie profinie.

□

Un autre langage non rationnel souvent étudié est le *langage de Dyck*.

langage de Dyck

Définition 2.81 *Le* langage de Dyck *associé à A est le langage D_A^* de $\tilde{A}^*$ défini par : $D_A^* = 1\pi^{-1}$.*

Proposition 2.82 *Le langage de Dyck D_A^* est fermé pour la topologie profinie. De plus, son complémentaire est dense pour la topologie profinie.*

Preuve. Il est montré dans [Pin91] que le morphisme π est continu pour la topologie profinie. Par conséquent, comme $\{1\}$ est un fermé de $F(A)$, il en résulte que D_A est fermé.

Par ailleurs, pour tout $u \in \tilde{A}^*$ et tout $n \geq |u|$, $ua^{n!} \notin D_A$. Or la suite $(ua^{n!})$ converge vers u, donc D_A^c est dense. □

2.7 Conclusion

Le calcul de la fermeture d'une partie rationnelle du groupe libre donnée par un automate fini peut donc se faire, pour peu que l'on sache calculer la fermeture d'un sous-groupe rationnel, de façon efficace. Tout l'intérêt d'avoir des algorithmes pour les automates plutôt que pour les expressions rationnelles provient du fait qu'il est facile de calculer l'intersection de deux sous-ensembles donnés par des automates, de même qu'il est facile de calculer le complémentaire d'un sous-ensemble donné par un automate déterministe, alors que cela devient difficile s'ils sont donnés par des expressions rationnelles. C'est ainsi que l'on peut facilement calculer l'ensemble dérivé d'un langage rationnel ou tester si un langage est fermé.

De plus, l'approche algébrique développée dans [PW97], ou l'approche topologique vue précédemment, ne permettent (actuellement) que de tester en temps polynomial si un langage est fermé pour la topologie profinie uniquement dans le cas où celui-ci est donné par un automate déterministe. On peut alors se demander si ce problème reste polynomial si le langage est donné par un automate non déterministe.

Chapitre 3

Automates réversibles

Sommaire

3.1	**Introduction**	**113**
3.2	**Définitions et propriétés caractéristiques**	**114**
3.2.1	Définitions et premières propriétés	114
3.2.2	Propriétés algébriques	117
3.2.3	Automates réversibles et groupe libre	118
3.2.4	Complémentaire d'un automate réversible	118
3.3	**Automates réversibles et topologie**	**121**
3.3.1	Propriétés topologiques des langages réversibles	121
3.3.2	Langages fermés et automates réversibles	125
3.4	**Algorithmes pour automates réversibles**	**128**
3.4.1	Test de réversibilité	128
3.4.2	Calcul d'un automate réversible	129
3.5	**Conclusion**	**139**

3.1 Introduction

Un automate réversible est un automate dans lequel chaque lettre induit une fonction injective de l'ensemble des états dans lui-même. Le système de transitions d'un automate réversible est donc déterministe, dans le sens où, étant donnés une lettre et un état, au plus une transition étiquetée par cette lettre part de cet état. De plus, l'injectivité impose que cette propriété est préservée si l'on inverse le sens des transitions (d'où le terme de "réversible"). Cependant, à l'instar des automates à groupe, un automate réversible n'est pas nécessairement déterministe, car il peut posséder plusieurs états initiaux.

Dans [Ang82], D. Angluin montre l'utilité des automates réversibles relativement à des problèmes d'intelligence artificielle. De même, leurs liens avec certains codes bipréfixes ont été mis en valeur par T. Hall dans [Hal84].

Par ailleurs, depuis quelques années, les informaticiens étudient un nouveau modèle de calcul : les ordinateurs quantiques. La réalisation pratique de telles machines n'est pas au point mais, d'un point de vue théorique, leurs capacités dépassent celles des machines de Turing, comme l'a montré P. Schor [Sch94] en donnant un algorithme polynomial de décomposition des entiers. Une version quantique des automates finis a alors été introduite par C. Moore et J. Crutchfield [MC00] (voir aussi [KW97]) et, peu après, A. Ambrainis et R. Freivalds ont montré des liens étroits entre ces modèles d'automates quantiques et les automates réversibles [AF98].

Signalons enfin que les automates réversibles sont utiles pour l'étude des semigroupes inversifs, comme l'ont montré par exemple P.V. Silva [Sil96] et B. Steinberg [Ste00c].

Le problème de la hauteur d'étoile d'un langage (quel est le nombre minimal d'étoiles imbriquées qu'il faut pour représenter un langage rationnel à l'aide d'une expression rationnelle?), fut longtemps ouvert. C'est K. Hashiguchi qui montra en 1978 [Has88] que l'on pouvait décider la hauteur d'étoile d'un langage. Cependant, l'algorithme proposé est très loin d'être efficace et R.S. Cohen [Coh70] avait déjà montré en 1970 que l'on pouvait décider, de façon bien plus efficace, la hauteur d'étoile d'un langage reconnaissable par un automate réversible ne possédant qu'un seul état initial et qu'un seul état final. Récemment, S. Lombardy et J. Sakarovitch [LS00] ont étendu ce résultat en montrant que la hauteur d'étoile d'un langage était "raisonnablement" décidable pour un langage reconnaissable par un automate réversible.

Une étude des propriétés caractéristiques des langages reconnaissables par des automates réversibles est détaillée dans un article de J.-E. Pin [Pin92], où l'on en trouve une description algébrique et topologique. Ces résultats sont rappelés dans la section 3.2.2.

Nous commencerons par définir les automates et langages réversibles en en donnant les propriétés principales. Ensuite, nous montrerons les propriétés topologiques des langages réversibles et nous finirons par des considérations algorithmiques.

3.2 Définitions et propriétés caractéristiques

Commençons par définir un *automate réversible.*

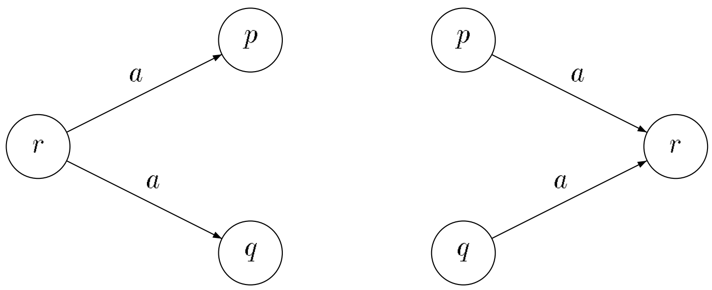
3.2.1 Définitions et premières propriétés

automate réversible

Définition 3.1 *Un automate* $\mathcal{A} = (Q, A, E, I, F)$ *est dit* réversible *si quelques soient les états* p, q, r *et la lettre* $a \in A$,

Si $(p, a, r) \in E$ *et* $(q, a, r) \in E$ *alors* $p = q$.
Si $(r, a, p) \in E$ *et* $(r, a, q) \in E$ *alors* $p = q$.

Autrement dit, un automate est réversible s'il ne possède aucune configuration du type :



avec $p \neq q$.

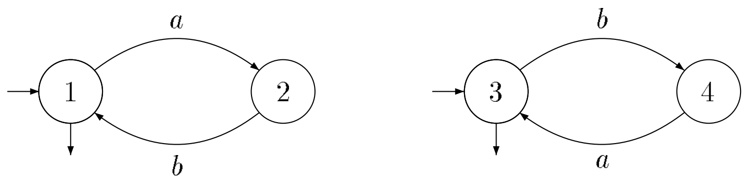
Il faut remarquer ici que la propriété de réversibilité est une propriété du graphe de l'automate et que les états initiaux et finaux n'entrent pas en jeu.

On définit alors très naturellement un *langage réversible*.

langage réversible

Définition 3.2 *Un langage de A^* est dit* réversible *s'il existe un automate réversible qui le reconnaît.*

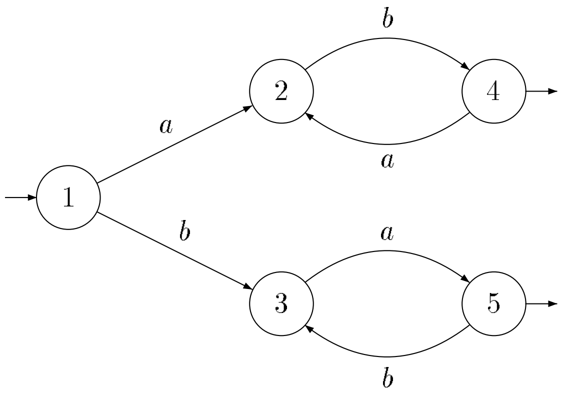
Exemple 3.3 Le langage $(ab)^* \cup (ba)^*$ est reconnu par l'automate réversible suivant :



$\triangledown$

Etant reconnaissable par un automate, un langage réversible est rationnel. Cependant, comme nous le verrons un peu plus loin, cette inclusion est stricte : certains langages rationnels ne sont pas reconnaissables par un automate réversible, comme par exemple $\{a, b\}^*a$.

L'exemple 3.3 ci-dessus nous montre de plus que l'automate minimal d'un langage réversible n'est pas nécessairement réversible. L'automate minimal de $(ab)^* \cup (ba)^*$ est en effet le suivant :



La situation est ici différente du cas des langages à groupe : leurs automates minimaux sont des automates à groupes. On touche là à un problème important. L'automate minimal est en effet une façon très pertinente et très efficace (d'un point de vue algorithmique), de décrire un langage rationnel. Cependant, comme on vient de le voir, l'automate minimal d'un langage réversible n'est pas nécessairement réversible ; on peut donc se demander comment tester si un langage (donné par son automate minimal), est réversible et, le cas échéant, comment calculer un automate réversible le reconnaissant. Ces questions seront abordées plus en détail à la section 3.4.

De plus, le fait qu'un automate réversible n'est pas nécessairement déterministe pose problème lorsque l'on est à la recherche des propriétés algorithmiques : il est difficile, par exemple, de tester l'inclusion de deux langages donnés par des automates non déterministes. D'après l'exemple 3.3, un langage peut être réversible sans que son automate minimal le soit, on peut se demander si un langage réversible peut toujours être reconnu par un automate réversible déterministe. La proposition ci-dessous nous montre que ce n'est pas le cas.

Proposition 3.4 *Le langage $(ab)^* \cup (ba)^*$ est réversible mais ne peut pas être reconnu par un automate réversible déterministe.*

Preuve. Supposons qu'il existe un automate réversible déterministe $\mathcal{A}$ reconnaissant L. Appelons i son état initial. Alors $i \cdot (ab) \neq i$ car sinon $abba \in L$. Comme l'automate a un nombre fini d'états, il existe donc deux entiers non nuls n et k tels que

$$i \cdot (ab)^n = i \cdot (ab)^{n+k}$$

Par conséquent, si on prend k minimal (non nul) vérifiant cette propriété, on peut arriver dans l'état $i \cdot (ab)^n$ en lisant b à partir des deux états distincts $i \cdot (ab)^{n-1}a$ et $i \cdot (ab)^{n+k-1}a$. Il en résulte que $\mathcal{A}$ n'est pas réversible. □

Proposition 3.5 *La classe des langages réversibles est close par image miroir, par union et intersection finie et par quotient gauche et droit.*

Preuve. Soient $\mathcal{A} = (Q, A, E, I, F)$ un automate et $E^\sim = \{(q, a, p) \mid (p, a, q) \in E\}$. Il est connu que l'automate $\mathcal{A}^\sim = (Q, A, E^\sim, F, I)$ reconnaît exactement $L(\mathcal{A})^\sim$. On vérifie facilement que si un automate $\mathcal{A}$ est réversible alors $\mathcal{A}^\sim$ l'est aussi, donc la classe des langages réversibles est close par image miroir. De même, l'union de deux automates réversibles étant aussi un automate réversible, on obtient la clôture par union. Par ailleurs, on vérifie aisément que le produit de deux automates réversibles est aussi réversible et donc que la classe des langages réversibles est close par intersection (voir section 1.6.8).

Si $\mathcal{A} = (Q, A, E, I, F)$ est un automate réversible, alors $(Q, A, E, I \cdot u, F)$ est réversible et reconnaît $u^{-1}L$. Par conséquent, la classe des langages réversibles est stable par quotient gauche. Et comme $(Lu^{-1})^\sim = (u^\sim)^{-1}L^\sim$, la classe des langages réversibles est stable par quotient droit. □

En revanche, le complémentaire d'un langage réversible n'est pas nécessairement réversible, comme nous le verrons à la proposition 3.15.

3.2.2 Propriétés algébriques

Le résultat suivant est montré dans [Pin92] et caractérise de façon algébrique les langages réversibles.

Théorème 3.6 *Un langage L est réversible si et seulement si les idempotents de son monoïde syntaxique commutent et si, de plus,*

$$\forall x, u, y \in A^*, \quad xu^+y \in L \quad \textit{implique} \quad xy \in L$$

Le sens direct de ce théorème se montre en prenant un automate réversible reconnaissant le langage : les propriétés découlent alors assez naturellement de la forte contrainte qu'impose la réversibilité. En revanche, la réciproque est plus complexe et sa preuve repose sur des résultats non triviaux de théorie des semigroupes finis. Cette propriété caractéristique permet d'établir un algorithme pour tester si un langage est réversible, comme nous le verrons à la section 3.4.

Par exemple, le langage $L = \{a, b\}^*a$ n'est pas réversible, car a^+ est inclus dans L, mais $\varepsilon \notin L$.

Notons que la classe des langages réversibles forme une variété positive de langages et que la variété de monoïdes ordonnés qui lui est associée est notée, dans la littérature, $\mathbf{E}^-_{\mathbf{com}}$.

3.2.3 Automates réversibles et groupe libre

Un automate réversible est proche d'un automate à groupe ; ce qui lui donne des propriétés vis-à-vis des groupes et notamment du groupe libre, comme nous allons le voir dans cette section.

Définition 3.7 *On note $\mathcal{C}$ la famille des parties S de $F(A)$ telles qu'il existe un automate réversible dual $\mathcal{A}$ sur $\tilde{A}$ vérifiant :*

$$S = L(\mathcal{A})\pi$$

En particulier, les sous-groupes rationnels du groupe libre sont dans $\mathcal{C}$, car un automate de sous-groupe est dual et réversible (il est même déterministe ; voir définition 1.74).

Le résultat suivant est prouvé dans [Pin92].

Théorème 3.8 *Un sous ensemble S du groupe libre $F(A)$ appartient à $\mathcal{C}$ si et seulement s'il est une union finie de classes à gauche de sous-groupes rationnels de $F(A)$.*

Une conséquence directe en est le théorème [Pin92] suivant, que l'on ne manquera pas de mettre en correspondance avec la proposition 2.35 et qui annonce déjà de forts liens avec la topologie profinie.

Théorème 3.9 *La classe $\mathcal{C}$ est la plus petite classe $\mathcal{S}$ de sous-ensembles de $F(A)$ telle que :*
- *$\emptyset \in \mathcal{S}$, pour tout $g \in F(A)$, $\{g\} \in \mathcal{S}$.*
- *Si $S_1, S_2 \in \mathcal{S}$, alors $S_1 \cup S_2 \in \mathcal{S}$.*
- *Si $S \in \mathcal{S}$ et $g \in F(A)$, alors $gS \in \mathcal{S}$.*
- *Si $S \in \mathcal{S}$, alors $\langle S \rangle \in \mathcal{S}$.*

On obtient ainsi une nouvelle caractérisation des langages réversibles :

Corollaire 3.10 *Un langage L sur l'alphabet A est réversible si et seulement s'il existe $K \in \mathcal{C}$ tel que*

$$L = K \cap A^*$$

3.2.4 Complémentaire d'un automate réversible

Dans [MP87], S.W. Margolis et J.-E. Pin donnent une caractérisation des langages dont les idempotents du semigroupe syntaxique commutent.

Proposition 3.11 *Soit L un langage rationnel de A^*. Les assertions suivantes sont équivalents :*

(i) Les idempotents du monoïde syntaxique de L commutent.

(ii) L est une combinaison booléenne de langages à groupe ou du type KaA^*, où K est un langage à groupe.

(iii) L est une combinaison booléenne de langages à groupe ou du type A^*aK, où K est un langage à groupe.

(iv) L est une combinaison booléenne de langages à groupe ou du type KaK', où K et K' sont des langages à groupe.

Une version plus précise de ce résultat, s'appliquant aux langages réversibles, peut être obtenue en combinant des résultats de [PW01a] et [PW01b].

Proposition 3.12 *Le complémentaire d'un langage réversible peut s'exprimer comme une combinaison finie d'unions et d'intersections de langages à groupe ou du type KaA^*, où K est un langage à groupe.*

Nous donnons ici une preuve directe de ce résultat qui n'utilise que des propriétés des automates réversibles.

Preuve. Soit $\mathcal{A}$ un automate réversible. En remarquant que $L(\mathcal{A})$ est l'union (finie) des $L(\mathcal{A}^{i,f})$, où i et f décrivent respectivement l'ensemble des états initiaux et des états finaux de $\mathcal{A}$, il suffit de prouver le résultat pour un automate réversible déterministe ne possédant qu'un seul état final.

Soit $\mathcal{A} = (Q, A, E, i, f)$ un tel automate. Si $\mathcal{A}$ est complet, alors c'est un automate à groupe et donc $L(\mathcal{A})^c$ est un langage à groupe. Supposons donc que $\mathcal{A}$ ne soit pas complet. Soit $\mathcal{A}_0 = (Q, A, E_0, i, f)$ un automate à groupe (déterministe) obtenu à partir de $\mathcal{A}$ en ajoutant des transitions à $\mathcal{A}$ (ce qui est toujours possible, mais pas nécessairement de façon unique). Comme on a ajouté des transitions, pour tout état $p \in Q$,

$$L(\mathcal{A}^{i,p}) \subseteq L(\mathcal{A}_0^{i,p})$$

Montrons que $L(\mathcal{A}_0^{i,p}) \subseteq L(\mathcal{A})^c$ dès que $p \neq f$. En effet, si $u \in L(\mathcal{A}_0^{i,p})$ avec $p \neq f$, alors soit $u \in L(\mathcal{A}^{i,p})$, auquel cas $u \notin L(\mathcal{A})$ (car $\mathcal{A}$ est déterministe), soit, pour lire u dans $\mathcal{A}_0^{i,p}$, on emprunte au moins une transition de $\mathcal{A}_0$ qui n'est pas une transition de $\mathcal{A}$, auquel cas $u \notin L(\mathcal{A})$ (car $\mathcal{A}$ est déterministe).

Considérons maintenant l'automate $\mathcal{A}_1$ obtenu à partir de $\mathcal{A}$ en le complétant. On appelle p_1 l'état puits ajouté et E_1 les transitions ajoutées. Rappelons que toutes les transitions de E_1 ont pour extrémité terminale p_1. Posons enfin $\mathcal{A}_2 = (Q \cup \{p_1\}, A, E_0 \cup E_1, i, p_1)$. Par construction, $L(\mathcal{A}_1^{i,p_1}) \subseteq L(\mathcal{A}_2)$. Le même raisonnement que ci-dessus permet de montrer que $L(\mathcal{A}_2) \subseteq L(\mathcal{A})^c$.

On en déduit

$$L(\mathcal{A}_1^{i,p_1}) \cup \bigcup_{p \in Q \setminus \{f\}} L(\mathcal{A}^{i,p}) \subseteq L(\mathcal{A}_2) \cup \bigcup_{p \in Q \setminus \{f\}} L(\mathcal{A}_0^{i,p}) \subseteq L(\mathcal{A})^c$$

Or,

$$L(\mathcal{A}_1^{i,p_1}) \cup \bigcup_{p \in Q \setminus \{f\}} L(\mathcal{A}^{i,p}) = L(\mathcal{A})^c$$

Par conséquent,

$$L(\mathcal{A}_2) \cup \bigcup_{p \in Q \setminus \{f\}} L(\mathcal{A}_0^{i,p}) = L(\mathcal{A})^c$$

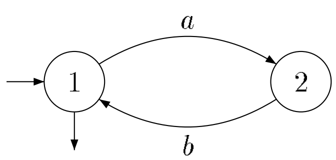
Comme les $L(\mathcal{A}_0^{i,p})$ sont des langages à groupe et comme

$$L(\mathcal{A}_2) = \bigcup_{(p,a,p_1) \in E_1} L(\mathcal{A}_0^{i,p}) a A^*$$

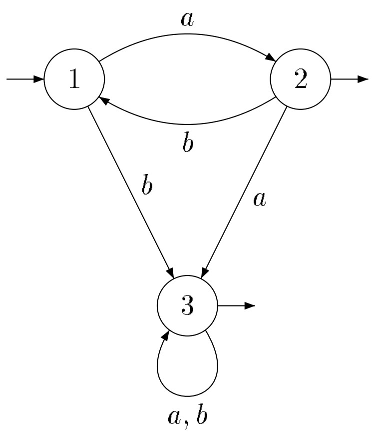
le complémentaire de $L(\mathcal{A})$ est bien une union/intersection de langages à groupe ou de langages du type KaA^* où K est un langage à groupe (les formules ci-dessus ne font apparaître que des unions : les intersections interviennent lorsque l'on revient au cas général, c'est-à-dire d'un automate réversible possédant plusieurs états initiaux et finaux). □

On peut remarquer que la preuve ci-dessus est constructive et que, de plus, si l'on se donne un automate réversible $\mathcal{A}$ à n états, on peut calculer une décomposition de $L(\mathcal{A})^c$ en une union/intersection de langages à groupe ou de langages du type KaA^* en temps polynomial en n.

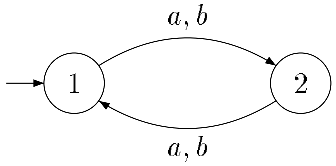
Exemple 3.13 Considérons par exemple le langage réversible $(ab)^*$ reconnaissable par l'automate réversible $\mathcal{A}$ suivant :



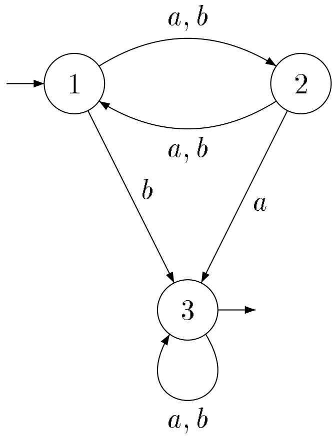
Son complémentaire est reconnaissable par l'automate $\mathcal{A}_1$ suivant :



Par ailleurs, on peut compléter $\mathcal{A}$ afin d'obtenir un automate à groupe $\mathcal{A}_0$ de la façon suivante :



On construit ainsi l'automate $\mathcal{A}_2$ suivant :



On obtient donc que $[(ab)^*]^c = (A^2)^*A \cup (A^2)^*AaA^* \cup (A^2)^*bA^*$.

$\triangledown$

3.3 Automates réversibles et topologie

3.3.1 Propriétés topologiques des langages réversibles

Si l'on compare le théorème 3.9 avec la proposition 2.35, on remarque qu'il y a un lien entre la classe des parties rationnelles fermées du groupe libre et la classe des parties reconnaissables par un automate réversible dual. Ce lien se préserve en passant au monoïde libre : les langages réversibles possèdent de remarquables propriétés topologiques.

Proposition 3.14 *Tout langage réversible est fermé pour la topologie profinie.*

Cette proposition est montrée dans [Pin92] en utilisant la caractérisation syntaxique des langages fermés (voir la proposition 2.20 et le théorème 3.6). Nous en donnons ici une preuve basée sur l'algorithme 2.57 FermetureGA* et donc plus topologique.

Preuve. Soit L un langage réversible. Il existe un automate réversible $\mathcal{A}$ tel que $L(\mathcal{A}) = L$. Montrons que $\mathcal{A}$ est invariant par l'algorithme FermetureGA*.

La première étape de l'algorithme consiste à ajouter des transitions $(p, \overline{a}, q)$ dans $\mathcal{A}$, chaque fois que (q, a, p) est une transition de $\mathcal{A}$ et que p et q sont dans la même composante fortement connexe. On obtient ainsi un nouvel automate $\mathcal{A}_1$. Pour la fin de l'algorithme il faut maintenant, chaque fois que l'on a un chemin dans $\mathcal{A}_1$ étiqueté par un mot de $\tilde{A}^*$ du type $u\overline{u}$ et allant d'un état p à un état q, ajouter une ε-transition de p à q, puis calculer la clôture transitive de ses ε-transitions. Il faut ensuite éliminer ces ε-transitions et, enfin, enlever toutes les transitions étiquetées par une lettre de $\overline{A}$.

Prouvons que si m est un chemin dans $\mathcal{A}_1$ tel que $[m] = u\overline{u}$ et allant de p à q, alors $p = q$. Faisons une démonstration par récurrence sur la longueur de u.

- Si $u \in A$, alors $p \cdot u = q \cdot u$. Comme $\mathcal{A}$ est réversible, cela impose que $p = q$. Il en va de même si $u \in \overline{A}$, en utilisant le déterminisme des transitions.
- Supposons que si $p \cdot u\overline{u} = q$ avec $|u| \leq n$, alors $p = q$. Considérons alors p et q tels que $p \cdot u\overline{u} = q$ et $|u| = n + 1$. On pose $u = va$ avec $a \in \tilde{A}$, $p' = p \cdot v$ et $q' = p \cdot u\overline{a}$. En appliquant l'hypothèse de récurrence à p' et q', on obtient que $p' = q'$. Par conséquent, $p \cdot v\overline{v} = q$ et donc, d'après l'hypothèse de récurrence, comme $|v| = n$, $p = q$.

Par conséquent, les ε-transitions ajoutées au début de l'algorithme sont des boucles de longueur 1. Elles forment donc un graphe transitif et la dernière étape consistant à les éliminer ne change en rien l'automate. Il en résulte que l'automate obtenu après application de l'algorithme est exactement l'automate de départ.

L'automate $\mathcal{A}$ est invariant par un algorithme calculant la fermeture profinie d'un langage donc $L(\mathcal{A})$ est fermé. □

On obtient maintenant, grâce à la proposition 3.14, une caractérisation des langages réversibles dont le complémentaire est réversible.

Proposition 3.15 *Un langage rationnel et son complémentaire sont réversibles si et seulement s'ils sont des langages à groupe.*

Preuve. Un langage réversible est fermé pour la topologie profinie. Par conséquent un langage réversible dont le complémentaire est aussi réversible est à la fois ouvert et fermé. Comme seuls les langages à groupe sont à la fois rationnels, ouverts et fermés, le langage est un langage à groupe. □

D'un point de vue général, en topologie, une partie est toujours l'union dijointe de son intérieur et de sa frontière. Dans le cas de la topologie profinie, on montre que l'intérieur d'un langage réversible est un langage à groupe et que sa frontière est un langage réversible.

Proposition 3.16 *Tout langage réversible L peut se décomposer sous la forme d'une union disjointe*

$$L = L_0 \cup L_1$$

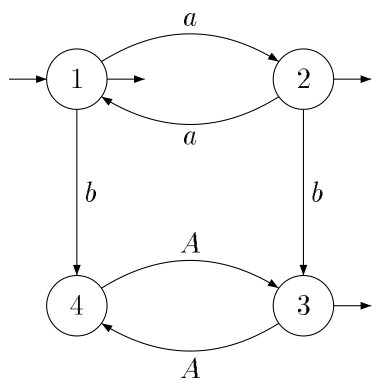
où L_0 est un langage à groupe et L_1 un langage réversible d'intérieur vide pour la topologie profinie.

Preuve. Soit L un langage réversible et L_0 son intérieur. Le complémentaire de L_0 est donc la fermeture du complémentaire de L. Comme L est fermé, L_0^c est donc la fermeture d'un langage rationnel ouvert et il est donc un langage à groupe (d'après le théorème 2.17). Par conséquent L_0 est aussi un langage à groupe.

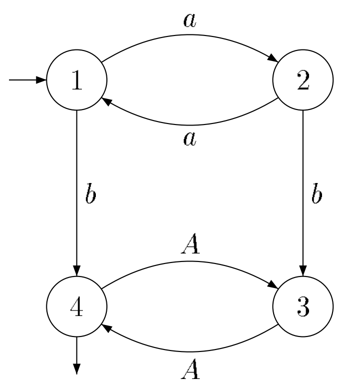
On pose alors $L_1 = L \setminus L_0 = L \cap L_0^c$. Or, comme L_0 est à groupe, son complémentaire l'est aussi et, comme l'intersection de deux langages réversibles est aussi réversible (proposition 3.5), L_1 est réversible. De plus, comme L_0 est l'intérieur de L, le langage L_1 est d'intérieur vide. □

Il est à noter que grâce aux algorithmes du chapitre 2 la décomposition peut se calculer explicitement. De plus, les langages à groupe forment une base d'ouverts de la topologie profinie donc, dans la décomposition précédente, L_1 ne contient aucun langage à groupe non vide.

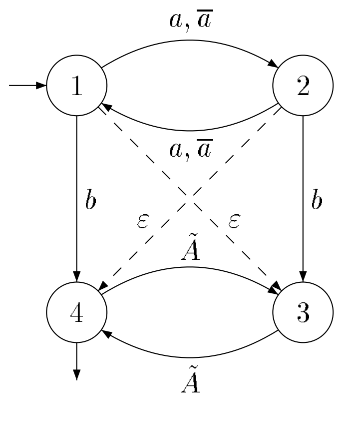
Exemple 3.17 Considérons le langage L donné par l'automate (minimal) suivant :



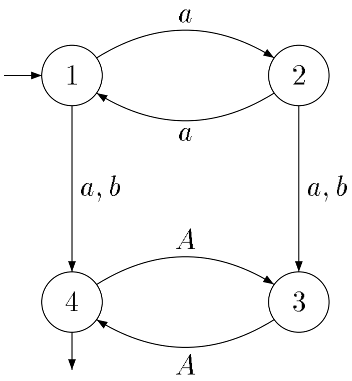
Le complémentaire de L est donc donné par l'automate suivant :



On applique l'algorithme FermetureGA* à l'automate ci-dessus : on rend les composantes fortement connexes duales et on ajoute une ε-transition entre p et q si l'on peut aller de p à q en lisant un mot de $1\pi^{-1}$. Ici, on ne rajoute ainsi que deux ε-transitions : $(1, \varepsilon, 3)$ et $(2, \varepsilon, 4)$. On obtient donc l'automate ci-dessous.



Après élimination des ε-transition, on a l'automate suivant reconnaissant le complémentaire de l'intérieur de L :



Après déterminisation et minimisation, on trouve que l'intérieur de L est le langage $L_0 = (A^2)^*$ constitué des mots de longueur paire.

Pour trouver L_1, il reste à calculer $L \cap L_0^c$, ce qui se fait par un produit d'automates. Ainsi on trouverait que $L_1 = a^*$.

On obtient donc que $L = (A^2)^* \cup a^*$.

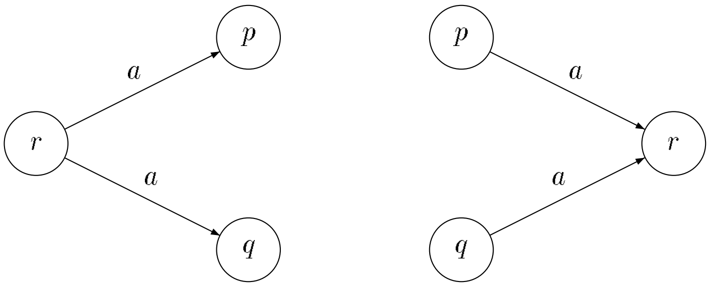
$\triangledown$

Dans l'exemple ci-dessus, la décomposition permet de calculer un automate réversible représentant le langage : les automates minimaux de $(A^2)^*$ et de a^* sont réversibles et donc leur union reconnaît L. Ça n'est pas toujours le cas. Par exemple, si on applique cette méthode au langage réversible $(A^3)^* \cup (ab)^* \cup (ba)^*$, on obtient $L_0 = (A^3)^*$ et $L_1 = (ab)^* \cup (ba)^*$ dont l'automate minimal n'est pas réversible.

3.3.2 Langages fermés et automates réversibles

Réciproquement, les automates réversibles sont omniprésents lorsque l'on étudie les propriétés topologiques des langages rationnels.

Définition 3.18 *Un automate $\mathcal{A}$ est dit* localement réversible *si ses composantes fortement connexes sont réversibles, c'est à dire que que l'on ne peut pas avoir la configuration suivante dans $\mathcal{A}$:* localement réversible



avec p, q et r dans la même composante fortement connexe.

Le théorème ci-dessous établit un lien fort entre la notion de réversibilité et les topologies pro-$\mathbf{H}$ qui sera exploité au chapitre 4.

Théorème 3.19 *Soit $\mathbf{H}$ une variété de groupes finis. L'automate minimal d'un langage fermé ou ouvert pour la topologie pro-$\mathbf{H}$ est localement réversible.*

Preuve. Soit $\mathbf{H}$ une variété de groupes finis. Comme la topologie pro-$\mathbf{H}$ est plus grossière que la topologie profinie, tout langage qui est fermé ou ouvert pour la topologie pro-$\mathbf{H}$ l'est aussi pour la topologie profinie. Par conséquent, il suffit de prouver le résultat pour un langage qui est fermé ou ouvert pour la topologie profinie.

Montrons d'abord le résultat pour un langage fermé L dont l'automate minimal est $\mathcal{A} = (Q, A, E, i, F)$. Raisonnons par l'absurde et supposons que l'on ait $p \cdot a = q \cdot a = r$, avec p, q et r dans la même composante fortement connexe de $\mathcal{A}$. On va montrer que pour tout $u \in A^*$ tel que $p \cdot u \in F$, on a $q \cdot u \in F$, ce qui montrera par symétrieque $p \sim_L q$, et donc que $p = q$ (voir section 1.6.7).

Soit donc $u \in A^*$ tel que $p \cdot u \in F$ et $w \in A^*$ tel que $i \cdot w = q$. Si on applique l'algorithme 2.57 `FermetureGA*` à $\mathcal{A}$, on voit que l'on construit une ε-transition de q à p (et aussi une de p à q). Il en résulte que wu appartient à la fermeture profinie de L. Comme L est fermé, cela implique que $wu \in L$ et donc que $i \cdot wu = q \cdot u \in F$.

Considérons maintenant un langage ouvert L. L'automate minimal $\mathcal{A}$ du complémentaire de L est donc localement réversible. Or, l'automate minimal de L est exactement l'automate que l'on obtient en complétant $\mathcal{A}$ et en changeant la finalité des états. Comme compléter un automate ajoute au plus une composante fortement connexe (qui n'a qu'un seul état et qui est donc réversible), l'automate minimal de L est aussi localement réversible. □

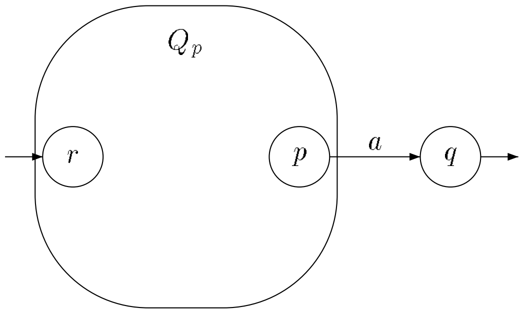
Proposition 3.20 *Tout langage reconnaissable par un automate localement réversible est une union finie de produits de langages réversibles.*

Preuve. Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate localement réversible.

Pour toute transition externe $t = (p, a, q)$ de $\mathcal{A}$ (voir définition 1.60). On note Q_p l'ensemble des états qui sont dans la même composante fortement connexe que p et E_p l'ensemble des transitions internes à Q_p. On considère ensuite les automates $\mathcal{A}_{t,r}$, où r est un état de Q_p, suivants :

$$\mathcal{A}_{t,r} = (Q_p \cup \{q\}, A, E_p \cup \{t\}, r, q)$$

Ce qui peut s'illustrer, pour $\mathcal{A}_{(p,a,q),r}$, par la figure ci-dessous.



On considère également tous les automates

$$\mathcal{A}_q(p) = (Q_q, A, E_q, p, F_q)$$

avec $p \in Q_q$, qui admettent pour graphe une des composantes fortement connexes de $\mathcal{A}$ et dont les états finaux sont les états finaux de $\mathcal{A}$ inclus dans cette composante.

Tout langage K du type

$$L(\mathcal{A}_{t_0,r_0}) \cdots L(\mathcal{A}_{t_n,r_n}) L(\mathcal{A}_q(p)) \tag{3.1}$$

où $r_0 \in I$ et pour tout $1 \leq i \leq n-1$, r_i est l'état d'arrivée de t_{i-1} et où l'état d'arrivée de t_n est p est un sous-langage de $L(\mathcal{A})$. En effet, si l'on considère l'ensemble Q' qui est l'union de tous les états des automates apparaissant dans (3.1) et, de même, E' qui est l'union de toutes les transitions des automates apparaissant dans (3.1), alors K est reconnaissable par l'automate (Q', A, E', r_0, F_q) qui peut se déduire de $\mathcal{A}$ en supprimant des transitions et des états.

De même, $L(\mathcal{A}_q(p)) \subseteq L(\mathcal{A})$, si $p \in I$.

Par ailleurs, si l'on considère un chemin réussi dans $\mathcal{A}$ et la suite des composantes fortement connexes qu'il traverse, on voit que son étiquette appartient à un des $L(\mathcal{A}_i)$, avec $i \in I$ si on reste dans la composante fortement connexe de i en parcourant le chemin, et que son étiquette appartient à un des langages définis dans (3.1) sinon.

Il en résulte que $L(\mathcal{A})$ est exactement l'union des $L(\mathcal{A}_i)$, avec $i \in I$ et des langages définis dans (3.1). Or, cette union est finie et comme les automates $\mathcal{A}_k$ et $\mathcal{A}_{t,r}$ sont réversibles, on obtient le résultat souhaité. □

On en déduit le corollaire suivant, qui peut être mis en corrélation avec le théorème 4.6.

Corollaire 3.21 *Soit* **H** *une variété de groupes finis. Tout langage rationnel qui est ouvert ou fermé pour la topologie pro-***H** *peut s'exprimer comme une union finie de produits de langages réversibles.*

On peut remarquer que la preuve de la proposition 3.20 est constructive et que l'on peut calculer de façon effective une telle décomposition. Par cette méthode, le nombre de produits peut être exponentiel par rapport au nombre d'états de l'automate minimal du langage (car il y en a un par chemin dans le graphe des composantes fortement connexes) ; cependant, dans chaque produit, le nombre de facteurs est, lui, borné par le nombre de composantes fortement connexes de l'automate minimal.

Par une preuve très similaire à celle du corollaire 3.21, on peut montrer le résultat suivant, à rapprocher aussi du théorème 4.6.

Corollaire 3.22 *Tout langage rationnel ouvert ou fermé est une union finie de langages du type* $L_0a_1L_1\ldots a_kL_k$ *où les* L_i *sont des langages réversibles et les* a_i *des lettres.*

La réciproque de ce résultat sera étudiée au chapitre 4.

3.4 Algorithmes pour automates réversibles

Cette section est consacrée à l'algorithmique des langages réversibles : on étudie tout d'abord comment tester si un langage donné par un automate déterministe est réversible, puis on s'intéresse alors à la construction effective d'un tel automate.

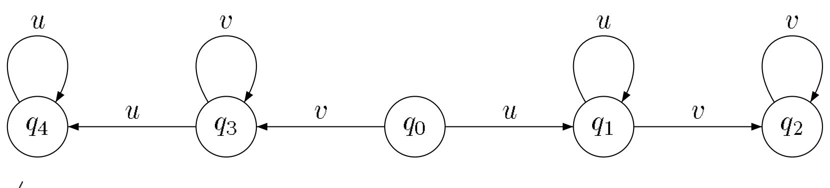
3.4.1 Test de réversibilité

Afin de tester si un langage est réversible un algorithme a été mis au point par J.-E. Pin dans [Pin92]. La méthode utilisée s'appuie sur la proposition 3.6 et consiste à tester successivement si le langage est fermé, puis si les idempotents de son semigroupe syntaxique commutent. Dans son article, J.-E. Pin propose de tester la fermeture en utilisant la caractérisation algébrique des langages fermés, ce qui est plus coûteux que l'algorithme 2.60 `TesteFerméGA*`.

Cependant l'étape lourde du test de réversibilité, même si elle est polynomiale, est la seconde, c'est-à-dire le test de la commutation des idempotents. Il en résulte que si, d'un point de vue théorique, notre algorithme (pour tester la fermeture) améliore le temps de calcul, d'un point de vue pratique cela ne change pas grand chose.

La proposition suivante est montré dans [Pin92].

Proposition 3.23 *Soit L un langage rationnel et $\mathcal{A}$ le complété de son automate minimal. Les idempotents du monoïde syntaxique de L commutent si et seulement si $\mathcal{A}$ ne possède aucune configuration du type :*



avec $q_2 \neq q_4$.

Pour vérifier une telle propriété on calcule l'automate $\mathcal{A}_1$ qui est le produit de quatre copies de $\mathcal{A}$. On calcule ensuite la clôture transitive du graphe non étiqueté sous-jacent à $\mathcal{A}_1$ et on obtient ainsi un graphe (V, E) dont les sommets sont des quadruplets d'états de $\mathcal{A}$. On peut alors montrer qu'un automate $\mathcal{A}$ ne possède pas de configuration ci-dessus si et seulement s'il n'y a pas dans (V, E) de paire d'arcs du type

$$\{((q_0, q_1, q_3, q_4), (q_1, q_1, q_4, q_4)), ((q_0, q_1, q_2, q_3), (q_3, q_2, q_2, q_3))\}$$

avec $q_2 \neq q_4$. On est donc amené à calculer la clôture transitive d'un graphe à n^4 sommets, ce qui demande donc un temps, dans le cas le pire, supérieur à n^8.

3.4.2 Calcul d'un automate réversible

En s'appuyant sur des résultats complexes de théorie des semigroupes, il est montré dans [Pin92] que si l'automate minimal d'un langage réversible L possède n états, alors L est reconnu par une union finie d'automates réversibles possédant chacun au plus $N = n^n$ états. Or, asymptotiquement, le nombre d'automates réversibles ayant moins de k états est exponentiel en k (pour une étude sur le dénombrement des automates réversibles, voir [Nic00]). Il en résulte que l'algorithme proposé demande de faire un nombre doublement exponentiel de tests (qui ne sont pas tous faisables en temps polynomial). L'algorithme consistant à énumérer tous les automates réversibles possédant moins de n^n états et à tester, à chaque fois, s'ils reconnaissent un sous-langage de L, est donc théoriquement acceptable, mais est en pratique inexploitable à moins peut-être d'améliorer la borne N.

Nous allons ici nous intéresser au problème opposé, celui de trouver une borne inférieure à un algorithme calculant un automate réversible à partir d'un automate minimal. Nous allons pour cela utiliser une méthode classique qui consiste à trouver une borne en espace, car la complexité en temps est toujours supérieure à la complexité en espace. Plus précisément, nous allons montrer qu'il existe une constante $\rho > 1$ et une suite de langages réversibles K_n possédant les propriétés suivantes.

(i) L'automate minimal de K_n possède $O(n)$ états.

(ii) Tout automate réversible reconnaissant K_n possède au moins $\Omega(\rho^n)$ états.

Rappelons que si L désigne un langage rationnel, alors $m(L)$ est le nombre d'états de son automate minimal. De façon similaire, si L est un langage réversible, on note $c(L)$ le nombre minimal d'états d'un automate réversible reconnaissant L. On pose alors

$$R_{\text{rev}}(n) = \max_{L \text{ réversible}} \{c(L) \mid m(L) \leq n\}$$

Cela signifie qu'un algorithme calculant, à partir de l'automate minimal à n états d'un langage réversible un automate réversible reconnaissant ce langage, utilisera dans le pire des cas au moins un espace $R(n)$.

Théorème 3.24 *Si* $|A| \geq 2$*, alors* $R_{\text{rev}}(n) = \Omega(\rho^n)$*, avec* $\rho = \left(\frac{3\sqrt{2}}{4}\right)^{\frac{1}{6}} \simeq$ 1.001

Pour la preuve, considérons le langage $L = \{aa, ab, ba\}$ sur l'alphabet $A = \{a, b\}$. Avant de prouver le théorème, nous avons besoin de quelques résultats techniques.

Lemme 3.25 *Soit* $\mathcal{A} = (Q, A, E, i, F)$ *un automate réversible émondé reconnaissant un sous-ensemble non vide de* $L = \{aa, ab, ba\}$*. Pour tout état* $s \in Q$*, soit* L_s *l'ensemble des mots que l'on peut lire de* i *à* s*. Alors on est dans l'un des cas suivants :*

(a) $|F| = 3$ *et* $|L_s| = 1$ *pour tout état* $s \in F$ *;*

(b) $|F| = 2$*,* $F = \{s_1, s_2\}$*,* $|L_{s_1}| = 2$ *et* $|L_{s_2}| = 1$ *;*

(c) $|F| = 2$*,* $F = \{s_1, s_2\}$*,* $|L_{s_1}| = 1$ *et* $|L_{s_2}| = 1$ *;*

(d) $|F| = 1$*,* $F = \{s\}$ *et* $|L_s| = 2$ *;*

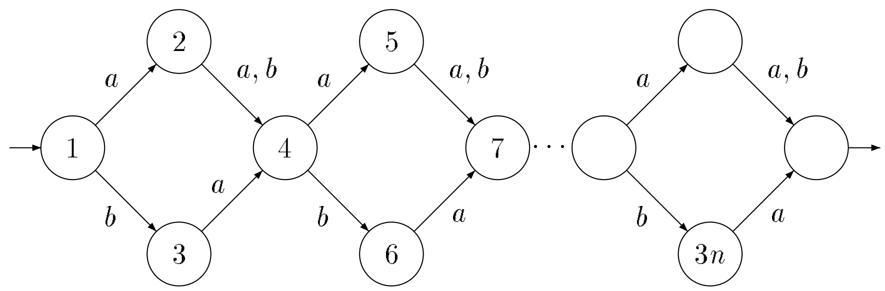
(e) $|F| = 1$*,* $F = \{s\}$ *et* $|L_s| = 1$*.*

Preuve. Si $s \neq t$, alors $L_s \cap L_t = \emptyset$ (en utilisant le déterminisme). Pour prouver le lemme il suffit donc de montrer que si F est un singleton, alors $L(\mathcal{A}) \neq L$.

Supposons que $F = \{s\}$ soit un singleton. On doit vérifier que $|L_s|$ est différent de 3 et donc que $\mathcal{A}$ n'accepte pas tous les mots de L. S'il n'y a pas de transition étiquetée par a sortant de i, alors $L_s = L(\mathcal{A}) = \{ba\}$ et on est dans le cas *(e)*. S'il n'y a pas de transition étiquetée par b sortant de i, alors $L_s \subseteq \{ab, aa\}$ et on est soit dans le cas *(d)*, soit dans le cas *(e)*. Considérons maintenant le cas où l'on a à la fois une transition étiquetée par a et une transition étiquetée par b sortant de i. Si $i \cdot a = i \cdot b$, alors il n'y a pas de transition étiquetée par b partant de $i \cdot a$ vers un état final car $bb \notin L$: on est alors dans le cas *(d)*. Finalement, supposons que $i \cdot a \neq i \cdot b$. Comme $\mathcal{A}$ est émondé, il y a une transition étiquetée par a allant de $i \cdot b$ à un état final s. Mais $\mathcal{A}$ est réversible, donc il n'y a pas de transition étiquetée par a allant de $i \cdot a$ à s et, par conséquent, le mot aa n'est pas accepté. En particulier on a $L_s = \{ab, ba\}$ et l'on est dans le cas *(d)*. □

Pour tout entier n, L^n est un langage fini et donc réversible. Nous allons prouver que $c(L^{2n}) \geq \rho^n$, alors que $m(L^{2n}) = O(n)$.

Observons en premier lieu que pour tout entier n, $|L^n| = 3^n$, que L^n est égal à $(L^n)^\sim$ et que tout mot de L^n a pour longueur $2n$. De plus, il est facile de vérifier que l'automate minimal de L^n est le suivant :



Par conséquent, l'automate minimal de L^n possède $m(L^n) = 3n + 1$ états. De plus, si $\mathcal{A}$ est un automate émondé qui reconnaît un sous-langage de L^n, alors pour toute paire d'états p et q de $\mathcal{A}$, tous les mots de $L_{p,q}$ ont la même longueur.

Soit maintenant $\mathcal{A} = (Q, A, E, i, f)$ un automate réversible possédant un unique état initial et un unique état final et acceptant un sous-langage de L^{2n}. Posons alors $M(\mathcal{A}) = i \cdot A^{2n}$, l'ensemble des états de $\mathcal{A}$ que l'on peut atteindre à partir de i en lisant un mot de longueur $2n$ (la moitié de la longueur des mots de L^{2n}). On considère enfin, pour tout état q, l'ensemble R_q des mots qui sont étiquette d'un chemin de q à f.

Proposition 3.26 *Soit $\mathcal{A} = (Q, A, E, i, f)$ un automate réversible déterministe ne possédant qu'un seul état final. Avec les notations ci-dessus, on a la relation*

$$|L(\mathcal{A})| \leq \frac{1}{2} \sum_{q \in M(\mathcal{A})} \left(|L_q|^2 + |R_q^{\sim}|^2\right) \tag{3.2}$$

Preuve. Par définition, on a :

$$L(\mathcal{A}) = \bigcup_{q \in M(\mathcal{A})} L_q R_q$$

Or $\mathcal{A}$ est déterministe, donc, si $p \neq q$, alors $L_p \cap L_q = \emptyset$. Par conséquent, si $p, q \in M(\mathcal{A})$, alors $L_p R_p \cap L_q R_q = \emptyset$. En effet, le préfixe de longueur $2n$ d'un mot de $L_p R_p \cap L_q R_q$ doit être dans $L_p \cap L_q$. Par conséquent,

$$|L(\mathcal{A})| = \sum_{q \in M(\mathcal{A})} |L_q||R_q|$$

Comme R_q est fini, on a $|R_q| = |R_q^{\sim}|$. Mais pour toute paire de réels (x, y), on a $xy \leq \frac{1}{2}(x^2 + y^2)$. On en déduit (3.2). □

Pour tout état q de Q, on note n_q le cardinal de L_q.

Proposition 3.27 *Soit $\mathcal{A} = (Q, A, E, i, F)$ un automate réversible ne possédant qu'un seul état initial et reconnaissant un sous-langage de L^n. On a l'égalité suivante :*

$$\sum_{q \in F} n_q^2 \leq 8^n \tag{3.3}$$

Afin de prouver la proposition 3.27, on procède par induction sur n. On a tout d'abord besoin de quelques lemmes techniques. Pour la preuve, on peut supposer, sans perte de généralité, que $\mathcal{A}$ est émondé.

Lemme 3.28 *Si $n = 1$, la proposition 3.27 est vraie.*

Preuve. En appliquant le lemme 3.25 à $\mathcal{A}$ on obtient

Dans le cas *(a)*, $\sum_{q \in F} n_q^2 = |L_{i,s_1}|^2 + |L_{i,s_2}|^2 + |L_{i,s_3}|^2 = 3$.
Dans le cas *(b)*, $\sum_{q \in F} n_q^2 = |L_{i,s_1}|^2 + |L_{i,s_2}|^2 = 4 + 1 = 5$.
Dans le cas *(c)*, $\sum_{q \in F} n_q^2 = |L_{i,s_1}|^2 + |L_{i,s_2}|^2 = 1 + 1 = 2$.
Dans le cas *(d)*, $\sum_{q \in F} n_q^2 = |L_{i,s_1}|^2 = 4$.
Dans le cas *(e)*, $\sum_{q \in F} n_q^2 = |L_{i,s_1}|^2 = 1$.

ce qui prouve le résultat. □

Soit S l'ensemble des états de $\mathcal{A}$ qui peuvent être atteints en lisant un mot de longueur $2n-2$ (chaque mot de L^n a pour longueur $2n$).

Lemme 3.29 *Soit $\mathcal{A} = (Q, A, E, i, F)$ un automate réversible ne possédant qu'un seul état initial et reconnaissant un sous-langage de L^n. Pour tout état q de F, on a*

$$n_q = \sum_{s \in S} n_s |L_{s,q}| \tag{3.4}$$

Preuve. Comme tout mot de $L(\mathcal{A})$ a pour longueur $2n$, on a

$$L_q = \bigcup_{s \in S} L_s L_{s,q} \tag{3.5}$$

Or $\mathcal{A}$ est déterministe donc, comme dans la preuve de la proposition 3.26, l'union dans (3.5) est disjointe. Comme pour tout état p de Q, $n_p = |L_p|$, on peut maintenant déduire la formule (3.4) de la formule (3.5). □

Montrons maintenant le lemme suivant.

Lemme 3.30 *Soit $\mathcal{A} = (Q, A, E, i, F)$ un automate réversible ne possédant qu'un seul état initial et reconnaissant un sous-langage de L^n. Pour tout $s \in S$, on a*

$$\sum_{q \in F} \sum_{t \in S} |L_{s,q}||L_{t,q}| \leq 8 \tag{3.6}$$

Preuve. Soit $s \in S$ et soit F_s l'ensemble des états de F que l'on peut atteindre à partir de s. Si $q \in F$, on note I_q l'ensemble des états S à partir desquels q peut être atteint. La somme, dans la formule (3.6), est en fait égale à

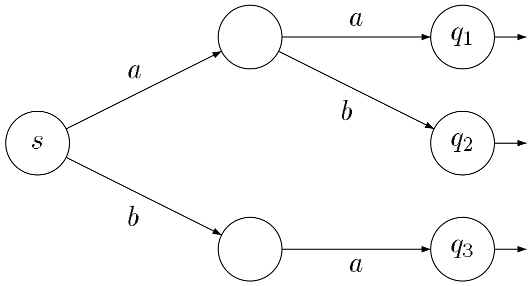
$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}|$$

Comme $\mathcal{A}$ est émondé, les mots que l'on peut lire d'un état de S à un état de F sont tous dans L. De plus, comme $\mathcal{A}$ est réversible, si $q \in F$, alors les langages $L_{t,q}$ ($t \in I_q$) sont deux à deux disjoints et, par conséquent,

$$\sum_{t \in I_q} |L_{t,q}| \leq 3$$

Considérons l'automate obtenu à partir de $\mathcal{A}$ en prenant pour état initial s et en émondant. Cet automate vérifie alors les hypothèses du lemme 3.25 et l'on a alors l'un des cas suivants (en faisant une disjonction des cas selon le cardinal de F_s) :

(a) $|F_s| = 3$ et $|L_{s,q}| = 1$ pour tout $q \in F_s$.
On a alors la configuration suivante :



Il faut remarquer ici qu'il n'y a pas d'état $t \in I_{q_1}$ tel que $t \cdot b = s \cdot a$: si c'était le cas, on aurait alors $t \cdot bb = q_2$ et donc on aurait $bb \in L_{t,q_2} \subseteq L$, ce qui constitue une contradiction. Comme $\mathcal{A}$ est émondé, il en résulte que

$$ba \notin \bigcup_{t \in I_{q_1}} L_{t,q_1}$$

et que, par conséquent,

$$\sum_{t \in I_{q_1}} |L_{t,q_1}| \leq 2$$

On a déjà vu que

$$\sum_{t \in I_{q_2}} |L_{t,q_2}| \leq 3 \quad \text{et} \quad \sum_{t \in I_{q_3}} |L_{t,q_3}| \leq 3$$

Donc

$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}| = \sum_{q \in F_s} \sum_{t \in I_q} |L_{t,q}| \leq 2 + 3 + 3 = 8$$

(b) $F_s = \{q_1, q_2\}$ avec $|L_{s,q_1}| = 2$ et $|L_{s,q_2}| = 1$. Comme $\mathcal{A}$ est réversible, on a soit $I_{q_1} = \{s\}$, soit $I_{q_1} = \{s, u\}$ avec $|L_{u,q_1}| = 1$.

(b1) Si $I_{q_1} = \{s\}$, on a alors

$$\sum_{t \in I_{q_1}} |L_{t,q_1}||L_{s,q_1}| = 4$$

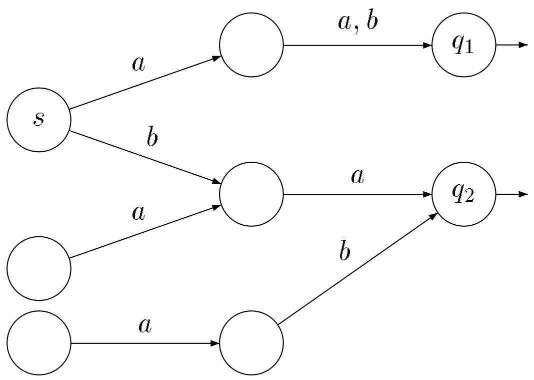
et

$$\sum_{t \in I_{q_2}} |L_{t,q_2}||L_{s,q_2}| = \sum_{t \in I_{q_2}} |L_{t,q_2}| \leq 3$$

D'où

$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}| \leq 4 + 3 = 7$$

Ce cas peut être illustré par la figure suivante :



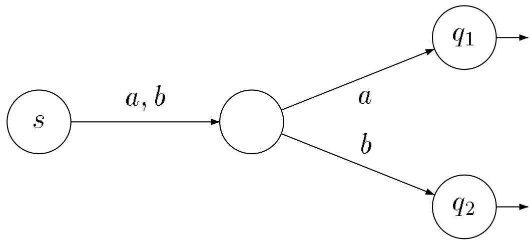
(b2) Supposons que $I_{q_1} = \{s, u\}$. Par hypothèse de réversibilité, on a $L_{u,q_1} = L \setminus L_{s,q_1}$. Comme $|L_{s,q_1}| = 2$, il s'en suit que $|L_{u,q_1}| = 1$, et donc

$$\sum_{t \in I_{q_1}} |L_{s,q_1}||L_{t,q_1}| = 2\,(|L_{s,q_1}| + |L_{u,q_1}|) = 6$$

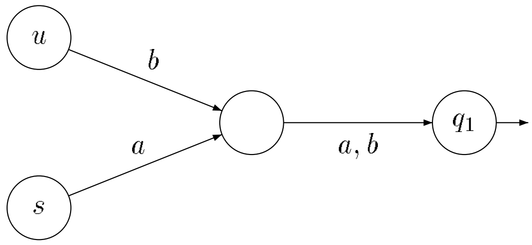
Si $I_{q_2} = \{s\}$, on peut conclure que

$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}| = 6 + 1 = 7$$

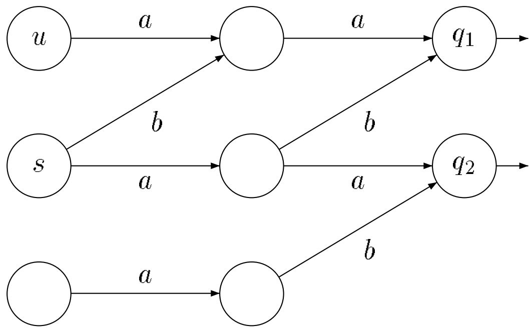
Avant d'aller plus loin, analysons un peu la situation. Si $L_{s,q_1} = \{aa, ba\}$, alors, par hypothèse de réversibilité, on a $s \cdot b = s \cdot a$. De plus, dans ce cas, $L_{s,q_2} = \{ab\}$, et donc $(s \cdot a) \cdot b = q_2$. Cela implique que $s \cdot bb = q_2$, donc que $bb \in L_{s,q_2}$; ce qui est contradictoire.



Maintenant si $L_{s,q_1} = \{aa, ab\}$, alors, par déterminisme, toutes les transitions étiquetées par a ou par b arrivant en q_1 proviennent de $s \cdot a$. De plus, $L_{u,q_1} = \{ba\}$, donc $u \cdot b = s \cdot a$ et on a alors $u \cdot bb = q_1$, ce qui impose que bb appartient à L_{u,q_1} et qui est encore une contradiction. La figure ci-dessous illustre cette situation.



Il en résulte que $aa \notin L_{s,q_1}$ et on a alors $L_{s,q_1} = \{ab, ba\}$, $L_{s,q_2} = L_{u,q_1} = \{aa\}$ et l'on est dans la configuration suivante :



Supposons que $v \in I_{q_2}$, $v \neq s$. Dans ce cas, en utilisant la réversibilité, $aa \notin L_{v,q_2}$. Par conséquent, $ba \notin L_{v,q_2}$ car sinon $v \cdot b = s \cdot a$ et $v \cdot bb = q_1$. Cela implique que $bb \in L_{v,q_1} \subseteq L$; ce qui est encore contradictoire. Donc, si v est différant de s et appartient à I_{q_2}, alors $L_{v,q_2} = \{ab\}$. On peut donc affirmer que

$$\sum_{t \in I_{q_2}} |L_{s,q_2}||L_{t,q_2}| = \sum_{t \in I_{q_2}} |L_{t,q_2}| \leq 2$$

Et donc

$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}| \leq 8$$

(c) $|F_s| = 2$ et $|L_{s,q}| = 1$ pour tout $q \in F_s$. Comme $\sum_{t \in I_q} |L_{t,q}| \leq 3$ pour tout $q \in F$, on a

$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}| = \sum_{q \in F_s} \sum_{t \in I_q} |L_{t,q}| \leq 2 \times 3 = 6$$

(d-e) $F_s = \{q\}$ et $|L_{s,q}| \leq 2$. En utilisant à nouveau l'inégalité

$$\sum_{t \in I_q} |L_{t,q}| \leq 3$$

on obtient

$$\sum_{q \in F_s} \sum_{t \in I_q} |L_{s,q}||L_{t,q}| \leq 6$$

□

Nous allons maintenant prouver la proposition 3.27.

Preuve. On effectue une preuve par récurrence sur n. D'après le lemme 3.28, la proposition est vraie pour $n = 1$.

Supposons maintenant que la propriété est vraie pour n. On considère un automate $\mathcal{A} = (Q, A, E, i, F)$ réversible, émondé et ne possédant qu'un seul état initial i et reconnaissant un sous-langage de L^{n+1}. D'après le lemme 3.29,

$$n_q^2 = \sum_{(s,t) \in S^2} n_s n_t |L_{s,q}||L_{t,q}|. \tag{3.7}$$

On a donc

$$\begin{aligned}
\sum_{q\in F} n_q^2 &= \sum_{q\in F}\sum_{(s,t)\in S^2} n_s n_t |L_{s,q}||L_{t,q}| \\
&= \sum_{(s,t)\in S^2}\left(\sum_{q\in F}|L_{s,q}||L_{t,q}|\right) n_s n_t \\
&\le \frac{1}{2}\sum_{(s,t)\in S^2}\left(\sum_{q\in F}|L_{s,q}||L_{t,q}|\right) n_s^2 + \frac{1}{2}\sum_{(s,t)\in S^2}\left(\sum_{q\in F}|L_{s,q}||L_{t,q}|\right) n_t^2 \\
&\le \sum_{(s,t)\in S^2}\left(\sum_{q\in F}|L_{s,q}||L_{t,q}|\right) n_s^2 = \sum_{s\in S}\left(\sum_{q\in F}\sum_{t\in S}|L_{s,q}||L_{t,q}|\right) n_s^2
\end{aligned}$$

On en déduit, par le lemme 3.30,

$$\sum_{q\in F} n_q^2 \le 8\sum_{s\in S} n_s^2$$

On peut donc appliquer maintenant l'hypothèse de récurrence à l'automate construit à partir de $\mathcal{A}$ en prenant comme ensemble d'états finaux S à la place de F, puis en émondant. On obtient alors

$$\sum_{s\in S} n_s^2 \le 8^n$$

Ce qui prouve le résultat. □

Nous pouvons maintenant prouver le théorème 3.24.

Preuve.

Soit n un entier positif et $\mathcal{A} = (Q, A, E, I, F)$ un automate réversible reconnaissant L^{2n}.

Il est clair que

$$L(\mathcal{A}) = \bigcup_{i\in I, f\in F} L(\mathcal{A}^{i,f}) \tag{3.8}$$

Par la formule (3.2), on a

$$|L(\mathcal{A}^{i,f})| \le \frac{1}{2}\sum_{q\in M(\mathcal{A}^{i,f})}\left(|L_q(\mathcal{A}^{i,f})|^2 + |R_q(\mathcal{A}^{i,f})^{\sim}|^2\right) \tag{3.9}$$

L'automate $(Q, A, E, \{i\}, M(\mathcal{A}^{i,f}))$ reconnaît un sous-langage de L^n. Par la formule (3.3), on a donc

$$\sum_{q \in M(\mathcal{A}^{i,f})} |L_q(\mathcal{A}^{i,f})|^2 \leq 8^n$$

Posons maintenant $E' = \{(p, a_i, q) \mid (q, a_i, p) \in E\}$. L'automate réversible $(Q, A, E', f, M(\mathcal{A}^{i,f}))$ reconnaît aussi un sous-langage de L^n et, d'après (3.3), on a

$$\sum_{q \in M(\mathcal{A}_{if})} |R_q(\mathcal{A}_{if})^{\sim}|^2 \leq 8^n$$

Donc $|L(\mathcal{A}_{if})| \leq 8^n$, par la formule (3.9).

En utilisant alors la formule (3.8), il en résulte que $|L(\mathcal{A})| \leq |I||F|8^n$. Or, $|L(\mathcal{A})| = 9^n$, donc

$$|I||F| \geq \left(\frac{9}{8}\right)^n$$

Il s'en suit que I ou F contient au moins r^n éléments avec $r = \frac{3\sqrt{2}}{4}$. Et comme I et F sont des sous-ensembles de Q, on a aussi $|Q| \geq r^n$.

On vient de prouver que $c(L^{2n}) \geq r^n$. Or $m(L^{2n}) = 6n+1$. Par conséquent, $R_{\text{rev}}(6n+1) = \Omega(r^n)$ et, comme $R_{\text{rev}}(n)$ est une fonction croissante de n, cela prouve le théorème avec $\rho = r^{\frac{1}{6}}$. $\square$

3.5 Conclusion

De nombreuses questions restent ouvertes à propos des langages réversibles, notamment sur l'algorithmique. Le problème majeur est très certainement celui du calcul d'un automate réversible reconnaissant un langage réversible donné par son automate minimal. On a prouvé que la complexité d'un tel algorithme est, en temps et en espace, au moins exponentielle. Il reste cependant une grande marge afin de rapprocher cette borne inférieure de la borne supérieure de [Pin92]. Par ailleurs, d'autres problèmes de décision n'ont pas été résolus. Par exemple, on ne dispose pas de notion d'automate réversible minimal et, par conséquent, la comparaison de deux langages donnés par des automates réversibles n'est pas simple. Peut-on tester plus facilement que dans le cas général s'ils reconnaissent le même langage? Plus simple encore, peut-on facilement tester si un automate réversible reconnaît A^* tout entier?

Il en va de même si l'on se donne un langage par une expression rationnelle ou par un automate non déterministe. Peut-on alors tester s'il est réversible de façon plus efficace qu'en calculant son automate minimal et en lui appliquant le test ? Y a-t-il par ailleurs une forme particulière d'expression rationnelle caractérisant les langages réversibles ?

On a vu qu'un langage est réversible si et seulement s'il est fermé pour la topologie profinie et si les idempotents de son semigroupe syntaxique commutent. La topologie profinie étant la plus fine de toutes les topologies pro-$\mathbf{H}$, un langage dont les idempotents du semigroupe syntaxique commutent et qui est fermé pour une topologie pro-$\mathbf{H}$, sera alors réversible. Il est alors naturel de se demander si un automate réversible le reconnaissant a une caractérisation simple (dépendant de $\mathbf{H}$). Que peut-on dire, par exemple, des langages fermés pour la topologie pro-p et dont les idempotents du monoïde syntaxique commutent ?

Chapitre 4

Hiérarchie des langages à groupe

Sommaire

4.1 Introduction et motivations 141
4.2 Niveau 0 : les langages à groupe 142
4.2.1 Résultats connus sur les langages à groupes 143
4.2.2 Propriétés topologiques 144
4.3 Niveau 1/2 : les langages ouverts 148
4.3.1 Résultats connus sur le niveau 1/2 148
4.3.2 Optimisation du test du niveau 1/2 149
4.3.3 Automates localement à groupe 152
4.4 Niveau 1 153
4.4.1 Résultats connus sur le niveau 1 153
4.4.2 Résultats connus sur la variété **ER** 155
4.4.3 Caractérisation des langages reconnaissables par un automate déterministe localement réversible 156
4.4.4 Amélioration du test du niveau 1 161
4.5 Niveau 3/2 et automates localement réversibles . 161
4.6 Conclusion 164

4.1 Introduction et motivations

Le niveau 0 de la hiérarchie des langages à groupe est constitué, comme son nom l'indique, des langages à groupe. Des trois hiérarchies communément étudiées, c'est la seule qui traite de langages non apériodiques.

Comme nous en avons déjà discuté à la section 1.9, la notion de hiérarchie est une notion qui apparut naturellement en classification des langages rationnels. Le but de la hiérarchie des langages à groupe paraît double. D'une part, et c'est la raison initiale de son introduction, elle est utile en théorie des semigroupes finis [Pin94, Pin95a, MP85]. D'autre part, les langages à groupe ont de très fortes propriétés algébriques qui fournissent de nombreux outils pour les étudier. Comme, de plus, il est montré dans [PW97], que le passage d'un niveau à l'autre d'une hiérarchie est une opération algébrique, on peut donc espérer que certaines propriétés vraies pour les langages à groupe le restent en progressant dans la hiérarchie. L'étude de la hiérarchie des langages à groupe est donc motivée par une volonté de prolonger des propriétés des langages à groupe à des classes de langages plus importantes.

La première utilisation significative des langages à groupe est due à R. McNaughton [McN67] qui montra en 67 que la hauteur d'étoile d'un langage à groupe était décidable, bien avant que K. Hashiguchi montre la décidabilité dans le cas général [Has88]. Remarquons ici que l'algorithme développé par K. Hashigushi n'est, en pratique, pas utilisable, même pour des langages très simples, alors que l'approche de R. McNaughton est de complexité plus "raisonnable".

Comme pour les langages réversibles, les langages à groupe jouent un rôle prépondérant dans l'approche quantique de la calculabilité au moyen d'automates finis, comme l'ont montré A. Brodsky et N. Pippenger dans [BP99].

On peut aussi citer ici les travaux de J.-C. Birget [Bir92], qui utilisa les automates à groupe pour analyser la complexité des algorithmes de base sur les automates finis.

Signalons aussi les travaux récents de C. Nicaud [Nic00] qui a commencé l'étude de l'analyse en moyenne des algorithmes sur les automates finis et qui a obtenu des résultats significatifs dans ce domaine pour les automates à groupe.

Enfin, et cela doit transparaître dans cette thèse, les langages à groupe sont primordiaux pour l'étude topologique des langages rationnels et ont des applications en théorie des semigroupes (voir à ce sujet [AD99, Del98, Pin88, Ste00d, Ste00c] et aussi le chapitre 6).

Dans ce chapitre allons étudier la hiérarchie des langages à groupe niveau par niveau (jusqu'au niveau 3/2), en précisant pour chacun les propriétés topologiques ainsi que les propriétés de représentation par des automates particuliers.

4.2 Niveau 0 : les langages à groupe

Nous étudions dans cette section les langages à groupe, notamment leurs propriétés topologiques.

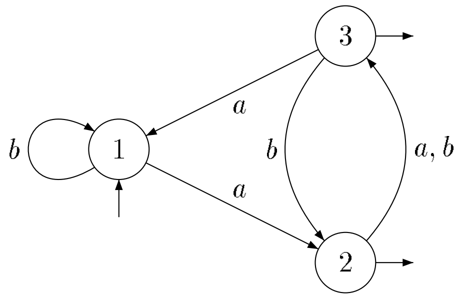
4.2.1 Résultats connus sur les langages à groupes

Intéressons nous pour commencer aux propriétés de stabilité des langages à groupe. Ces résultats sont bien connus, sans qu'il y ait de référence particulière à leur sujet. Rappelons qu'un langage à groupe est un langage reconnaissable par un automate à groupe, c'est-à-dire un automate dans lequel chaque lettre induit une bijection de l'ensemble des états dans lui-même. Rappelons aussi qu'un langage est à groupe si et seulement si son monoïde syntaxique est un groupe, ou encore si et seulement si son automate minimal est un automate à groupe.

Proposition 4.1 *La classe des langages à groupe est stable par union, intersection et complémentation, mais n'est stable ni par produit ni par étoile.*

Preuve. La stabilité par union, intersection et complémentation, découle du fait que **G** est une variété de monoïde. Pour le produit de langage à groupe, il suffit de considérer les deux langages à groupe $(A^2)^*$ et $(A^3)^*$. Leur produit $(A^2)^*(A^3)^*$ est exactement le langage $A^* \setminus A$, qui n'est pas un langage à groupe car son complémentaire, qui est fini, ne l'est pas.

Pour l'opération étoile, considérons le langage L dont l'automate minimal est :



Cet automate est à groupe donc L est un langage à groupe.

Pour tout $n > 1$, $a^n b \in L^*$, donc $a^{n!} b \in L^*$. Il en résulte que $b \in \mathrm{Cl}_{A^*}(L)$, d'après la proposition 2.7. Or tout langage à groupe étant fermé pour la topologie profinie de A^*, si L^* était à groupe, il en résulterait que $b \in L^*$, ce qui n'est pas le cas. $\square$

4.2.2 Propriétés topologiques

Nous commentons ici les propriétés topologiques des langages à groupe. Un certain nombre d'entres elles ont été données au chapitre 2. Rappelons qu'un langage rationnel est à la fois ouvert et fermé, pour la topologie pro-$\mathbf{H}$, si et seulement si c'est un langage à groupe dont le monoïde syntaxique appartient à $\mathbf{H}$ et que ces langages forment une base d'ouverts pour la topologie pro-$\mathbf{H}$. Par ailleurs, la fermeture d'un langage rationnel ouvert est un langage rationnel à la fois ouvert et fermé.

Nous allons montrer maintenant qu'il existe des langages à la fois ouverts et fermés (pour la topologie profinie), qui ne sont pas rationnels. Ce résultat n'est pas, en soi, très surprenant et montre une des limites des liens entre la topologie profinie et les langages rationnels.

Pour cela, introduisons, pour les trois propositions suivantes, les langages :
- $L_p = \{u \in A^* \mid |u| = (2k+1)2^p,\ k \in \mathbb{N}\}$
- $K = \{u \in A^* \mid |u| = 3k, k \in \mathbb{N}\}$
- $\mathcal{P} = \{p \in \mathbb{N} \mid p \text{ premier et } p > 2\}$
- $L = \bigcup_{p \in \mathcal{P}} L_p$ et $R = L \cup K$

Proposition 4.2 *Les langages K et L_p (pour tout p appartenant à $\mathbb{N}$) sont à la fois ouverts, fermés (pour la topologie profinie) et rationnels.*

Preuve. En effet, considérons le morphisme de monoïde

$$\psi \ : \ \begin{cases} A^* & \rightarrow \quad \mathbb{Z}/3\mathbb{Z} \\ u & \mapsto \quad |u| \mod 3 \end{cases}$$

On a alors $K = \{0\}\psi^{-1}$, qui est donc un langage à groupe (et par conséquent rationnel, ouvert et fermé).

De la même manière, considérons les morphismes de monoïdes :

$$\varphi_p \ : \ \begin{cases} A^* & \rightarrow \quad \mathbb{Z}/2^{p+1}\mathbb{Z} \\ u & \mapsto \quad |u| \mod 2^{p+1} \end{cases}$$

Les langages $L_p = \{2^p\}\varphi_p^{-1}$ sont des langages à groupe et sont donc ouverts, fermés et rationnels. □

Proposition 4.3 *Le langage R est à la fois ouvert et fermé pour la topologie profinie.*

Preuve. Le langage R étant défini par une union de langages ouverts, il est lui-même ouvert.

Prouvons maintenant que $\mathrm{Cl}_{A^*}(L) \subseteq L \cup \{\varepsilon\}$. En effet, si (u_n) est une sous-suite de L convergeant vers $u \in A^*$, alors on est dans l'un des cas suivants (on distingue deux cas suivant que (u_n) est à valeur dans un nombre fini ou infini de L_p) :

- *Cas 1 :* Il existe un sous-ensemble fini $\mathcal{Q}$ de $\mathcal{P}$ tel que

$$\forall n \in \mathbb{N}, \quad u_n \in \bigcup_{p \in \mathcal{Q}} L_p$$

Comme chaque L_p est fermé, $\bigcup_{p \in \mathcal{Q}} L_p$ l'est aussi. Par conséquent,

$$u \in \bigcup_{p \in \mathcal{Q}} L_p \subseteq L$$

- *Cas 2 :* Il existe un nombre infini de p tels que $\{u_n \mid n \in N\} \cap L_p$ soit non vide. La même propriété est alors vérifiée pour toutes les sous-suites $(u_{n \geq k})$. Donc

$$\forall k \in \mathbb{N}, \quad \forall P \in \mathbb{N}, \quad \exists q \in \mathcal{P}, \quad \exists n \geq k \quad \text{tel que } u_n \in L_q \text{ et } q > P \tag{4.1}$$

Le mot u est la limite de (u_n), donc

$$\forall p \in \mathcal{P}, \quad \exists N_p \in \mathbb{N}, \quad \text{tel que } \forall n \geq N_p, \quad u_n\varphi_p = u\varphi_p \tag{4.2}$$

Soit $p \in \mathcal{P}$. On prend dans (4.1) $P = p$ et $k = N_p$ (où N_p est défini dans (4.2)). Il en résulte qu'il existe $q \in \mathcal{P}$ tel que, d'une part, $q > p$ et, d'autre part, il existe $n_p \geq N_p$ vérifiant $u_{n_p} \in L_q$.
Par conséquent, $u_{n_p} \in L_q$ et, par (4.2), on obtient

$$u_{n_p}\varphi_p = u\varphi_p$$

Il en résulte qu'il existe i et j dans $\mathbb{N}$ tels que

$$|u_{n_p}| = (2i+1)2^q \quad \text{and} \quad |u_{n_p}| = |u| + j2^{p+1}$$

Or $p < q$, donc 2^p divise 2^q et, par conséquent, 2^p divise $|u|$.
Comme cela est vrai pour tout $p \in \mathcal{P}$, $|u| = 0$ et $u = \varepsilon$.

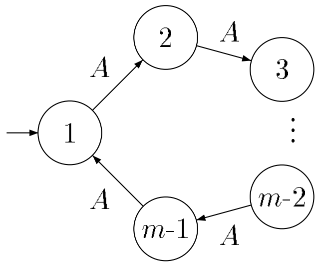
On en déduit que

$$\mathrm{Cl}_{A^*}(R) \subseteq L \cup \{\varepsilon\} \cup \mathrm{Cl}_{A^*}(K)$$

Or $\mathrm{Cl}_{A^*}(K) = K$ et $\varepsilon \in K$, donc $\mathrm{Cl}_{A^*}(R) \subseteq R$, ce qui montre que R est fermé.
Par conséquent, R est à la fois ouvert et fermé. □

Proposition 4.4 *Le langage R n'est pas rationnel.*

Preuve. Nous allons montrer le résultat par l'absurde. Supposons donc que R est un langage rationnel. Comme il est à la fois ouvert et fermé, c'est un langage à groupe. Mais, comme on a juste besoin de connaître la longueur d'un mot pour savoir s'il appartient à R, chaque lettre de A induit la même transformation dans l'automate minimal de R. Par conséquent, l'automate minimal de R est une boucle dont on note m la longueur.



Automate minimal de R

On pose $m = 2^r n$ avec n impair. On a alors un des cas suivants (selon que r appartient ou non à $\mathcal{P}$).

- *Cas 1:* $r \notin \mathcal{P}$.
  Soient alors $p \in \mathcal{P}$ tel que $p > r + 1$ et $s = p - r$. Les entiers n et 2^{s+1} sont premiers entre eux, donc il existe deux entiers relatifs t et ℓ tels que

$$tn + \ell 2^{s+1} = 1$$

Donc, pour tout $i \in \mathbb{N}$, on a, en multipliant par $(2^s - 1 - 2i)$:

$$(2^s - 1 - 2i)tn + (2^s - 1 - 2i)\ell 2^{s+1} = 2^s - 1 - 2i$$

Si $\ell \leq 0$, alors on choisit $i = 0$. Sinon on choisit i tel que $(2^s - 1 - 2i) < 0$ et tel que $1+2i$ ne soit pas un multiple de 3 (ce qui est toujours possible). Enfin, soit $j = -(2^s - 1 - 2i)\ell$ et $k = (2^s - 1 - 2i)t$. Le choix fait pour i implique que j est positif. De plus,

$$kn - j2^{s+1} = 2^s - 1 - 2i$$

Donc

$$kn - (1 + 2j)2^s = -1 - 2i$$

Par conséquent,

$$kn + (1 + 2i) = (1 + 2j)2^s$$

En multipliant par 2^r, on obtient alors

$$km + (1+2i)2^r = (1+2j)2^p$$

Comme j est positif, si $a \in A$, alors $a^{(1+2j)2^p} \in L_p \subseteq R$.
Or, l'automate minimal de R est une boucle de taille m, et $km+(1+2i)2^r$ est positif, donc l'état que l'on atteint en lisant $a^{(1+2i)2^r+km}$ est le même que celui qu'on atteint en lisant $a^{(1+2i)2^r}$. Par conséquent, $a^{(1+2i)2^r}$ est un mot de R. Mais, comme on a choisi i tel que $2i+1$ ne soit pas un multiple de 3, $r \in \mathcal{P}$ et on a une contradiction.

- *Cas 2 :* $r \in \mathcal{P}$.
  Dans l'automate minimal de R, l'état que l'on atteint en lisant $a^{m+2^{r+1}}$ est le même que celui que l'on atteint en lisant $a^{2^{r+1}}$. Mais deux éléments de $\mathcal{P}$ ne sont jamais consécutifs, donc $r+1 \notin \mathcal{P}$. Il s'en suit que $i \cdot a^{2^{r+1}}$ n'est pas final (i est l'état initial de l'automate minimal de R). De plus, $m + 2^{r+1} = (n+2)2^r$ et $n+2$ est impair, donc $i \cdot a^{m+2^{r+1}}$ est un état final, ce qui est contradictoire.

Il en résulte que R n'est pas rationnel. □

Terminons cette section par une propriété des langages à groupe, concernant toutes les topologies pro-**H**, où **H** est une variété fermée par extension.

Proposition 4.5 *Soit L un langage à groupe et* **H** *une variété de groupes finis close par extension. La fermeture pro-***H** *de L est un langage à groupe dont le monoïde syntaxique est dans* **H**.

Preuve. Soit L un langage à groupe et $\mathcal{A} = (Q, A, E, i, F)$ son automate minimal. Comme L est l'union des $L(\mathcal{A}^{i,f})$ lorsque f décrit F et comme l'union de deux langages à groupe dont le monoïde syntaxique est dans **H**, est aussi un langage à groupe dont le monoïde syntaxique est dans **H** , il suffit de prouver le résultat pour $L(\mathcal{A}^{i,f})$.

Soit $u \in A^*$ tel que $i \cdot u = f$. Comme un automate à groupe est fortement connexe, on a montré au lemme 2.41, que

$$\mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A}^{i,f})) = \mathrm{Cl}_F^{\mathbf{H}}\ (\langle L(\mathcal{A}^{i,i})\rangle)u$$

Comme on identifie A^* avec un sous-monoïde de $F(A)$, on peut remarquer ici que $L(\mathcal{A}^{i,f})$ et $L(\mathcal{A}^{i,f})\pi$ sont identifiés. On pose $H = \langle L(\mathcal{A}^{i,i})\rangle$. Le sous-groupe H est un sous-groupe rationnel de $F(A)$ reconnu par l'automate dual de $\mathcal{A}^{i,i}$ (voir proposition 2.36). Or $\mathcal{A}^{i,i}$ est complet, donc $\langle L(\mathcal{A}^{i,i})\rangle$ est un sous-groupe d'indice fini de $F(A)$. Par conséquent, il existe $v_1, \cdots, v_k \in F(A)$ tels que

$$Hv_1 \cup Hv_2 \cup \cdots \cup Hv_k = F(A)$$

Donc

$$\mathrm{Cl}_F^{\mathbf{H}}(H)v_1 \cup \mathrm{Cl}_F^{\mathbf{H}}(H)v_2 \cup \cdots \cup \mathrm{Cl}_F^{\mathbf{H}}(H)v_k = F(A)$$

Par conséquent, comme tous les $\mathrm{Cl}_F^{\mathbf{H}}(H)v_i$ sont fermés pour la topologie pro-$\mathbf{H}$, et qu'une union finie de fermés est aussi fermée, tous les $\mathrm{Cl}_F^{\mathbf{H}}(H)v_i$ sont aussi ouverts. De plus, il existe v_i tel que $\mathrm{Cl}_F^{\mathbf{H}}(H)v_i = \mathrm{Cl}_F^{\mathbf{H}}(H)u$. Il en résulte que $\mathrm{Cl}_F^{\mathbf{H}}(L(\mathcal{A}^{i,f}))$ est à la fois ouvert et fermé dans $F(A)$, donc $\mathrm{Cl}_{A^*}^{\mathbf{H}}(L(\mathcal{A}^{i,f}))$ est un langage à groupe dont le monoïde syntaxique est dans $\mathbf{H}$ (en utilisant le théorème 2.17). □

4.3 Niveau 1/2 : les langages ouverts

4.3.1 Résultats connus sur le niveau 1/2

On peut citer deux résultats importants sur le niveau 1/2 de la hiérarchie des langages à groupe. Le premier met en valeur un lien avec la topologie profinie et fut prouvé par J.-E. Pin [Pin94] il correspond aux assertions (i) et (ii) du théorème ci-dessous). Le second résultat, dû à M. Beaudry, F. Lemieux et D. Thérien [BLT97], est assez surprenant : de même que l'on peut définir un langage reconnaissable par un monoïde, on peut définir un langage reconnaissable par une structure finie assez simple (un ensemble muni d'une loi binaire non associative), appelée *groupoïde*, et qui, dans le cas général, reconnaît exactement tous les langages algébriques (une classe de langages plus large que la classe des langages rationnels). M. Beaudry et al. ont montré que si l'on s'intéresse aux langages reconnus par des *loops*, c'est-à-dire des groupoïdes pour lesquels chaque élément possède des inverses, alors les langages reconnus sont non seulement rationnels, mais sont de plus exactement les langages de niveau 1/2 de la hiérarchie des langages à groupe. Une conséquence de ce résultat est l'assertion (iv) du théorème ci-dessous. Les résultats de ces deux études peuvent se résumer par le théorème suivant.

Théorème 4.6 *Soit L un langage rationnel. Les assertions suivantes sont équivalentes.*

(i) L est un langage de niveau 1/2 de la hiérarchie des langages à groupe.

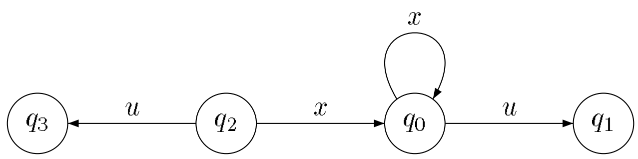
(ii) L est ouvert pour la topologie profinie.

(iii) Il existe un langage à groupe R contenant ε tel que L soit une union finie de langages du type $Ra_1Ra_2\cdots a_kR$, où les a_i sont des lettres.

(iv) L est union finie de langages du type $L_0L_1\cdots L_k$ où les L_i sont des langages à groupe.

La caractérisation syntaxique des langages ouverts donnée à la proposition 2.21 permet de tester si un langage est de niveau 1/2 dans la hiérarchie des langages à groupe.

Proposition 4.7 *Soit L un langage rationnel et $\mathcal{A}$ le complété de son automate minimal. Le langage L est de niveau 1/2 dans la hiérarchie des langages à groupe si et seulement s'il n'y a dans $\mathcal{A}$ aucune configuration du type :*



avec $u, v, x \in A^*$, $q_3 \in F$ et $q_1 \notin F$.

On peut tester si ce motif apparaît dans un automate $\mathcal{A}$ en calculant son carré $\mathcal{A}_1$, puis en calculant la clôture transitive (V, E) du graphe non étiqueté sous-jacent à $\mathcal{A}_1$. On montre alors [PW97] que $\mathcal{A}$ n'a aucune configuration du type ci-dessus si et seulement si on n'a aucune paire d'arcs dans (V, E) du type

$$\{((q_2, q_0), (q_0, q_0)), ((q_2, q_0), (q_3, q_1))$$

avec $q_3 \in F$ et $q_1 \notin F$. Cet algorithme demande donc un temps de calcul supérieur à n^4 (car on doit calculer la clôture transitive d'un graphe à n^2 sommets).

4.3.2 Optimisation du test du niveau 1/2

On peut obtenir un meilleur algorithme en utilisant les théorèmes 4.6 et 2.59.

Corollaire 4.8 *On peut tester en temps $O(n^3)$ si un langage donné par un automate déterministe à n états est dans le niveau 1/2 de la hiérarchie des langages à groupe.*

Pour cela, on utilise l'algorithme `TesteG1/2` suivant, qui teste si un langage rationnel est ouvert pour la topologie profinie et qui s'applique à un automate déterministe complet $\mathcal{A} = (Q, A, E, i, F)$.

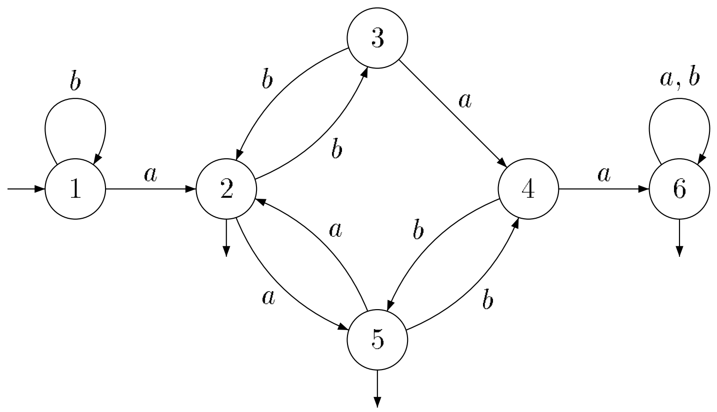
Algorithme 4.9

Retourner <u>`TesteFerméGA*`</u>$((Q, A, E, i, Q \setminus F))$.

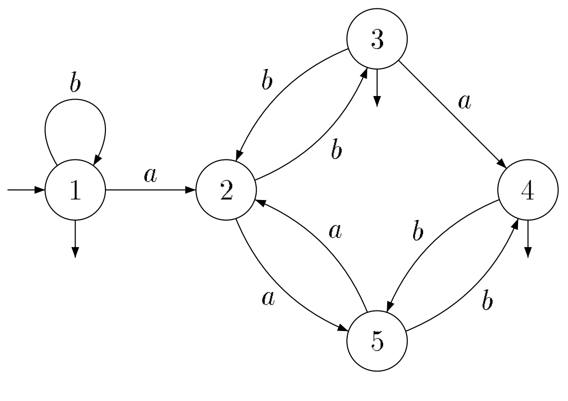
$\diamondsuit$

Donnons un exemple d'utilisation de cet algorithme.

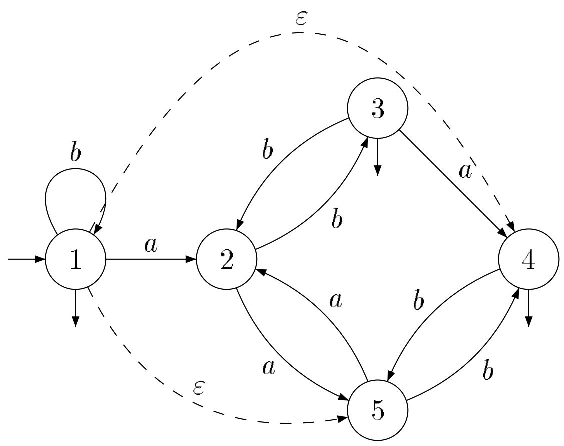
Exemple 4.10 Considérons le langage L donné par son automate minimal $\mathcal{A}$ suivant :



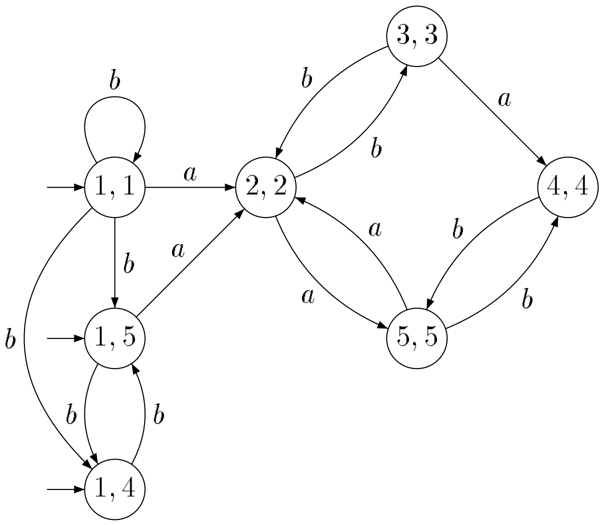
On applique l'algorithme <u>`TesteFerméGA*`</u> à l'automate du complémentaire de L suivant :



La fermeture profinie de L^c est reconnue par l'automate suivant (voir le chapitre 2 pour les détails) :



Après élimination des ε-transitions, on trouve que $\mathrm{Cl}_{A^*}(L^c) \cap L$ est reconnu par l'automate suivant :



Cet automate reconnaît l'ensemble vide car il ne possède aucun état final. Donc L^c est fermé pour la topologie profinie et, par conséquent, L est ouvert.

$\triangledown$

4.3.3 Automates localement à groupe

On peut déduire de la proposition 3.20 le corollaire suivant.

Corollaire 4.11 *Tout langage de niveau 1/2 de la hiérarchie des langages à groupe est reconnaissable par un automate localement réversible déterministe.*

Il ne s'agit pas là d'une propriété caractéristique des langages de niveau 1/2 de la hiérarchie des langages à groupe, car, par exemple, les langages finis ont la même propriété.

Afin de caractériser par des automates les langages de niveau 1/2 de la hiérarchie des langages à groupe, on peut introduire, de même que pour les automates réversibles, la notion d'automate *localement à groupe*.

localement à groupe

Définition 4.12 *Un automate est dit* localement à groupe *si chaque lettre induit une permutation des états de chacune de ses composantes fortement connexes.*

On obtient alors une traduction (partielle) du théorème 4.6.

Proposition 4.13 *Un langage rationnel est ouvert pour la topologie profinie si et seulement si il est reconnaissable par un automate localement à groupe.*

Preuve. On montre qu'un automate localement réversible reconnaît un langage de niveau 1/2 en utilisant la même méthode que dans la preuve de la proposition 3.20.

Considérons maintenant un langage de niveau 1/2. Il peut s'exprimer comme une union finie de langages du type

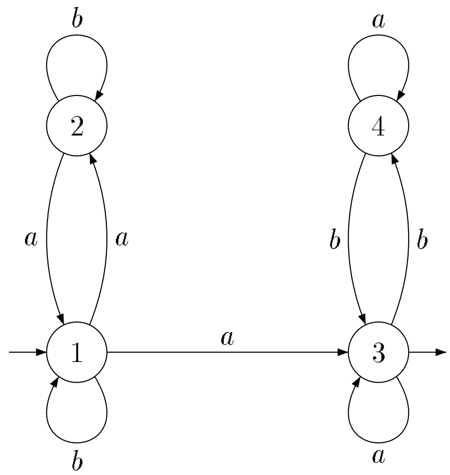
$$K = L_0 a_1 L_1 \dots a_n L_n$$

où les L_k sont des langages à groupes et les a_k des lettres.
Soit $\mathcal{A}_k = (Q_k, A, E_k, i_k, F_k)$ les automates minimaux des langages L_k (on suppose que les ensembles Q_k sont deux à deux disjoints). L'automate

$$\left(\bigcup_{k=0}^{n} Q_k, A, (\bigcup_{k=0}^{n-1}\{(p, a_k, i_{k+1}) \mid p \in F_k\}) \cup \bigcup_{k=0}^{n} E_k, i_0, F_n\right)$$

est localement à groupe (ses composantes fortement connexes sont exactement les (Q_k, E_k)) et reconnaît K. Comme une union d'automates localement à groupe est localement à groupe, il en résulte que tout langage de niveau 1/2 est reconnaissable par un automate localement à groupe. □

Exemple 4.14 Par exemple, le langage L de l'exemple 4.10 est reconnaissable par l'automate localement à groupe suivant :



▽

La preuve du théorème 4.6 repose sur des arguments de théorie des semigroupes qui ne donnent pas de moyen efficace de calculer, pour un langage de niveau 1/2, un automate localement à groupe reconnaissant le langage : comme pour les automates réversibles les bornes données sont trop importantes pour permettre un calcul en pratique.

4.4 Niveau 1 et automates déterministes localement réversibles

4.4.1 Résultats connus sur le niveau 1

La décidabilité du niveau 1 résulte de résultats profonds de la théorie des semigroupes finis [Pin95a, MP85] et fut longtemps un problème ouvert.

Dans [MP85], S. Margolis et J.-E. Pin montrent qu'un langage est de niveau 1 dans la hiérarchie des langages à groupe si et seulement si son monoïde syntaxique appartient à la variété de monoïdes finis **PG** (pour "Power Group"), qui est la variété engendrée par tous les monoïdes finis de la forme $(\mathcal{P}(G), \cdot)$, où G est un groupe fini. Ils montrent de plus que les variétés de monoïdes finis **BG** (monoïdes dont les blocs sont des groupes, voir [MP85]), et **EJ** (monoïdes finis dont les idempotents engendrent un monoïde $\mathcal{J}$-trivial) sont égales et contiennent **PG**. Ce qui peut se résumer par :

$$\mathbf{PG} \subseteq \mathbf{BG} = \mathbf{EJ}$$

La question de savoir si cette inclusion est stricte resta longtemps ouverte et fut finalement démontré par K. Henckel et J. Rhodes [HR91] (voir aussi [Pin95a]) : on a en effet

$$\mathbf{PG} = \mathbf{BG} = \mathbf{EJ}$$

ce qui assure la décidabilité du niveau 1.

Théorème 4.15 *Soit L un langage rationnel. On a équivalence entre les assertions suivantes :*

(i) L est un langage de niveau 1 dans la hiérarchie des langages à groupes.

(ii) $M(L)$ appartient à **PG**.

(iii) $M(L)$ appartient à **BG**.

(iv) $M(L)$ appartient à **EJ**.

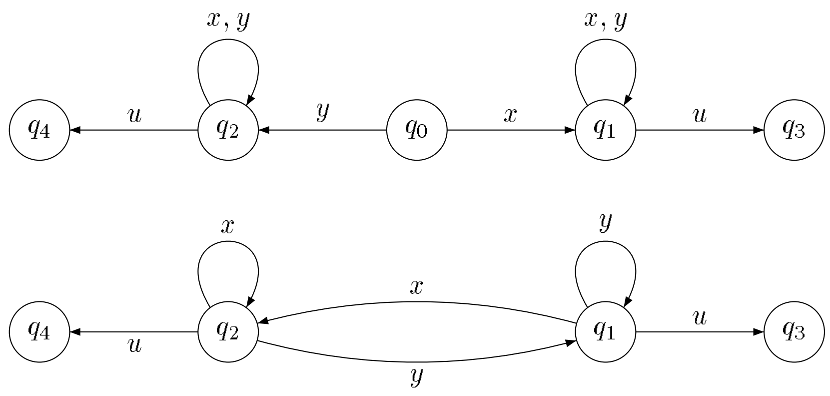
(v) Quels que soient x et y dans $M(L)$, $(x^\omega y^\omega)^\omega = (y^\omega x^\omega)^\omega$.

Cela conduit au résultat suivant [PW97].

Théorème 4.16 *Soient L un langage de A^* et $\mathcal{A}$ le complété de son automate minimal. Les assertions suivantes sont équivalentes :*

(i) L est dans le niveau 1 de la hiérarchie des langages à groupe.

(ii) $\mathcal{A}$ ne contient aucune configuration d'un des deux types suivants :



où $x, y, u \in A^*$, q_3 est un état non final et q_4 un état final.

Appelons $\mathcal{A}_1$ l'automate obtenu en faisant le cube d'un automate déterministe $\mathcal{A}$ et appelons (V_1, E_1) la clôture transitive du graphe non étiqueté sous-jacent à $\mathcal{A}_1$. Alors $\mathcal{A}$ n'a aucune configuration du premier type ci-dessus si et seulement si il n'y a pas dans E_1 de quadruplet d'arcs du type

$$\{((q_0, q_1, q_2), (q_2, q_1, q_2)), ((q_0, q_1, q_2),(q_1, q_1, q_2)), \\ ((q_1, q_1, q_2),(q_3, q_3, q_4)), ((q_2, q_1, q_2), (q_4, q_3, q_4))\}$$

avec $q_4 \in F$ et $q_3 \notin F$.

De même, en appelant $\mathcal{A}_2$ l'automate obtenu en faisant le carré d'un automate déterministe $\mathcal{A}$ et (V_2, E_2) la clôture transitive du graphe non étiqueté sous-jacent à $\mathcal{A}_2$, $\mathcal{A}$ n'a aucune configuration du second type ci-dessus si et seulement si il n'y a pas dans E_2 de triplet d'arcs du type

$$\{((q_1, q_2), (q_2, q_2)), ((q_1, q_2), (q_1, q_1)), ((q_1, q_2), (q_3, q_4))\}$$

avec $q_4 \in F$ et $q_3 \notin F$.

L'algorithme qui découle du théorème 4.16 nécessite donc de calculer la clôture transitive d'un graphe à n^3 sommets et elle est donc de complexité supérieure à n^6.

4.4.2 Résultats connus sur la variété ER

L'objectif est maintenant de caractériser la classe des langages reconnaissables par un automate déterministe localement réversible en la situant précisément vis-à-vis de la hiérarchie des langages à groupe. Pour cela nous allons utiliser la décomposition unitaire-préfixielle vue à la section 1.10 et avons besoin des quelques résultats suivants.

Proposition-Définition 4.17 *Un monoïde M est dit* $\mathcal{R}$-trivial *si quels que soient x, y et z appartenant à M, $x = xyz$ implique $x = xy$. La classe des monoïdes finis $\mathcal{R}$-triviaux est une variété de monoïdes finis notée* **R**. $\mathcal{R}$-trivial

Pour une description des langages dont le monoïde syntaxique est $\mathcal{R}$-trivial voir section 5.3.2.

Les propositions suivantes sont prouvées dans [Eil76].

Proposition-Définition 4.18 *On note* **ER** *la classe de monoïdes finis M pour lesquels les idempotents engendrent un monoïde $\mathcal{R}$-trivial. La classe* **ER** *est une variété de monoïdes finis.*

On va montrer dans la section suivante que la classe **ER** est la variété de monoïdes finis associée à la variété des langages reconnaissable par un automate déterministe localement réversible.

Proposition 4.19 *Le monoïde syntaxique d'un langage L appartient à* **ER** *si et seulement si les monoïdes unitaires de sa décomposition unitaire-préfixielle sont réversibles.*

Proposition 4.20 *Soit $(M_0, a_1, M_1, \ldots, a_k, M_k)$ un monôme unitaire-préfixiel. Si chacun des M_i est réversible, alors le monoïde syntaxique du langage associé appartient à* **ER**.

4.4.3 Caractérisation des langages reconnaissables par un automate déterministe localement réversible

Intéressons nous à la représentation des langages de niveau 1 par des automates particuliers.

Proposition 4.21 *La classe des langages reconnaissables par un automate déterministe localement réversible est stable par les opérations booléennes.*

PREUVE. Il suffit de montrer que la classe des langages reconnaissables par un automate déterministe localement réversible est close par intersection et complémentaire.

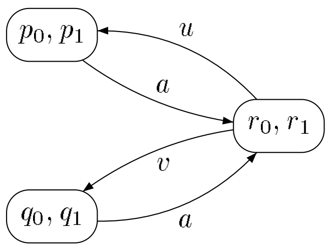
En effet, si $\mathcal{A}$ est un automate déterministe localement réversible, alors son complété l'est aussi. En effet, si $\mathcal{A}$ n'est pas complet, en le complétant on ne modifie pas ses composantes fortement connexes, on en rajoute seulement une contenant un unique état, donc réversible. Par conséquent, comme la propriété d'être localement réversible ne dépend pas des états finaux, la classe des langages reconnaissables par un automate déterministe localement réversible est stable par passage au complémentaire.

Soient $\mathcal{A}_0 = (Q_0, A, E_0, i_0, F_0)$ et $\mathcal{A}_1 = (Q_1, A, E_1, i_1, F_1)$ deux automates déterministes localement réversibles. Prouvons que le produit $\mathcal{A}$ de ces automates est aussi déterministe localement réversible. Comme le produit de deux automates déterministes est déterministe (voir chapitre 1), $\mathcal{A}$ est déterministe. Supposons qu'il existe (p_0, p_1), (q_0, q_1) et (r_0, r_1) des éléments de $Q_0 \times Q_1$ et une lettre a, tels que, d'une part,

$$(p_0, p_1) \cdot a = (q_0, q_1) \cdot a = (r_0, r_1)$$

et, d'autre part, il existe u et v dans A^* tels que

$$(r_0, r_1) \cdot u = (p_0, p_1) \quad \text{et} \quad (r_0, r_1) \cdot v = (q_0, q_1)$$



Dans ce cas $p_i \cdot a = q_i \cdot a = r_i$ et $r_i \cdot u = p_i$ et $r_i \cdot v = q_i$ (avec $i = 0, 1$), donc, comme $\mathcal{A}_0$ et $\mathcal{A}_1$ sont localement réversibles, $p_0 = q_0$ et $p_1 = q_1$.

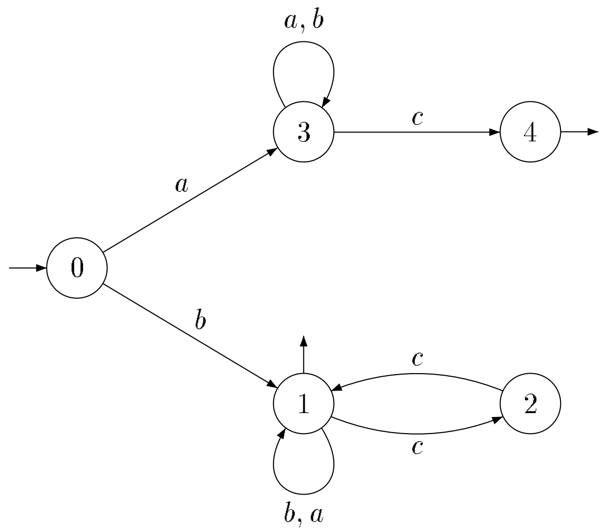
Il en résulte que $\mathcal{A}$ est déterministe et localement réversible. Donc, la classe des langages reconnaissables par un automate déterministe localement réversible est stable par intersection, ce qui termine la preuve. $\square$

On en déduit les résultats suivant.

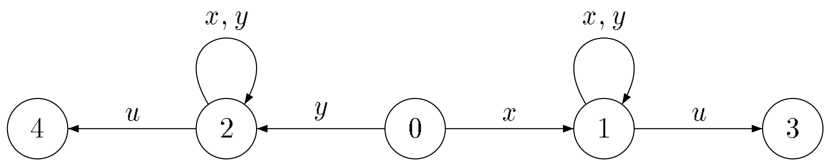
Proposition 4.22 *Tout langage de niveau* 1 *de la hiérarchie des langages à groupe est reconnaissable par un automate déterministe localement réversible.*

PREUVE. Le niveau 1 de la hiérarchie des langages à groupe est la clôture booléenne du niveau 1/2. Or, d'après le corollaire 4.11, tout langage de niveau 1/2 est reconnaissable par un automate déterministe localement réversible. On conclut en utilisant la proposition 4.21 □

On peut utiliser le théorème 4.16 pour montrer que l'inclusion précédente est stricte. Considérons le langage L dont l'automate minimal est le suivant :



Cet automate est localement réversible et déterministe. On a cependant la configuration suivante :



avec $x = b$, $y = a$, $u = c$ et l'état 3 n'est pas final alors que l'état 4 l'est. Ce qui prouve, par le théorème 4.16, que L n'est pas dans le niveau 1 de la hiérarchie des langages à groupe.

Corollaire 4.23 *Tout langage de niveau 1 de la hiérarchie des langages à groupe est une union finie de langages du type $R_1R_2 \cdots R_k$, où les R_k sont des langages réversibles.*

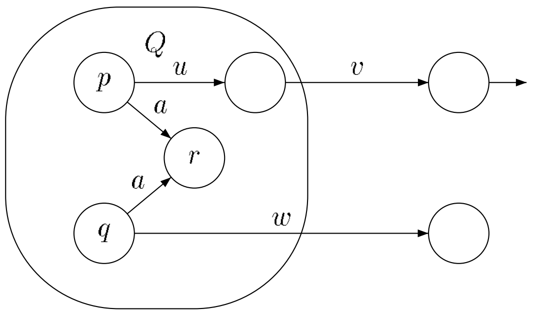
Preuve. Il suffit de prouver que tout langage reconnaissable par un automate localement réversible est reconnaissable par une union finie de produits de langages réversibles. Cette preuve est identique à celle de la proposition 3.20. □

Avant de caractériser les langages reconnaissables par un automate déterministe localement réversible, montrons le résultat suivant qui découle de la proposition 4.19.

Proposition 4.24 *L'automate minimal d'un langage dont le monoïde syntaxique appartient à* **ER** *est localement réversible.*

Preuve. Soit L un langage dont le monoïde syntaxique appartient à **ER** et $\mathcal{A}_0$ son automate minimal. D'après la proposition 4.19, pour tout monôme unitaire-préfixiel $(M_0, a_1, M_1, \cdots, a_k, M_k)$ de la décomposition de $L(\mathcal{A}_0)$, chaque M_j est réversible.

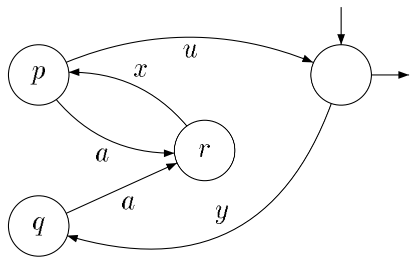
Supposons que $\mathcal{A}_0 = (Q_0, A, E_0, i_0, F_0)$ n'est pas localement réversible. On peut donc supposer qu'il existe deux états p et q distincts d'une même composante fortement connexe (Q, E) et $a \in A$ tels que $p \cdot a = q \cdot a \in Q$. Comme $\mathcal{A}_0$ est minimal il existe w tel que $p \cdot w$ soit final mais pas $q \cdot w$ (ou le contraire). Soit u le plus long préfixe de w tel que $p \cdot u \in Q$ et v tel que $uv = w$. Soit m un chemin sans boucle de $\mathcal{A}$ allant de $p \cdot u$ à $p \cdot w$. Soit aussi $a_0a_1 \cdots a_k = [m]$ l'étiquette de m.



Alors, lors du calcul de la décomposition unitaire-préfixielle de $\mathcal{A}_0$, lorsque l'on examine le calcul qui part de $p{\cdot}w$ et qui "remonte" m, c'est-à-dire qui commence par calculer $L((Q_0, A, E_0, p{\cdot}w, p{\cdot}w))$, puis $L((Q_0, A, E_0, p{\cdot}a_0a_1 \cdots a_{k_1}, p{\cdot} a_0a_1 \cdots a_{k_1}))$, etc, on en vient à calculer le monoïde unitaire

$$M = L((Q_0, A, E_0, p \cdot u, p \cdot u)) = L((Q, A, E, p \cdot u, p \cdot u))$$

Comme (Q, E) est un graphe fortement connexe, il existe x et y dans A^* tels que $q \cdot ax = p$ et $p \cdot uy = q$.



Pour tout entier $n \geq 0$ on a,

$$ya(xa)^n xu \in M$$

En particulier, pour tout entier $n \geq 0$ on a,

$$y(xa)^{n!}u \in M$$

Comme M est réversible (proposition 4.19), M est fermé pour la topologie profinie (proposition 3.14), donc $yu \in M$. Par conséquent $q \cdot u = p \cdot u$ et donc $q \cdot w \in F$, ce qui est contradictoire avec les hypothèses. □

Nous allons maintenant donner une caractérisation précise des langages reconnaissables par un automate déterministe localement réversible et montrer que ce sont exactement les langages dont le monoïde syntaxique appartient à **ER**. Avant cela, nous avons besoins de montrer le lemme suivant.

Lemme 4.25 *Soit $\mathcal{A}$ un automate déterministe localement réversible. Tout monoïde unitaire de la décomposition unitaire-préfixielle de $\mathcal{A}$ est réversible.*

Preuve. Par construction de la décomposition unitaire préfixielle d'un automate, chaque monoïde unitaire M y apparaissant est reconnaissable par un automate dont le graphe est un sous-graphe d'une composante fortement connexe de $\mathcal{A}$. Comme $\mathcal{A}$ est localement réversible, M est réversible. □

On est maintenant en mesure de prouver le résultat principal.

Théorème 4.26 *Un langage L de A^* est reconnaissable par un automate déterministe localement réversible si et seulement si son monoïde syntaxique appartient à* **ER**.

Preuve. Soit L un langage dont le monoïde syntaxique appartient à **ER**. D'après la proposition 4.24, l'automate minimal de L est déterministe localement réversible.

Soit maintenant un langage L reconnaissable par un automate déterministe localement réversible $\mathcal{A}$. D'après le lemme 4.25, pour tout monôme unitaire-préfixiel $(M_0, a_1, M_1, \cdots, a_k, M_k)$ de la décomposition de $\mathcal{A}$, chaque M_j est réversible. Donc, par la proposition 4.20, $M_0 a_1 M_1 \ldots a_k M_k$ est un langage dont le monoïde syntaxique appartient à **ER**. Comme **ER** est une variété, il en résulte que L est un langage dont le monoïde syntaxique appartient à **ER**. □

On peut déduire de ce théorème un moyen efficace de tester si un langage est reconnaissable par un automate déterministe localement réversible.

Corollaire 4.27 *Lorsque l'on minimise un automate déterministe localement réversible, on obtient un automate déterministe localement réversible.*

Preuve. Si $\mathcal{A}$ est un automate déterministe localement réversible alors $M(L(\mathcal{A}))$ appartient à **ER** d'après le théorème 4.26. Donc, par la proposition 4.24, l'automate minimal de L, qui est par définition le minimisé de $\mathcal{A}$, est localement réversible. □

Pour tester si un langage à son monoïde syntaxique dans **ER**, il suffit de calculer son automate minimal et de vérifier s'il est localement réversible. On utilise pour cela l'algorithme `TesteDetLocRev` suivant qui s'applique à un automate déterministe $\mathcal{A}$.

Algorithme 4.28

1. Minimiser $\mathcal{A}$.
2. Tester si l'automate obtenu est localement révesible.

◇

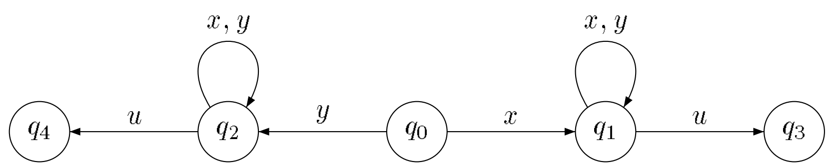
4.4.4 Amélioration du test du niveau 1

Du corollaire 4.27 et du théorème 4.16 on déduit le résultat suivant.

Proposition 4.29 *Soient L un langage de A^* et $\mathcal{A}$ son automate minimal. Les assertions suivantes sont équivalentes :*

(i) L est dans le niveau 1 de la hiérarchie des langages à groupe.

(ii) $\mathcal{A}$ est localement réversible et ne contient aucune configuration du type :



où $x, y, u \in A^*$, q_3 est un état non final et q_4 un état final.

Preuve. Si L est de niveau 1, alors L est reconnaissable par un automate déterministe localement réversible donc son automate minimal est localement réversible.

Réciproquement, si l'automate minimal de L est localement réversible, il ne contient aucune configuration du second type du théorème 4.16. $\square$

Le résultat ci-dessus permet une légère amélioration du test : il est plus facile et rapide de tester si un automate est localement réversible que de tester s'il ne comporte pas de configuration du second type du théorème 4.16. Cependant l'étape la plus lourde du test étant le test de la première configuration, on n'obtient pas là d'amélioration de la complexité théorique dans le cas le pire.

4.5 Niveau 3/2 et automates localement réversibles

Peu de choses sont connues sur le niveau 3/2 de la hiérarchie des langages à groupe. Nous mettons ici en évidence un lien entre les automates localement réversibles et ce niveau.

Proposition 4.30 *Soit L un langage rationnel. Les assertions suivantes sont équivalentes :*

(i) L est reconnaissable par un automate localement réversible.

(ii) L appartient à la clôture polynomiale des langages reconnaissables par un automate déterministe localement réversible.

(iii) L appartient à la clôture polynomiale des langages réversibles.

Preuve.

$(i) \Longrightarrow (iii)$: La preuve est identique à celle du lemme 3.20. Il suffit de décomposer un langage L donné par un automate localement réversible le reconnaissant, en une union de produits alternés de lettres et de langages reconnus par les composantes fortement connexes de cet automate.

$(iii) \Longrightarrow (ii)$: Tout langage réversible est une union de langages reconnaissables par un automate déterministe réversible, donc, a fortiori par un automate déterministe localement réversible. Par conséquent, si un langage appartient à la clôture polynomiale des langages réversibles, il appartient aussi à la clôture polynomiale des langages reconnaissables par un automate déterministe localement réversible.

$(ii) \Longrightarrow (i)$: Si $L_0, \cdots, L_k$ sont des langages reconnaissables par des automates localement réversibles $\mathcal{A}_j = (Q_j, A, E_j, I_j, F_j)$ et si $a_1, \cdots, a_k$ sont des lettres, alors

$$L_0 a_1 L_1 \cdots a_k L_k$$

est un langage reconnaissable par un automate localement réversible. En effet, l'automate $\mathcal{A} = (Q, A, E, I, F)$ où

- $Q = \bigcup Q_j$,
- $E = \bigcup E_j \bigcup \{(p, a_j, q) \mid p \in F_{j-1}, q \in I_j\}$,
- $I = I_0$,
- $F = F_k$

reconnaît, par construction, $L_0 a_1 L_1 \cdots a_k L_k$. Il est de plus, par construction aussi, localement réversible. On en déduit que si L appartient à la clôture polynomiale des langages reconnaissables par un automate déterministe localement réversible, alors L est aussi reconnaissable par un automate localement réversible.

□

Théorème 4.31 *Un langage est de niveau* $3/2$ *dans la hiérarchie des langages à groupe si et seulement s'il est reconnu par un automate localement réversible.*

Preuve.

Afin de simplifier la preuve notons $\mathcal{DLR}$ l'ensemble des langages de A^* reconnaissables par un automate déterministe localement réversible, $\mathcal{R}$ l'ensemble des langages réversibles de A^* et $\mathcal{G}_n$ l'ensemble des langages de A^* qui sont de niveau n dans la hiérarchie des langages à groupes.

On a montré à la proposition 4.22 que

$$\mathcal{G}_1 \subseteq \mathcal{DLR}$$

Donc, par définition,

$$\mathcal{G}_{3/2} \subseteq \mathrm{Pol}\mathcal{DLR}$$

Par ailleurs, les langages réversibles sont fermés pour la topologie profinie et donc, d'après le théorème 4.6,

$$\mathcal{R} \subseteq \mathcal{G}_1$$

Par conséquent,

$$\mathrm{Pol}\mathcal{R} \subseteq \mathcal{G}_{3/2}$$

On a donc

$$\mathrm{Pol}\mathcal{R} \subseteq \mathcal{G}_{3/2} \subseteq \mathrm{Pol}\mathcal{DLR}$$

Or, d'après la proposition 4.30,

$$\mathrm{Pol}\mathcal{R} = \mathrm{Pol}\mathcal{DLR}$$

Ce qui permet de conclure que

$$\mathrm{Pol}\mathcal{R} = \mathcal{G}_{3/2} = \mathrm{Pol}\mathcal{DLR}$$

et d'en déduire le théorème. □

Ce résultat peut encore s'énoncer de la façon suivante.

Théorème 4.32 *Les assertions suivantes sont équivalentes.*

(i) L est de niveau 3/2 de la hiérarchie des langages à groupe.

(ii) L est reconnaissable par un automate localement réversible.

(iii) L est une union finie de langages du type $R_0a_1R_1 \cdots a_kR_k$, où les R_i sont des langages réversibles et les a_i des lettres.

(iv) L est dans la clôture polynomiale des langages réversibles.

(v) L est une union finie de langages du type $R_0R_1 \cdots R_k$, où les R_i sont des langages réversibles.

Preuve. Pour montrer que (iii) est équivalent à (v) il suffit de remarquer que, d'une part, pour tout lettre a, $\{a\}$ est un langage réversible et que, d'autre part, si R_1 et R_2 sont des langages réversibles alors

$$R_1R_2 = \bigcup_{a \in A} (R_1a^{-1})aR_2$$

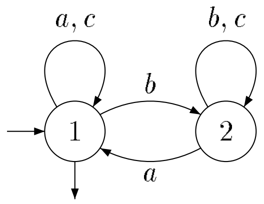
et R_1a^{-1} est un langage réversible. □

On ne connaît pas de borne sur la taille d'un automate localement réversible reconnaissant un langage de niveau 3/2. La question paraît d'autant plus difficile qu'elle permettrait de résoudre la décidabilité du niveau 3/2.

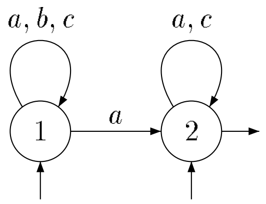
De plus, l'automate minimal d'un langage reconnaissable par un automate localement réversible n'est pas nécessairement localement réversible. Par exemple, l'automate minimal de

$$L = \{a, b, c\}^* a \{a, c\}^* \cup \{a, c\}^*$$

n'est pas localement réversible. C'est en effet l'automate suivant :



Or L est bien de niveau $3/2$ car reconnaissable par l'automate :



4.6 Conclusion

Lorsque l'on étudie les automates finis, on s'intéresse naturellement aux propriétés de leurs graphes. On vient de voir que la notion de réversibilité, qui est intuitive, joue un rôle important dans la description des langages de la hiérarchie des langages à groupe. Cependant, d'un point de vue algorithmique, de nombreux problèmes restent à résoudre : étant donné un langage de niveau $1/2$, on ne sait pas calculer efficacement un automate localement à groupe le reconnaissant, ni calculer un automate localement réversible reconnaissant un langage du niveau $3/2$. On est dans le même cas que dans l'étude des langages réversibles : on possède de bonnes descriptions de ces langages par des automates particuliers, mais on ne sait pas comment calculer efficacement de telles représentations.

Cependant, pour un langage de niveau 1, on sait calculer un automate déterministe localement réversible le reconnaissant : son automate minimal.

Le théorème 4.32 nous montre que si l'on souhaite introduire comme nouvelle hiérarchie de langages la hiérarchie des langages réversibles, dont le niveau 0 serait constitué des langages réversibles, son étude se ramènerait à l'étude de la hiérarchie des langages à groupe (avec un niveau de décalage).

Pour chaque niveau de la hiérarchie des langages à groupe, des problèmes algorithmiques restent ouverts. On ne sait pas, par exemple, quelle est la complexité exacte pour tester si un langage donné par un automate non déterministe est à groupe. Intuitivement, il paraîtrait par ailleurs raisonnable que la minimisation d'un automate à groupe déterministe puisse se faire de façon plus efficace que par l'algorithme d'Hopcroft [Hop71], en utilisant les fortes propriétés des langages à groupe. De plus, contrairement aux autres hiérarchies, la décidabilité du niveau 3/2 est toujours une question ouverte.

Chapitre 5

Hiérarchie de Straubing-Thérien

Sommaire

5.1 Introduction et motivations **167**
5.2 Niveau 1/2 . . . . **168**
5.2.1 Propriétés connues 168
5.2.2 Automates saturés 170
5.2.3 Propriétés topologiques 179
5.3 Niveau 1 . . . . **179**
5.3.1 Caractérisation des langages de niveau 1 179
5.3.2 Automates déterministes localement triviaux 180
5.4 Niveau 3/2 . . . . **181**
5.4.1 Résultats connus 182
5.4.2 Automates localement triviaux 183
5.4.3 Calcul d'une expression rationnelle polynomiale . . . 183
5.4.4 Propriétés topologiques 184
5.5 Langages sans-étoile et topologie profinie **188**
5.6 Conclusion **190**

5.1 Introduction et motivations

La hiérarchie de Straubing-Thérien fut introduite implicitement par D. Thérien [Thé81] et explicitement par H. Straubing [Str85]. Elle permet un classement des langages sans-étoile. Le niveau 0 de cette hiérarchie est composé du langage vide et des langages du type A^*, où A est un alphabet fini.

Cette hiérarchie établit une double classification des langages sans-étoile : une classification algébrique inhérente à toute hiérarchie de concaténation [PW97] et, comme l'a montré W. Thomas [Tho82], une classification logique.

La hiérarchie de Straubing-Thérien correspond de façon bijective à une hiérarchie de la logique du premier ordre. D'autres aspects logiques de la hiérarchie de Straubing-Thérien sont développés dans [Pin96] et [PP86].

Le niveau 1/2 de la hiérarchie de Straubing-Thérien est très lié aux propriétés du produit de mélange sur A^* (voir définition 1.18), ce qui lui donne une place intéressante en théorie combinatoire des mots [Lot83].

Le niveau 1 de la hiérarchie de Straubing-Thérien est algébriquement riche, comme l'a montré I. Simon [Sim75], dont les travaux ont permis à J. Stern [Ste85] d'élaborer un algorithme polynomial de décision d'appartenance à ce niveau (pour un langage donné par un automate déterministe). Les propriétés algébriques de ce niveau ont aussi été étudiées par J. Almeida [Alm90] ainsi que par H. Straubing et D. Thérien [ST85].

Le niveau 3/2 de la hiérarchie de Straubing-Thérien représente le niveau maximal que l'on sache décider actuellement (le problème est ouvert pour les niveaux supérieurs). Les langages de ce niveau peuvent s'exprimer par des expressions rationnelles particulières [Arf87, Arf91], ce qui les rend utiles en vérification de systèmes. A ce sujet, on pourra se reporter aux travaux d'Abdulla et al. [ABJ98, AAB99], ainsi qu'aux résultats récents de A. Bouajjani, A. Muscholl et T. Touili [BMT01] qui ont montré que ce niveau est stable par semi-commutation.

On ne dispose que de résultats de décidabilité partiels pour les langages de niveau 2 de la hiérarchie Straubing-Thérien : D. Cowan [Cow93] a montré que l'on pouvait décider si un langage dont le monoïde syntaxique est inversif appartient au niveau 2 de la hiérarchie de Straubing-Thérien. Par ailleurs, H. Straubing [Str88] a montré que sur un alphabet à deux lettres, on pouvait aussi décider si un langage appartient au niveau 2.

Dans ce chapitre, nous commencerons par faire une étude des langages du niveau 1/2 en étudiant plus particulièrement leurs différentes représentations. Ensuite, nous aborderons aussi les problèmes de représentation des langages de niveau 3/2, ainsi que leurs propriétés topologiques. Nous finirons par de courtes considérations sur les propriétés topologiques des langages sans-étoile.

5.2 Niveau 1/2

Nous étudions ici les propriétés des langages de niveau 1/2 de la hiérarchie de Straubing-Thérien.

5.2.1 Propriétés connues

Les langages de A^* de niveau 1/2 de la hiérarchie de Straubing-Thérien sont, par définition, les langages qui sont une union finie de langages du type

$$A^*a_1A^*a_2\cdots A^*a_kA^*$$

où les a_i sont des lettres.

Afin de montrer les liens entre les langages de niveau 1/2 de la hiérarchie de Straubing-Thérien et la combinatoire des mots, on a besoin d'introduire la notion d'*idéal de mélange*.

idéal de mélange

Définition 5.1 *Un langage L de A^* est un* idéal de mélange *si pour tout mot $u \in L$ et tout mot $v \in A^*$, $u \sqcup\!\sqcup\, v \in L$.*

En utilisant un résultat important de G. Higman [Hig52, Lot83] de la théorie combinatoire des mots, on obtient le résultat suivant.

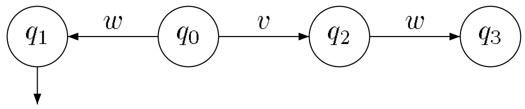
Proposition 5.2 *Un langage est de niveau* 1/2 *dans la hiérarchie de Straubing-Thérien si et seulement si c'est un idéal de mélange.*

Ce qui peut encore s'écrire [PW97].

Proposition 5.3 *Un langage L est de niveau* 1/2 *dans la hiérarchie de Straubing-Thérien si et seulement si pour tous mots u, v et w, si $uw \in L$, alors $uvw \in L$.*

A nouveau, cela peut se vérifier sur des configuration de l'automate minimal.

Proposition 5.4 *Soit L un langage de A^* et $\mathcal{A}$ le complété de son automate minimal. Le langage L est de niveau* 1/2 *dans la hiérarchie de Straubing-Thérien si et seulement si il n'y a pas dans $\mathcal{A}$ de configuration du type :*



avec q_3 non final.

On vérifie qu'un automate déterministe $\mathcal{A}$ ne possède aucune configuration du type ci-dessus en calculant la clôture transitive (V, E) du graphe non étiqueté sous-jacent au carré de $\mathcal{A}$ et en vérifiant qu'il n'y a pas dans (V, E) de paire de transitions du type :

$$\{((q_0, q_2), (q_2, q_4)), ((q_0, q_2), (q_1, q_3))\}$$

avec q_1 final et q_3 non final.

Cette méthode exige de calculer la clôture transitive d'un graphe à n^2 sommets, et sa complexité dépasse donc n^4.

5.2.2 Automates saturés

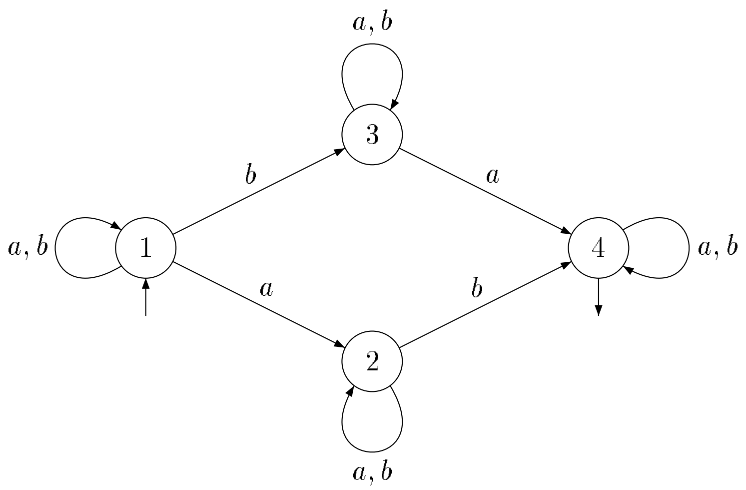
Les résultats précédents vont nous permettre de donner une représentation par des automates finis particuliers des langages de niveau 1/2.

automate saturé

Définition 5.5 *Un automate fini $\mathcal{A} = (Q, A, E, I, F)$ est dit* saturé *si pour tout $p \in Q$ et tout $a \in A$, $p \in p \cdot a$.*

Graphiquement, cela signifie que sur chaque état il y a une boucle étiquetée par chacune des lettres.

Exemple 5.6 Par exemple, l'automate suivant est saturé.



$\triangledown$

Ce qui nous permet de donner la caractérisation suivante des langages de niveau 1/2.

Proposition 5.7 *Un langage est de niveau $1/2$ dans la hiérarchie de Straubing-Thérien si et seulement s'il est reconnu par un automate saturé.*

Montrons d'abord le lemme suivant.

Lemme 5.8 *Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate fini et $\mathcal{A}' = (Q, A, E', I, F)$ où*

$$E' = E \cup \bigcup \{(p, a, p) \mid p \in Q, a \in A\}$$

On a $L(\mathcal{A}') = L(\mathcal{A}) \sqcup\!\sqcup A^*$.

PREUVE. $\mathcal{A} = (Q, A, E, I, F)$ un automate fini et $\mathcal{A}'(Q, A, E', I, F)$ construit comme ci-dessus. Soit $w \in L(\mathcal{A}')$. Il existe un chemin réussi m dans $\mathcal{A}'$ reconnaissant w. On peut décomposer m en une suite alternée de chemins utilisant d'une part les transitions de E et d'autre part les transitions de $E' \setminus E$. On a alors

$$m = r_1 s_1 r_2 s_2 \cdots r_n s_n$$

où les r_i sont des chemins dans $\mathcal{A}$ et où les s_i sont des chemins n'utilisant que des boucles de longueur un. Il en résulte que $r_1 r_2 \cdots r_n$ est un chemin réussi dans $\mathcal{A}$ et donc que

$$w \in [r_1 \cdots r_n]_{\mathcal{A}} \sqcup\!\sqcup A^* \subseteq L(\mathcal{A}) \sqcup\!\sqcup A^*$$

Réciproquement, si $w \in L(\mathcal{A}) \sqcup\!\sqcup A^*$ alors $w = u_0 v_0 \cdots u_n v_n$ avec $u = u_0 u_1 \cdots u_n \in L(\mathcal{A})$. Alors, en considérant un chemin réussi dans $\mathcal{A}$ d'étiquette u, on peut, à chaque fois que l'on a fini de lire un u_i, intercaler des transitions de E' pour lire v_i et ainsi construire un chemin réussi dans $\mathcal{A}'$ d'étiquette w. □

On peut maintenant prouver la proposition 5.7.

PREUVE. Soit L un idéal de mélange et $\mathcal{A}$ un automate le reconnaissant. On considère l'automate $\mathcal{A}'$ construit à partir de $\mathcal{A}$ en le rendant saturé, comme dans le lemme 5.8. L'automate $\mathcal{A}'$ reconnaît $L \sqcup\!\sqcup A^* = L$ car L est un idéal de mélange. Par conséquent, L est reconnaissable par un automate saturé.

Réciproquement si L est reconnaissable par un automate saturé $\mathcal{A}$, alors, on a $\mathcal{A}' = \mathcal{A}$ (où $\mathcal{A}'$ est construit comme ci-dessus). Donc, d'après le lemme 5.8, $L = L \sqcup\!\sqcup A^*$. Par conséquent L est un idéal de mélange. □

On peut remarquer que la preuve ci-dessus est constructive, c'est à dire qu'en partant d'un automate reconnaissant un langage L de niveau 1/2 de la hiérarchie de Straubing-Thérien, on peut construire un automate saturé le reconnaissant.

Corollaire 5.9 *Soit L un langage de niveau 1/2 de la hiérarchie de Straubing-Thérien donné par un automate à n états et m transitions. On peut calculer en temps $O(n+m)$ un automate saturé reconnaissant L.*

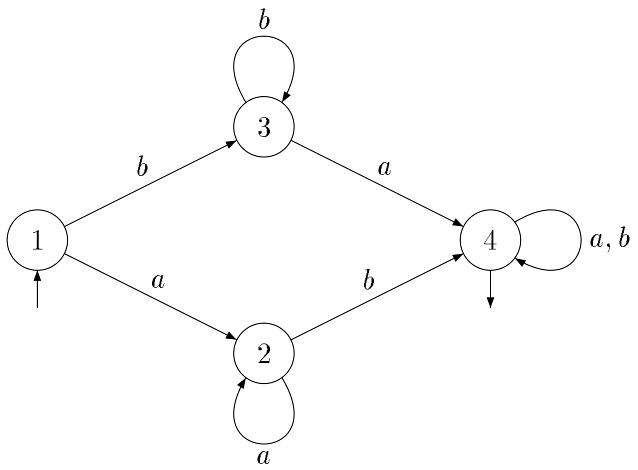
On utilise pour cela l'algorithme `SatureAutomate` suivant, qui s'applique à un automate $\mathcal{A} = (Q, A, E, I, F)$ et dont la justification découle de la proposition 5.7.

Algorithme 5.10

1. Calculer $E' = E \cup \{(p, a, p) \mid p \in Q, a \in A\}$.
2. Retourner (Q, A, E', I, F).

$\diamondsuit$

Exemple 5.11 L'algorithme est tout simple, si on part de l'automate suivant :



Alors, après application de l'algorithme, on obtient l'automate de l'exemple 5.6.

$\triangledown$

On en déduit un algorithme pour tester si un langage est de niveau 1/2 dans la hiérarchie de Straubing-Thérien.

Proposition 5.12 *Soit L un langage rationnel de A^* donné par un automate déterministe à n états. On peut tester en temps $O(n^2)$ si L est du niveau* 1/2 *dans la hiérarchie de Straubing-Thérien.*

Pour cela, on applique l'algorithme `TesteST1/2` suivant, qui s'applique à un automate déterministe $\mathcal{A}$ et qui teste si $L(\mathcal{A})$ est dans le niveau 1/2 de la hiérarchie de Straubing-Thérien.

Algorithme 5.13

1. Calculer $\mathcal{A}' =$ `SatureAutomate`$(\mathcal{A})$.

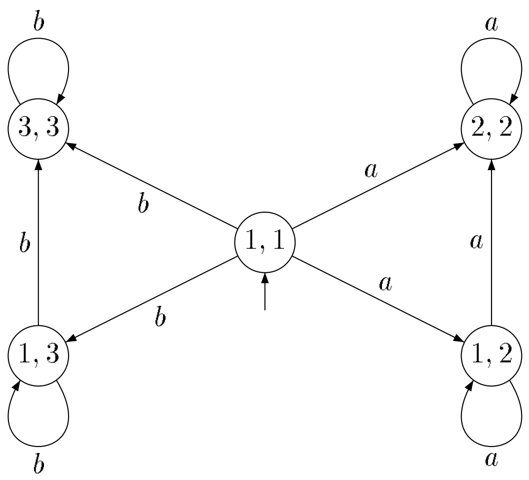
2. Tester si $L(\mathcal{A}') \cap L(\mathcal{A})^c \neq \emptyset$.

$\diamondsuit$

Preuve. Vérifions la complexité de cet algorithme. Comme $\mathcal{A}$ est déterministe, il possède $O(n)$ transitions, donc la première étape se fait en temps linéaire. De plus, comme en effectuant cette étape, on ajoute au plus un nombre linéaire de transitions, l'automate $\mathcal{A}'$ possède $O(n)$ transitions. La seconde étape s'effectue aussi en temps linéaire. En effet, comme $\mathcal{A}$ est déterministe, on peut calculer en temps $O(n)$ un automate à $O(n)$ états reconnaissant $L(\mathcal{A})^c$. Construire un automate reconnaissant l'intersection se fait par un simple produit d'automates, donc en temps $O(n^2)$. Et comme les deux automates ont $O(n)$ transitions, l'automate obtenu possède $O(n^2)$ transitions. Par conséquent, tester si ce produit reconnaît le vide se fait en temps $O(n^2)$, par le calcul d'un arbre couvrant. La complexité totale est donc de $O(n^2)$.

Montrons maintenant que l'algorithme répond à la question souhaitée. Remarquons tout d'abord que, par construction, $L(\mathcal{A}) \subseteq L(\mathcal{A}')$, donc tester si $L(\mathcal{A}') \cap L(\mathcal{A})^c \neq \emptyset$ est équivalent à tester si $L(\mathcal{A}) = L(\mathcal{A}')$. D'après la preuve de la proposition 5.7, si $L(\mathcal{A})$ est dans le niveau 1/2 de la hiérarchie de Straubing-Thérien, alors $L(\mathcal{A}) = L(\mathcal{A}')$ et donc, dans ce cas, l'algorithme fournit la bonne réponse. Réciproquement, si $L(\mathcal{A}) = L(\mathcal{A}')$, alors $L(\mathcal{A})$ est reconnaissable par un automate saturé et appartient donc, d'après la proposition 5.7, au niveau 1/2 de la hiérarchie de Straubing-Thérien. $\square$

Exemple 5.14 Considérons le langage donné par l'automate $\mathcal{A}$ de l'exemple 5.11. Après la première étape on obtient l'automate $\mathcal{A}'$ de l'exemple 5.6. Il faut maintenant tester si $L(\mathcal{A}') \cap L(\mathcal{A})^c \neq \emptyset$. On construit donc le produit des automates et on obtient :



Cet automate reconnaît le langage vide (il n'y a pas d'état final), donc le langage de départ est bien de niveau 1/2 dans la hiérarchie de Straubing-Thérien.

$\triangledown$

Il est à noter que l'algorithme décrit ci-dessus est très simple et plus efficace que celui décrit dans [PW97], qui nécessite le calcul de la fermeture transitive d'un graphe à n^2 sommets, et dont la complexité dépasse donc n^4.

Intéressons nous maintenant à des automates plus particuliers que les automates saturés qui reconnaissent aussi exactement les idéaux de mélange.

localement trivial

Définition 5.15 *Un automate est dit* localement trivial *si ses composantes fortement connexes sont chacune réduite à un élément.*

Ceci signifie que tous les cycles simples du graphe de l'automate sont de longueur un. Finalement, on peut résumer les résultats par la proposition ci-dessous.

Proposition 5.16 *On a équivalence entre les assertions suivantes :*

(i) L est de niveau 1/2 dans la hiérarchie de Straubing-Thérien.

(ii) L est un idéal de mélange.

(iii) L est reconnaissable par un automate saturé.

(iv) L est reconnaissable par un automate saturé et localement trivial.

Preuve. Les équivalences entre (i), (ii), et (iii) ont été montrées ou énoncées précédemment.

Par ailleurs, tout langage de niveau 1/2 est une union finie de langages du type

$$A^*a_1A^*a_2\cdots A^*a_kA^*$$

où les a_i sont des lettres ; il est donc reconnu par l'union des automates

$$(\{0,\ldots,k\}, A, E, 0, k)$$

où

$$E = \{(i, a_{i+1}, i+1) \mid 0 \le i \le k-1\} \cup \{(i, a, i) \mid a \in A\}$$

qui sont des automates saturés. Cela prouve que (i) implique (iv). De plus, il est clair que (iv) implique (iii). $\square$

Intéressons nous maintenant aux calculs d'expressions rationnelles particulières. On rappelle que si $\mathcal{E}$ est une expression rationnelle, alors $\mathcal{E}$ désigne le langage qu'elle représente.

expression polynomiale

Définitions 5.17 *On appelle expression rationnelle* polynomiale *toute expression rationnelle qui est une union finie d'expressions du type*

$$\mathcal{A}_0^*\mathcal{B}_1\mathcal{A}_1^*\cdots\mathcal{A}_{k-1}^*\mathcal{B}_k\mathcal{A}_k^*$$

où les expressions $\mathcal{A}_i$ *sont du type* $\mathcal{A}_i = \cup_{a\in A_i} a$*, avec* $A_i \subseteq A$ *et les expressions* $\mathcal{B}_i$ *sont du type* $\mathcal{B}_i = \cup_{a\in B_i} a$*, avec* $B_i \subseteq A$ *et* $B_i \neq \emptyset$*. On pose* $\mathcal{A} = \cup_{a\in A} a$*. Une expression rationnelle polynomiale est dite* saturée *si pour tout* i*, on a* $A_i = A$*.* saturée

Par définition, les expressions polynomiales saturées caractérisent les langages de niveau 1/2, car elles représentent exactement, à factorisations près, la décomposition d'un langage de la clôture polynomiale du niveau 0.

Proposition 5.18 *Soit* L *un langage de* A^* *de niveau* $1/2$ *dans la hiérarchie de Straubing-Thérien donné par un automate à* n *états. On peut calculer en temps* $O((|A|+1)^n)$ *une expression rationnelle polynomiale saturée reconnaissant* L*.*

Pour cela, on utilise l'algorithme `CalExpST1/2` suivant, qui s'applique à un automate $\mathcal{A}$ déterministe à n état reconnaissant un idéal de mélange.

Algorithme 5.19

1. Calculer l'ensemble K des mots de $L(\mathcal{A})$ de longueur inférieure ou égale à n.
2. Retourner

$$\mathcal{E} = \bigcup_{a_0a_1\cdots a_k\in K} \mathcal{A}^*a_0\mathcal{A}^*a_1\mathcal{A}^*\cdots\mathcal{A}^*a_k\mathcal{A}^*$$

$\diamondsuit$

Preuve. Commençons par analyser la complexité de l'algorithme. Pour la première étape, il suffit de tester, pour tous les mots de longueur au plus n, s'ils appartiennent à $L(\mathcal{A})$. Chaque test s'effectue en temps $O(n)$ et il y a $O(|A|^{n+1})$ mots de longueur plus petite que n. Cette étape s'effectue donc en temps $O(n|A|^{n+1}) = O((|A|+1)^n)$. La seconde étape s'effectue en $|K|$ étapes nécessitant chacune un temps $O(n)$, donc aussi en temps $O((|A|+1)^n)$.

Justifions maintenant l'algorithme. Comme $L(\mathcal{A})$ est un idéal de mélange, pour tout $u \in K$, $u \sqcup\!\sqcup K \subseteq L$ et, par conséquent, $\mathcal{E}\nu \subseteq L$. Réciproquement, si $v \in L$, alors, il existe un chemin réussi m dans $\mathcal{A}$ d'étiquette v. Or, on peut écrire m sous la forme

$$m = m_0m_1m_2\cdots m_{2k}$$

où les m_{2j+1} sont des boucles et où $m_0m_2m_4\cdots m_{2k}$ est un chemin sans boucle d'étiquette u. Le mot u est alors un sous-mot de v de longueur inférieure à n.

Il en résulte qu'en posant $u = a_0 \cdots a_k$, on a

$$v \in (\mathcal{A}^* a_0 \mathcal{A}^* a_1 \mathcal{A}^* \cdots \mathcal{A}^* a_k \mathcal{A}^*)\nu$$

Par conséquent, $L \subseteq \mathcal{E}\nu$. $\square$

Montrons que tout algorithme calculant une telle expression demande au moins un espace exponentiel dans le pire des cas. Pour cela on va exhiber une suite de langages K_n de niveau 1/2 dans la hiérarchie de Straubing-Thérien dont la taille de l'automate minimal est en $O(n)$ et dont la taille de toute expression rationnelle polynomiale saturée reconnaissant K_n est en $O(r^n)$ avec $r \geq 1$. Plus formellement, introduisons l'invariant

$$R_{S_{1/2}}(n) = \max_{\substack{L \text{ idéal de mélange} \\ m(L) \leq n}} \left[\min_{\mathcal{E}\nu = L} \{\mathcal{E}\tau \mid \mathcal{E} \text{ polynomiale saturée}\} \right]$$

Proposition 5.20 *On a* $R_{S_{1/2}}(n) = \Omega(2^{n/3})$ *et* $R_{S_{1/2}}(n) = O(|A|^{n+1})$.

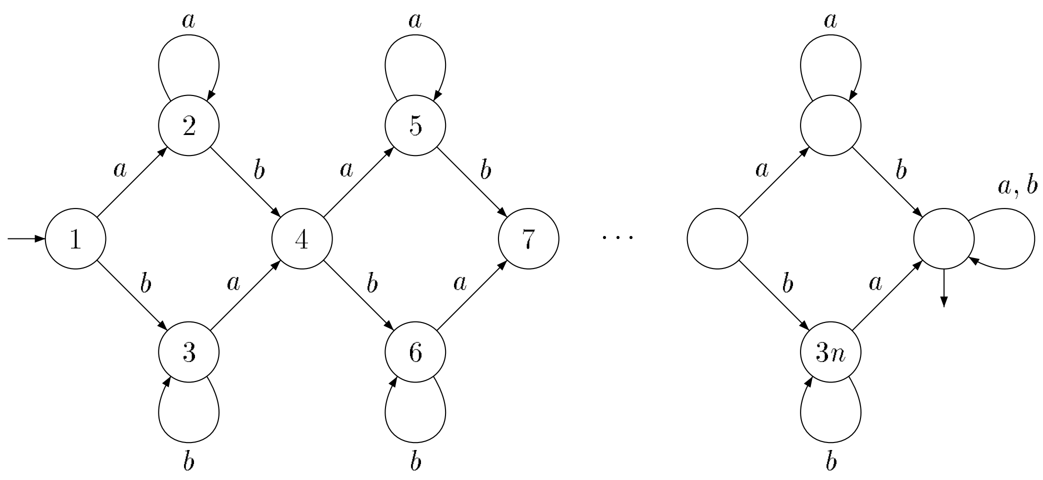
Cette explosion de la taille d'un expression rationnelle reconnaissant un langage donné provient du nombre des unions qui peut être exponentiel, alors que la taille des sous-expressions est linéaire (voir corollaire 3.21).

Preuve. On a montré à la proposition 5.18 que, pour tout idéal de mélange L tel que $m(L) = n$, il existe une expression rationnelle polynomiale saturée reconnaissant L de taille au plus $O(|A|^{n+1})$. Ce qui permet de conclure que

$$R_{S_{1/2}}(n) = O(|A|^{n+1})$$

Pour la borne inférieure on travaille sur l'alphabet $A = \{a, b\}$. Considérons le langage L défini par l'automate de l'exemple 5.6. Comme on l'a vu lors de l'exemple 5.14, L est un langage de niveau 1/2. Or, comme ce niveau est stable par produit, pour tout entier n, L^n est aussi un langage de niveau 1/2.

Or l'automate minimal de L^n est le suivant :



Donc, $m(L^n) = 3n + 1$. Soit $\mathcal{E}$ une expression polynomiale saturée reconnaissant L. Supposons que

$$\mathcal{A}^*\mathcal{A}_1\mathcal{A}^* \cdots \mathcal{A}_k\mathcal{A}^*$$

soit une expression apparaissant dans $\mathcal{E}$. Alors $u = a_0a_1 \cdots a_k$, avec $a_i \in \mathcal{A}_i\nu$ appartient à L^n.

Comme aucun mot de L n'a une longueur inférieure à $2n$, il en résulte que chaque expression du type $\mathcal{A}^*\mathcal{A}_0\mathcal{A}^* \cdots \mathcal{A}_k\mathcal{A}^*$ apparaissant dans $\mathcal{E}$ vérifie $k \geq 2n$. On pose $R = \{ab, ba\}$. Tout mot de $u \in R^n$ est dans L^n et a pour longueur $2n$. Par conséquent, il existe une expression du type

$$\mathcal{E}_0 = \mathcal{A}^*\mathcal{A}_1\mathcal{A}^* \cdots \mathcal{A}_k\mathcal{A}^*$$

apparaissant dans $\mathcal{E}$ telle que $u \in \mathcal{E}_0\nu$. Comme $k \geq 2n$, cela impose que $k = 2n$ et donc que $u \in (\mathcal{A}_1 \cdots \mathcal{A}_k)\nu$.

Montrons que si $(\mathcal{A}_1 \cdots \mathcal{A}_k)\nu \subseteq L^n$ (avec $\mathcal{A}_i\nu \neq \emptyset$), alors pour tout $1 \leq i \leq k$, $|\mathcal{A}_i\nu| = 1$. En effet, supposons qu'il existe i tel que $\mathcal{A}_i\nu = \{a, b\}$. Dans ce cas, il existe x et y dans A^* tels que $|x| = i - 1, |y| + |x| = 2n - 1$ et $xay, xby \in R^n$. Donc, quand on lit xay et xby dans l'automate minimal de L, on n'emprunte aucune boucle. Cela impose que $|x|$ est impair (sinon, après avoir lu xa il faudrait lire un b, et après avoir lu xb un a : la première lettre de y serait à la fois a et b). En partant de l'état final, le même argument permet de dire que $|y|$ est impair. Ce qui est contradictoire avec le fait que $|x| + |y| = 2n - 1$.

Il en résulte que toutes les expressions du type

$$\mathcal{A}^*a_1\mathcal{A}^* \cdots a_{2n}\mathcal{A}^*$$

avec $a_1 \cdots a_{2n}\nu \in R^n$ apparaissent dans $\mathcal{E}$. Or $(\mathcal{A}^* a_1 \mathcal{A}^* \cdots a_{2n}\mathcal{A}^*)\tau \geq 6n+2$ et il y a $|R^n| = 2^n$ expressions de ce type dans $\mathcal{E}$. Donc $\mathcal{E}\tau \geq 3n2^{n+1}$.

On en déduit, en remarquant que $R_{S_{1/2}}$ est une fonction croissante de n, que

$$R_{S_{1/2}}(n) = \Omega(2^{n/3})$$

□

Intéressons nous maintenant au passage, pour un langage de niveau 1/2, d'une expression rationnelle générale à une expression polynomiale saturée. Introduisons alors

$$T_{S_{1/2}}(n) = \max_{\substack{L \text{ idéal de mélange} \\ e(L) \leq n}} \left[\min_{\mathcal{E}\nu = L} \{\mathcal{E}\tau \mid \mathcal{E} \text{ polynomiale saturée}\} \right]$$

Là encore, $T_{S_{1/2}}(n)$ est au moins exponentiel.

Proposition 5.21 *On a $T_{S_{1/2}}(n) = \Omega(2^{n/6})$ et $T_{S_{1/2}}(n) = O(|A|^{n+1})$.*

Preuve. On montre que $T_{S_{1/2}}(n) = O(|A|^{n+1})$ par le même argument que pour $R_{S_{1/2}}(n)$.

Pour la borne inférieure, considérons les langages L^n définis dans la preuve de la proposition 5.20. Comme

$$L^n = ((a^+b \cup b^+a)^n(a \cup b)^*)\nu$$

il en résulte que si $n \leq 2$, alors

$$e(L^n) \leq 5n + 2 \leq 6n$$

Par ailleurs, comme on l'a vu

$$\min_{\mathcal{E}\nu = L^n} \{\mathcal{E}\tau \mid \mathcal{E} \text{ polynomiale saturée}\} \geq 3n2^{n+1}$$

Il s'en suit que si n est un multiple de 6,

$$e(L^{n/6}) \geq \frac{1}{2}n2^{1+n/6}$$

Par conséquent, $T_{S_{1/2}}(n) = \Omega(2^{n/6})$. □

5.2.3 Propriétés topologiques

Comme A^* et $\emptyset$ sont des langages à groupe, chaque niveau de la hiérarchie de Straubing-Thérien est inclus dans le niveau correspondant de la hiérarchie des langages à groupe. Les langages du niveau 1/2 de la hiérarchie de Straubing-Thérien sont donc en particulier tous ouverts. Plus précisément, on a la proposition suivante.

Proposition 5.22 *Tout langage de niveau* $1/2$ *de la hiérarchie de Straubing-Thérien est un ouvert dense pour la topologie profinie.*

Preuve. C'est une conséquence directe de la remarque ci-dessus, et de la proposition 2.64. □

On peut en déduire le résultat suivant, qui est montré dans [PW97] de façon algébrique (mais simple aussi).

Proposition 5.23 *Tout langage de* A^* *de niveau* $1/2$ *de la hiérarchie de Straubing-Thérien dont le complémentaire est aussi de niveau* $1/2$*, est soit vide, soit* A^**.*

Preuve. Soit L un langage de niveau 1/2 dont le complémentaire est aussi du niveau 1/2. Dans ce cas, il est à la fois ouvert et fermé, et, s'il n'est pas vide, il est dense. Et comme le seul langage à la fois fermé et dense est A^*, il en résulte que $L = A^*$. □

5.3 Niveau 1

Nous nous intéressons ici au niveau 1 de la hiérarchie de Straubing-Thérien qui contient les langages finis et co-finis.

5.3.1 Caractérisation des langages de niveau 1

Dans un automate localement trivial $\mathcal{A}$, on peut ordonner les états de la façon suivante : $p \geq_{\mathcal{A}} q$ si et seulement s'il existe un chemin dans $\mathcal{A}$ allant de p à q.

Dans [Sim75], I. Simon donne une caractérisation des langages de niveau 1.

Théorème 5.24 *Soit* L *un langage rationnel de* A^* *et* $\mathcal{A} = (Q, A, E, i, F)$ *son automate minimal. Le langage* L *est de niveau* 1 *dans la hiérarchie de Straubing-Thérien si et seulement* $\mathcal{A}$ *vérifie les deux propriétés suivantes :*

- $\mathcal{A}$ *est localement trivial.*

- *Pour tout alphabet $B \subseteq A$, chaque composante connexe de l'automate*

$$\mathcal{B} = (Q, B, E \cap \{(p, a, q) \mid a \in B, p, q \in Q\}, i, F)$$

possède un unique élément minimal pour l'ordre $\geq_{\mathcal{B}}$.

Dans [Ste85], J. Stern a montré comment tester cette condition en temps polynomial.

Les langages du niveau 1 forment un variété de langages dont la variété de monoïdes finis associée est notée **J**.

5.3.2 Automates déterministes localement triviaux

Nous allons ici donner un résultat similaire à celui prouvé dans le chapitre 4, où l'on a montré que l'ensemble des langages du niveau 1 de la hiérarchie des langages à groupe est inclus dans l'ensemble des langages reconnaissables par un automate déterministe localement réversible. Ce résultat est connu (voir théorème 5.24), mais on en donne une preuve similaire à celle faite dans le cas de la hiérarchie des langages à groupe afin de souligner le parallèle entre les constructions.

Théorème 5.25 *Tout langage de niveau 1 de la hiérarchie de Straubing-Thérien est reconnaissable par un automate déterministe localement trivial.*

Avant de montrer ce théorème, on a besoin de deux résultats intermédiaires.

Proposition 5.26 *La classe des langages reconnaissables par un automate déterministe localement trivial est stable par les opérations booléennes.*

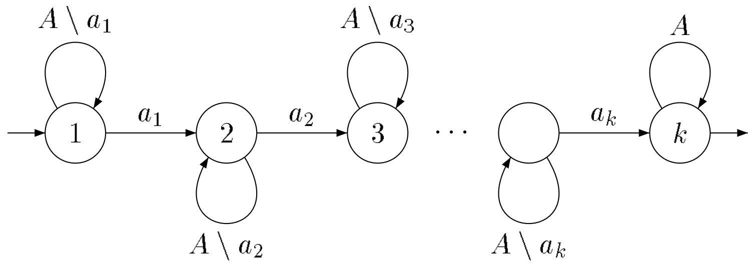
Preuve. La preuve est identique à celle de la proposition 4.31 : compléter un automate localement trivial le laisse localement trivial (on ajoute au plus une composante fortement connexe qui est triviale), et on vérifie aisément que le produit de deux automates localement triviaux est localement trivial. □

Proposition 5.27 *Tout langage de niveau 1/2 de la hiérarchie de Straubing-Thérien est reconnaissable par un automate déterministe localement trivial.*

Preuve. D'après la proposition 5.26, il suffit de prouver le résultat pour les langages du type

$$A^*a_1A^* \cdots A^*a_kA^*$$

Or ce langage est reconnu par l'automate déterministe localement réversible suivant :



C'est en effet l'automate que l'on obtient en déterminisant l'automate localement trivial saturé associée au langage $A^*a_1A^* \cdots A^*a_kA^*$. □

On est maintenant en mesure de prouver le théorème 5.25.

PREUVE. Tout langage de niveau 1/2 est reconnaissable par un automate déterministe localement trivial et comme la classe des langages reconnaissables par un automate déterministe localement trivial est close par les opérations booléennes, tout langage de niveau 1 est accepté par un automate déterministe localement trivial. □

Notons ici que les automates déterministes localement triviaux sont appelés *extensifs* dans [Pin84].

automate extensif

On dispose d'une caractérisation syntaxique précise des langages reconnaissables par un automate déterministe localement trivial [Eil76]

Théorème 5.28 *Un langage est reconnaissable par un automate déterministe localement trivial si et seulement si son monoïde syntaxique est $\mathcal{R}$-trivial.*

Dans la littérature, la variété de monoïdes finis de tous les monoïdes $\mathcal{R}$-triviaux (définition 4.17) est notée **R**. La variété de langages associée est la classe des langages reconnaissables par un automate déterministe localament trivial.

5.4 Niveau 3/2

Nous nous intéressons ici aux propriétés des langages de niveau 3/2 de la hiérarchie de Straubing-Thérien.

5.4.1 Résultats connus

Le résultat suivant est prouvé dans [Arf87, Arf91].

Proposition 5.29 *Les assertions suivantes sont équivalentes :*

(i) *L est de niveau 3/2 dans la hiérarchie de Straubing-Thérien.*

(ii) *L est une union finie de langages du type*

$$A_0^* a_1 A_1^* \cdots a_k A_k^*$$

où les $a_i \in A$ et $A_i \subseteq A$.

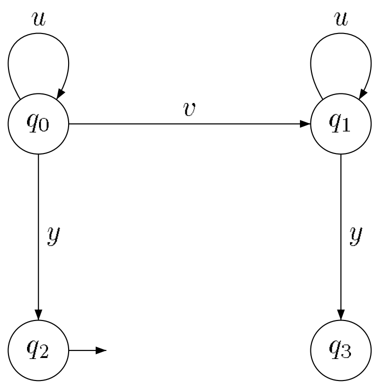
Dans [Arf87, Arf91], il est aussi montré que le niveau 3/2 est décidable, mais la preuve, qui repose sur des résultats de K. Hashiguchi [Has88], ne permet pas de construire un algorithme utilisable. En revanche, l'approche algébrique [PW97] permet de fournir un algorithme en temps polynomial. On utilise pour cela la caractérisation suivante.

Proposition 5.30 *Un langage L de A^* est de niveau 3/2 dans la hiérarchie de Straubing-Thérien si et seulement si pour tous mots x et y de A^* et pour tous mots u et v de A^* utilisant les mêmes lettres,*

$$xu^*y \subseteq L \quad \textit{implique} \quad xu^*vu^*y$$

Cela se traduit par le résultat suivant [PW97].

Proposition 5.31 *Un langage L de A^* est de niveau 3/2 dans la hiérarchie de Straubing-Thérien si et seulement si le complété de son automate minimal ne comporte aucune configuration du type :*



où u et v utilisent exactement les mêmes lettres et où q_3 n'est pas final.

Récemment A. Bouajjani, A. Muscholl et T. Touli [BMT01] ont montré, en utilisant des arguments de complexité, que ce problème est décidable en temps polynomial à la fois en la taille d'un automate déterministe reconnaissant le langage et en la taille de l'alphabet (alors que dans l'algorithme de [PW97], on considère que la taille de l'alphabet est une constante).

Proposition 5.32 *Soit L un langage rationnel de A^* donné par un automate déterministe à n états. On peut tester en temps polynomial en $|A| + n$, si L appartient au niveau 3/2 de la hiérarchie de Straubing-Thérien.*

5.4.2 Automates localement triviaux

Afin de souligner le parallèle avec ce que l'on a démontré pour la hiérarchie des langages à groupe, le résultat suivant peut se déduire facilement la proposition 5.29

Corollaire 5.33 *Un langage est de niveau 3/2 dans la hiérarchie de Straubing-Thérien si et seulement s'il est reconnu par un automate localement trivial.*

5.4.3 Calcul d'une expression rationnelle polynomiale

De même que pour les langages de niveau 1/2, il est intéressant [ABJ98, AAB99] de savoir calculer, pour un langage de niveau 3/2, une expression rationnelle polynomiale le reconnaissant. Les seuls algorithmes connus actuellement [Arf87, Arf91, PW97] ne sont pas utilisables en pratique (car la complexité est trop importante). Nous montrons ici qu'un tel algorithme est, en espace mémoire, au moins exponentiel. Introduisons pour cela :

$$R_{S_{3/2}}(n) = \max_{\substack{L \text{ de niveau } 3/2 \\ m(L) \leq n}} \left[\min_{\mathcal{E}\nu = L} \{\mathcal{E}\tau \mid \mathcal{E} \text{ polynomiale}\} \right]$$

Proposition 5.34 *On a $R_{S_{1/2}}(n) \leq 2|A|((R_{S_{3/2}}(n))$.*

Preuve. Remarquons d'abord que si

$$\mathcal{E} = \bigcup \mathcal{A}_0^* \mathcal{B}_1 \mathcal{A}_1^* \cdots \mathcal{B}_k \mathcal{A}_k^*$$

est une expression polynomiale reconnaissant un idéal de mélange L, alors l'expression polynomiale saturée

$$\mathcal{E}' = \bigcup \mathcal{A}^* \mathcal{B}_1 \mathcal{A}^* \cdots \mathcal{B}_k \mathcal{A}^*$$

avec $\mathcal{A} = \cup_{a \in A} a$ reconnaît aussi L.

Or, $\mathcal{E}'\tau \leq 2|A|\mathcal{E}\tau$. En effet, pour passer de $\mathcal{A}_i$ à $\mathcal{A}$, on ajoute au plus $2(|A|-1)$ symboles.

Il en résulte que, pour tout langage L de niveau $1/2$,

$$\min_{\mathcal{E}\nu=L}\{\mathcal{E}\tau \mid \mathcal{E} \text{ polynomiale}\} \geq 2|A| \min_{\mathcal{E}\nu=L}\{\mathcal{E}\tau \mid \mathcal{E}\text{polynomiale saturée}\}$$

Par conséquent, $R_{S_{1/2}}(n) \leq 2|A|((R_{S_{3/2}}(n))$. □

En utilisant la proposition 5.20, on en déduit la borne suivante.

Corollaire 5.35 *Si $|A| \geq 2$, alors $R_{S_{3/2}}(n) = \Omega(2^{n/3})$.*

Intéressons nous maintenant à la borne pour un langage donné par une expression rationnelle. On pose alors

$$T_{S_{3/2}}(n) = \max_{\substack{L \text{ de niveau } 3/2 \\ e(L)\leq n}} \left[\min_{\mathcal{E}\nu=L}\{\mathcal{E}\tau \mid \mathcal{E} \text{ polynomiale}\}\right]$$

Proposition 5.36 *On a $T_{S_{3/2}}(n) = \Omega(2^{n/6})$.*

Preuve. Par la même preuve que pour la proposition 5.34, on montre que

$$T_{S_{3/2}}(n) = \Omega(T_{S_{1/2}}(n)) = \Omega(2^{n/6})$$

□

5.4.4 Propriétés topologiques

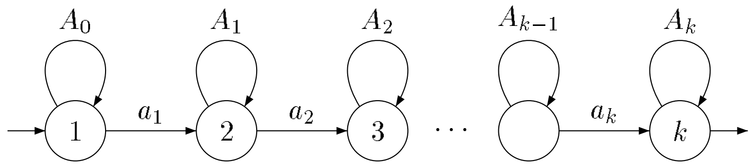
Il est intéressant de voir que les langages de niveau $3/2$ ont de bonnes propriétés topologiques.

Proposition 5.37 *Le niveau $3/2$ de la hiérarchie de Straubing-Thérien est stable par fermeture profinie.*

Preuve. Comme le niveau $3/2$ est stable par union finie et comme la fermeture profinie d'une union finie est l'union des fermetures, il suffit de prouver le résultat pour un langage L du type

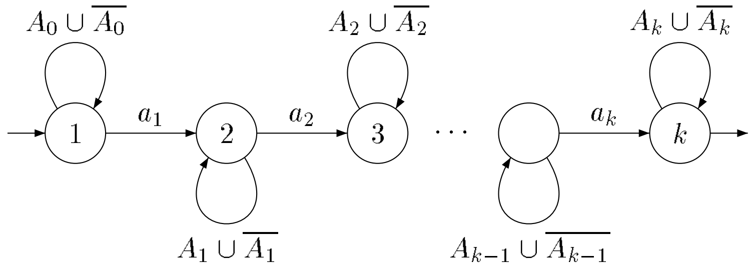
$$L = A_0^* a_1 A_1^* a_2 \cdots a_k A_k^*$$

Considérons donc l'automate $\mathcal{A}$ suivant :



Par définition de L, on a $L(\mathcal{A}) = L$. Considérons l'automate $\mathcal{A}_0 =$ obtenu en appliquant l'algorithme FermetureGA* 2.57 à $\mathcal{A}$. Montrons que $\mathcal{A}_0$ est localement trivial.

La première étape consiste à ajouter les transitions duales à l'intérieur des composantes fortement connexes et donc de considérer l'automate $\mathcal{A}_1$ suivant :



Ensuite, dans $\mathcal{A}_1$, on ajoute des ε-transitions entre i et j si et seulement s'il existe un mot $u \in \tilde{A}^*$ tel que $i \cdot_{\mathcal{A}_1} u = j$ et $u\pi = 1$. Soit (i, ε, j) une ε-transition ainsi ajoutée. Comme dans $\mathcal{A}_1$ toutes les lettres réalisent une fonction croissante de l'ensemble des états dans lui-même, $i \leq j$. Or, éliminer les ε-transitions ainsi construites n'introduit pas de nouveau cycle dans l'automate, donc ne modifie pas les composantes fortement connexes. Il en résulte que l'automate $\mathcal{A}_2$ obtenu en éliminant toutes ces ε-transitions est localement trivial. La dernière étape de l'algorithme, pour obtenir $\mathcal{A}_0$, consiste à supprimer dans $\mathcal{A}_2$ toutes les transitions étiquetées par une lettre de $\overline{A}$, ce qui préserve la propriété de trivialité locale. Par conséquent, $\mathcal{A}_0$ est un automate localement trivial reconnaissant $\mathrm{Cl}_{A^*}(L)$. Donc, d'après la proposition 5.29, $\mathrm{Cl}_{A^*}(L)$ est de niveau 3/2 dans la hiérarchie de Straubing-Thérien. □

Corollaire 5.38 *L'ensemble des points d'accumulation, pour la topologie profinie, d'un langage de niveau* 3/2 *de la hiérarchie de Straubing-Thérien est aussi de niveau* 3/2.

Preuve. Soit L un langage de niveau 3/2 et L^d l'ensemble de ses points d'accumulation pour la topologie profinie. D'après le corollaire 2.71 on a

$$L^d = \mathrm{Cl}_{A^*}(L) \cap L_0$$

où L_0 est un langage co-fini. Or, comme les langages co-finis sont tous dans le niveau 3/2, qui est fermé par intersection, L^d est dans le niveau 3/2. □

Proposition 5.39 *Un langage L de niveau* 3/2 *de la hiérarchie de Straubing-Thérien. Les assertions suivantes sont équivalentes :*

(i) L est dense pour la topologie profinie

(ii) Pour toute expression rationnelle polynomiale

$$\mathcal{E} = \bigcup_{i=1}^{n} \mathcal{A}_{0,i}^* a_{1,i} \mathcal{A}_{1,i}^* \cdots a_{k_i,i} \mathcal{A}_{k_i,i}^*$$

telle que $\mathcal{E}\nu = L$, il existe au moins un des $\mathcal{A}_{i,j}$ tel que $\mathcal{A}_{i,j} = A$.

(ii) Il existe une expression rationnelle polynomiale

$$\mathcal{E} = \bigcup_{i=1}^{n} \mathcal{A}_{0,i}^* a_{1,i} \mathcal{A}_{1,i}^* \cdots a_{k_i,i} \mathcal{A}_{k_i,i}^*$$

telle que $\mathcal{E}\nu = L$ et telle qu'au moins un des $\mathcal{A}_{i,j}$ vérifie $\mathcal{A}_{i,j} = A$.

Preuve. Comme l'existence d'une expression rationnelle polynomiale reconnaissant L est assurée par le fait que L est de niveau 3/2, il est clair que (ii) implique (iii).

Montrons que (iii) implique (i). Supposons que

$$\mathcal{E} = \bigcup_{i=1}^{n} \mathcal{A}_{0,i}^* a_{1,i} \mathcal{A}_{1,i}^* \cdots a_{k_i,i} \mathcal{A}_{k_i,i}^*$$

soit une expression polynomiale telle que $\mathcal{A}_{k,\ell}\nu \neq \emptyset$ et telle qu'il existe i et j tels que $A_{i,j} = A$. Alors $\mathcal{E}\nu$ contient un langage du type KA^*R avec K et R

non vides. Donc, d'après la proposition 2.65, $\mathcal{E}\nu$ est dense pour la topologie profinie.

Montrons pour finir que (i) implique (ii) Supposons que L soit un langage dense et que

$$\mathcal{E} = \bigcup_{i=1}^{n} \mathcal{A}_{0,i}^* a_{1,i} \mathcal{A}_{1,i}^* \cdots a_{k_i,i} \mathcal{A}_{k_i,i}^*$$

est une expression polynomiale telle que pour tout k, ℓ, $A_{k,\ell} \neq \emptyset$. Alors, d'après la preuve de la proposition 5.37, $\mathrm{Cl}_{A^*}(L)$ est reconnaissable par un automate localement trivial $\mathcal{A} = (Q, A, E, I, F)$ dont les boucles sont exactement les $A_{i,j}$. Or, comme L est dense, cet automate reconnaît exactement A^*. Raisonnons par l'absurde et supposons qu'aucun des $A_{i,j}$ ne soit égal à A. On pose alors $n = |Q|$ et $A = \{b_1, \cdots, b_m\}$. On considère de plus la relation sur Q définie par

$$p \geq_{\mathcal{A}} q \quad \text{si et seulement si} \quad \exists u \in A^* \ q \in p \cdot u$$

Cette relation est réflexive, transitive et, comme $\mathcal{A}$ est localement trivial, antisymétrique. C'est donc une relation d'ordre. On considère, enfin, la relation sur Q, $p >_{\mathcal{A}} q$, qui est la relation d'ordre stricte déduite de $\geq_{\mathcal{A}}$. Comme, pour tout état p, il y a au moins une lettre qui ne permet pas de boucler sur p,

$$p \cdot (b_1 b_2 \cdots b_m) > p$$

et donc

$$p \notin p \cdot (b_1 b_2 \cdots b_m) \tag{5.1}$$

Or, comme $L(\mathcal{A}) = A^*$, il existe $p_0 \in I$ tel que

$$p_0 \cdot (b_1 b_2 \cdots b_m)^n \cap F \neq \emptyset$$

Par conséquent, pour tout $k \in \{1, \ldots, n+1\}$,

$$p_0 \cdot (b_1 b_2 \cdots b_m)^k \neq \emptyset$$

Et donc, on peut choisir, pour chaque k, $p_k \in p_0 \cdot (b_1 b_2 \cdots b_m)^k$. On a de plus, d'après (5.1),

$$p_0 < p_1 < p_2 < \cdots < p_n$$

On a construit une suite de $n+1$ états tous distincts, ce qui est contradictoire avec $|Q| = n$. $\square$

Proposition 5.40 *Tout langage ouvert (pour la topologie profinie) non vide de niveau 3/2 de la hiérarchie de Straubing-Thérien est dense.*

Preuve. Soit L un langage ouvert non vide de niveau 3/2 de la hiérarchie de Straubing-Thérien. Par la proposition 5.37, $M(\mathrm{Cl}_{A^*}(L))$ est un groupe et, d'après le théorème 1.96, $M(\mathrm{Cl}_{A^*}(L))$ est apériodique. Il en résulte que $M(\mathrm{Cl}_{A^*}(L))$ est trivial et donc $\mathrm{Cl}_{A^*}(L) = \emptyset$ ou $\mathrm{Cl}_{A^*}(L) = A^*$. Or, comme $L \neq \emptyset$, on a $\mathrm{Cl}_{A^*}(L) = A^*$. □

Il existe des langages ouverts de niveau 3/2 qui ne sont pas de niveau 1/2. Par exemple le langage $L = (ab)^c$, qui est ouvert (car co-fini) et qui est de niveau 3/2 car $L = bA^* \cup a\emptyset^* aA^* \cup a\emptyset^* b\emptyset^* AA^*$, n'est pas de niveau 1/2. En effet, $b \in L$ et $ab \in a \sqcup\!\sqcup L$, donc $a \sqcup\!\sqcup L \neq L$.

5.5 Langages sans-étoile et topologie profinie

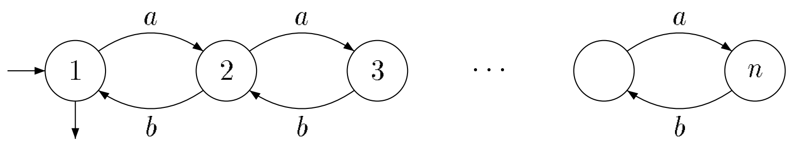
Les propriétés topologiques remarquables des langages de niveau 3/2 nous conduisent à nous interroger sur les liens plus généraux entre la topologie profinie et les langages sans-étoile.

Proposition 5.41 *Pour tout entier n, il existe un langage de niveau $n+1$ dans la hiérarchie de Straubing-Thérien qui est fermé pour la topologie profinie et qui n'est pas de niveau n.*

Preuve. Il est connu [BK78] que la suite de langages $(L_n)_{n\geq 1}$ définie par $L_1 = (ab)^*$ et $L_{n+1} = (aL_nb)^*$ vérifie :

- L_n est de niveau n dans la hiérarchie de Straubing-Thérien.
- L_n n'est pas de niveau $n-1$ dans la hiérarchie de Straubing-Thérien.

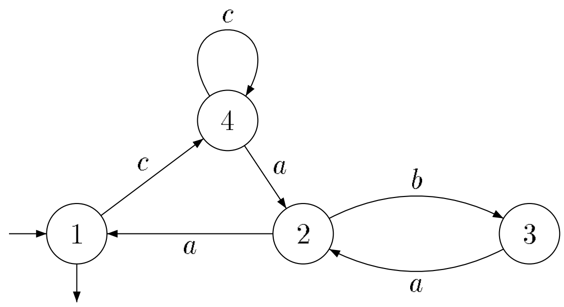
De plus l'automate minimal de L_n est :



Donc, L_n est un langage réversible et, par conséquent, fermé pour la topologie profinie. □

Proposition 5.42 *La classe des langages sans-étoile n'est pas stable par fermeture profinie.*

Preuve. Considérons le langage L donné par l'automate suivant :

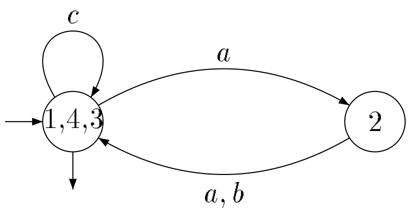


En posant, $A = \{a, b, c\}$,

$$\begin{aligned}
L_1 &= cA^*aa \\
L_2 &= A^*c\{a,c\}A^* \\
L_3 &= A^*baA^* \cup (A^*bA^*)^c \\
L_4 &= A^*caaA^* \cup A^*cabA^* \\
L_5 &= A^*baaA^* \cup A^*baabA^* \\
L_6 &= A^*aacA^*
\end{aligned}$$

on peut vérifier que $L = L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 \cap L_6$. Par conséquent, comme chaque L_i est de niveau 2, L est un langage de niveau 2.

Par ailleurs, la fermeture profinie de L est donnée par l'automate $\mathcal{A}$ suivant (en appliquant l'algorithme 2.57 FermetureGA*) :



Or, $L(\mathcal{A})$ n'est pas sans-étoile car, dans $M(L(\mathcal{A}))$, $a^\omega = a^2 \neq a^3$. □

On a de plus montré le résultat suivant.

Corollaire 5.43 *Le niveau 2 de la hiérarchie de Straubing-Thérien n'est pas stable par fermeture profinie.*

5.6 Conclusion

On peut considérer qu'il y a actuellement deux problèmes clés ouverts au sujet de la hiérarchie de Straubing-Thérien : le calcul d'une expression polynomiale (ou d'un automate localement trivial) pour les langages de niveau 3/2 et la décidabilité du niveau 2. Pour le premier problème, l'étude du niveau 1/2 suggère que la présentation par automate localement trivial est plus pertinente (d'un point de vue algorithmique) que la présentation par expression polynomiale : on code les mêmes informations en prenant moins de place. On peut comparer cette situation avec le cas des langages finis : il est plus intéressant de les coder par leurs automates minimaux que par la liste exhaustive, même triée, de tous leurs éléments.

On ne peut pas s'empêcher ici de faire le rapprochement entre les hiérarchies des langages à groupe et de Straubing-Thérien. En termes de langages, la première est une extension de la seconde et l'on vient de voir que l'extension peut aussi se lire par des représentations particulières par des automates : l'extension de la notion d'"automate trivial saturé" est "automate à groupe" et l'extension de la notion d'"automate trivial" est "automate réversible".

Par ailleurs, le corollaire 5.43 nous invite à nous interroger sur l'action de la topologie profinie (et même, en général, des topologies pro-**H**) sur les variétés de langages. En particulier, l'ensemble des langages décrits par la fermeture des langages d'une variété ou d'une variété positive est-il une variété ou une variété positive (voir section 1.9) ? D'un point de vue plus général, que peut-on dire du monoïde syntaxique de la fermeture d'un langage par rapport au monoïde syntaxique du langage ?

Chapitre 6

Algorithmique des monoïdes

Sommaire

6.1 Introduction et motivations 191
6.2 Calcul des noyaux d'un monoïde fini 192
6.2.1 Produit de Mal'cev et décidabilité 192
6.2.2 Cas d'une variété close par extension 193
6.2.3 Calcul du noyau profini 196
6.3 Conclusion 197

6.1 Introduction et motivations

Comme on l'a vu au chapitre 1 et plus précisément à la section 1.9, l'utilisation des semigroupes a facilité l'étude des langages rationnels, en donnant notamment un moyen efficace de les classifier. Nous allons ici exploiter ce lien entre variétés de langages et variétés de monoïdes finis dans l'autre sens : les automates finis permettent de résoudre de nombreux problèmes algorithmiques sur les monoïdes. Ainsi le calcul des relations de Green d'un monoïde fini (nous ne les définirons pas ici et nous invitons le lecteur intéressé à consulter, par exemple, [Pin84]), peut se faire de façon très efficace en considérant un graphe de Cayley de ce monoïde comme un automate fini [FP97].

Nous allons ici nous intéresser aux applications des résultats du chapitre 2 à l'algorithmique des monoïdes. En effet, les topologies pro-**H** du monoïde libre furent tout d'abord utilisées pour résoudre des problèmes de décidabilité concernant des variétés de monoïdes [Pin88, HMPR91, Pin95a, PR91]. Nous commencerons donc par un bref rappel de ces résultats.

6.2 Calcul des noyaux d'un monoïde fini

6.2.1 Produit de Mal'cev et décidabilité

Nous allons introduire ici quelques notions de théorie des semigroupes utiles pour préciser nos motivations algorithmiques. Nous nous en tiendrons au strict nécessaire.

morphisme relationnel

Définition 6.1 Un morphisme relationnel *entre deux monoïdes M et N est une application θ de M dans les parties de N telle que :*

- *Pour tout s et tout t de M, $(s\theta)(t\theta) \subseteq (st)\theta$.*
- *Pour tout $s \in M$, $s\theta \neq \emptyset$.*
- $1_N \in 1_M\theta$.

produit de Mal'cev

Proposition-Définition 6.2 *Soient* $\mathbf{V}$ *une variété de monoïdes finis et* $\mathbf{H}$ *une variété de groupes finis.* Le produit de Mal'cev *de* $\mathbf{V}$ *par* $\mathbf{H}$*, noté* $\mathbf{V} ⓜ \mathbf{H}$ *, est la classe de tous les monoïdes M tels qu'il existe un morphisme relationnel θ de M dans G, avec $G \in \mathbf{H}$ vérifiant*

$$1_G\theta^{-1} \in \mathbf{V}$$

La classe $\mathbf{V} ⓜ \mathbf{H}$ *est une variété de monoïdes finis.*

Notons que les notions de morphisme relationnel et de produit de Mal'cev peuvent s'étendre à d'autres types de variétés [PW96].

La portée de la notion de variété de monoïdes finis dépasse le cadre de la classification des langages rationnels et les variétés sont étudiées, pour elles-mêmes, par les mathématiciens. Le produit de Mal'cev des variétés participe à cette étude et permet, par exemple, de montrer des égalités de variétés [HR91, Pin95a] ou d'obtenir des résultats de décomposition. Nous ne discuterons pas plus ici de son intérêt mathématique.

Le problème que l'on va considérer maintenant est celui de la décidabilité de la variété $\mathbf{V} ⓜ \mathbf{H}$, où $\mathbf{H}$ est une variété de groupes finis et $\mathbf{V}$ une variété de monoïdes finis décidable (c'est à dire qu'il existe un algorithme qui, étant donné un monoïde fini, permet de dire s'il appartient à $\mathbf{V}$). On va pour cela définir la notion de *noyau d'un monoïde.*

noyau

Définition 6.3 *Soit M un monoïde fini et* $\mathbf{H}$ *une variété de groupes finis. On définit* le noyau *pro-*$\mathbf{H}$ *de M, noté $K_{\mathbf{H}}(M)$, comme étant le sous-monoïde de M suivant :*

$$K_{\mathbf{H}}(M) = \bigcap 1\theta^{-1}$$

où θ décrit l'ensemble des morphismes relationnels de M dans les groupes de $\mathbf{H}$ *.*

Le résultat très important qui nous intéresse est le suivant [RT72].

Proposition 6.4 *Soit* $\mathbf{V}$ *une variété de monoïdes finis et* $\mathbf{H}$ *une variété de groupes finis. Alors un semigroupe* S *appartient à* $\mathbf{V} \circledM \mathbf{H}$ *si et seulement si* $K_{\mathbf{H}}(S)$ *appartient à* $\mathbf{V}$.

Il résulte de cette propriété que si $\mathbf{V}$ est une variété décidable de monoïdes et si l'on sait de plus calculer $K_{\mathbf{H}}(M)$, alors la variété $\mathbf{V} \circledM \mathbf{H}$ est décidable. On va donc s'intéresser au calcul des noyaux d'un monoïde fini.

Le résultat clé de l'approche topologique de cette étude fut montré par J.-E. Pin [Pin88].

Théorème 6.5 *Soient* M *un monoïde fini engendré par* A, $m \in M$, φ *un morphisme de* A^* *dans* M *et* $\mathbf{H}$ *une variété de groupes finis. L'élément* m *appartient à* $K_{\mathbf{H}}(M)$ *si et seulement si* $1 \in \mathrm{Cl}_F^{\mathbf{H}}(m\varphi^{-1})$.

6.2.2 Cas d'une variété close par extension

On va utiliser le théorème 6.5 et les résultats du chapitre 2 pour montrer comment calculer le noyau pro-$\mathbf{H}$ d'un monoïde fini lorsque $\mathbf{H}$ est une variété de groupes finis close par extension.

Commençons par rappeler ce qu'est un *graphe de Cayley*.

Définitions 6.6 *Soit* M *un monoïde fini engendré par* A. *On appelle* graphe de Cayley engendré par A *le graphe* (M, E) *où*

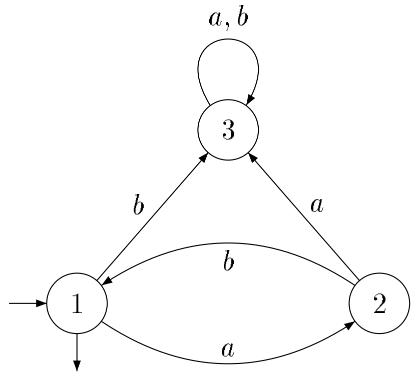
graphe de Cayley

$$E = \{(m, a, m') \mid ma = m', a \in A\}$$

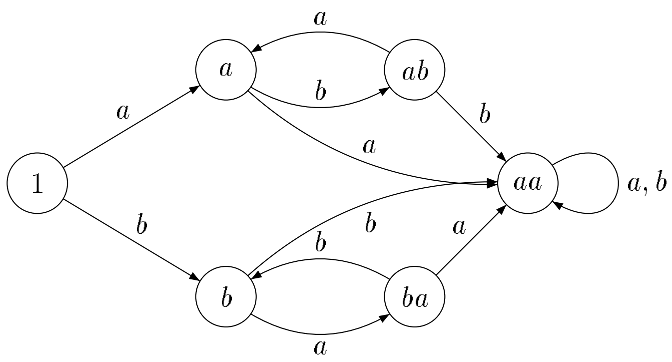
Pour tout sous-ensemble F *de* M, *on notera* $\mathcal{A}_{M,F}$ *l'automate fini*

$$\mathcal{A}_{M,F} = (M, A, E, 1_M, F)$$

Exemple 6.7 Considérons l'automate sur l'alphabet $A = \{a, b\}$ suivant :



Le graphe de Cayley (engendré par A) de son monoïde des transitions est donné par le graphe suivant :



$\triangledown$

Proposition 6.8 *Soit* $\mathbf{H}$ *une variété de groupes finis close par extension. On suppose que l'on dispose d'un algorithme qui, étant donné un automate de sous-groupe* $\mathcal{A}_H$ *à* n *états reconnaissant* H*, calcule en temps* $O(n\mathsf{f})$ *l'automate* $\mathcal{A}_{\mathrm{Cl}_F^{\mathbf{H}}(H)}$ *de la fermeture pro-*$\mathbf{H}$ *de* H*. Soit* M *un monoïde fini. On peut calculer en temps* $O(|M|\mathsf{f} + |M|^3)$ *le sous-monoïde* $K_{\mathbf{H}}(M)$ *de* M*.*

Pour cela, on utilise l'algorithme CalculNoyauH qui s'applique à $\mathcal{A}_{M,M}$.

Algorithme 6.9

1. Calculer $\mathcal{A}_0$ =FermetureH($\mathcal{A}_{M,M}$).
2. Calculer la matrice $N = (n_{m_i,m_j})$ =BenoisSakarovitch($\mathcal{A}_0$).
3. Retourner tous les $m \in M$ vérifiant $n_{1_M,m} = 1$.

$\diamondsuit$

PREUVE. L'étape 1 s'effectue en temps $O((|M|\mathsf{f}) + |M|^3)$ d'après le théorème 2.45. De plus, l'automate $\mathcal{A}_0$ a au plus $|M|$ états donc, d'après la proposition 1.71, l'étape 2 s'exécute en temps $O(|M|^3)$. Pour l'étape 3, il suffit de retourner la première ligne de la matrice, donc elle peut s'effectuer en temps $O(|M|)$. La complexité finale est donc en $O((|M|^3 + |M|\mathsf{f})$.

La correction de cet algorithme est une conséquence directe du théorème 6.5. En effet, m est retourné par l'algorithme si et seulement si il existe $u \in \tilde{A}^*$ tel que, dans $\mathcal{A}_0$, $1_M \cdot u = m$ et $u\pi = 1$. Or, comme l'algorithme FermetureH ne change pas les composantes fortement connexes, cela équivaut à dire que dans l'automate obtenu en appliquant l'algorithme FermetureH à l'automate émondé déduit de $\mathcal{A}_{M,m}$, $1_M \cdot u = m$. Ce qui équivaut encore à dire que $1 \in \mathrm{Cl}_F^{\mathbf{H}}\ (L(\mathcal{A}_{M,m})$ et donc, d'après le théorème 6.5, que $m \in K_{\mathbf{H}}\ (M)$. $\square$

En particulier, on peut calculer le noyau pro-p d'un monoïde fini.

Corollaire 6.10 *Soit M un monoïde fini engendré par A donné par son graphe de Cayley engendré par A. On peut calculer le noyau pro-p de M en temps $O(|M|^5)$.*

Par ailleurs, on peut aussi un obtenir un algorithme efficace pour tester si un élément donné appartient au noyau pro-$\mathbf{H}$.

Proposition 6.11 *Soit $\mathbf{H}$ une variété de groupes finis close par extension. On suppose que l'on dispose d'un algorithme qui, étant donné un automate de sous-groupe $\mathcal{A}_H$ à n états reconnaissant H, calcule en temps $O(n\mathsf{f})$ l'automate $\mathcal{A}_{\mathrm{Cl}_F^{\mathbf{H}}\ (H)}$ de la fermeture pro-$\mathbf{H}$ de H. Soit M un monoïde fini et $m \in M$. On peut tester en temps $O(|M|\mathsf{f} + |M|^3)$ si $m \in K_{\mathbf{H}}\ (M)$.*

Pour cela, on utilise l'algorithme TesteNoyauH qui s'applique à $\mathcal{A}_{M,m}$ et qui teste si m appartient au noyau pro-$\mathbf{H}$ de M.

Algorithme 6.12

1. Calculer $\mathcal{A}_0$ l'automate obtenu en minimisant $\mathcal{A}_{M,m}$.
2. $\mathcal{A}_1$ =`FermetureH`($\mathcal{A}_0$).
3. $\mathcal{A}_2$ =`ProjectionA*`($\mathcal{A}_1$).
4. Tester si ε appartient à $L(\mathcal{A}_2)$.

$\diamondsuit$

PREUVE. Ici aussi, la correction de l'algorithme est une conséquence directe du théorème 6.5. La preuve de la compléxité est semblable à celle de l'algorithme 6.9. $\square$

On voit que l'algorithme ci-dessus n'est pas, dans le pire des cas, plus efficace que l'algorithme calculant tout le noyau. Cependant, une différence importante dans l'algorithme `TesteNoyauH` intervient lors de l'étape 1 : on minimise le graphe de Cayley et il est raisonnable de penser que l'automate ainsi obtenu possède beaucoup moins d'états que $\mathcal{A}_{M,m}$, ce qui laisse espérer un meilleur temps de calcul en pratique.

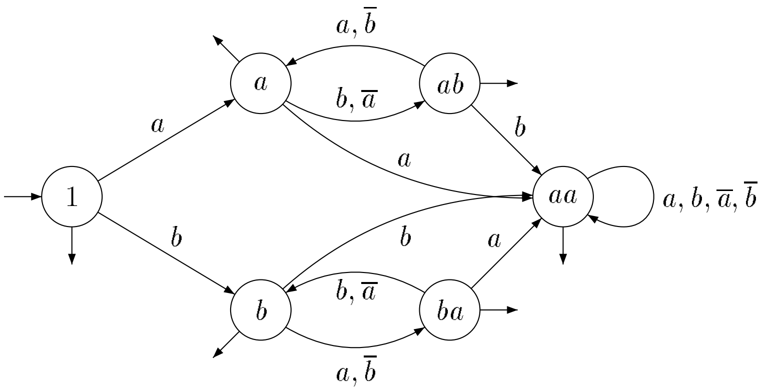
6.2.3 Calcul du noyau profini

On a vu au chapitre 2 que les algorithmes de fermeture sont bien plus simples dans le cas profini. Cela se répercute sur le calcul du noyau profini.

Proposition 6.13 *Soit M un monoïde fini engendré par A donné par son graphe de Cayley engendré par A. On peut calculer en temps $O(|M|^3)$ le noyau profini de M.*

PREUVE. L'algorithme est le même que celui de la proposition 6.8, en remplaçant `FermetureH` par `FermetureG`. La complexité découle directement de la proposition 2.49. $\square$

Exemple 6.14 On reprend le monoïde défini dans l'exemple 6.7. En appliquant l'algorithme `FermetureG` à l'automate $\mathcal{A}_{M,M}$, on obtient l'automate suivant :



Les états que l'on peut atteindre de l'état initial en lisant u tel que $u\pi = 1$ sont les états ab (en lisant $a\overline{a}$), ba (en lisant $b\overline{b}$) et aa (en lisant $ab\overline{b}\overline{a}$) et 1 (en lisant ε). En revanche, on ne peut pas atteindre ainsi les états a et b. Par conséquent, le noyau profini de ce monoïde est $\{1, ab, ba, aa\}$.

$\triangledown$

6.3 Conclusion

L'algorithme pour le calcul du noyau profini est, dans le cas le pire, asymptotiquement équivalent (en terme de complexité) à l'algorithme algébrique décrit dans [Pin88]. Cependant, notre méthode à l'avantage de s'adapter au calcul d'autres noyaux : si $\mathbf{H}$ est une variété de groupes finis close par extension, on a montré que le calcul du noyau pro-$\mathbf{H}$ peut se ramener au calcul de la fermeture pro-$\mathbf{H}$ d'un sous-groupe rationnel du groupe libre. Ce qui nous permet, en particulier, de montrer que l'on peut efficacement calculer le noyau pro-p d'un monoïde fini.

De plus, il est peut-être possible d'élargir ces méthodes, en suivant les considérations décrites dans [Del98], afin de calculer les noyaux pro-$\mathbf{G}_{\mathrm{com}}$ des monoïdes finis. En effet, même si la variété $\mathbf{G}_{\mathrm{com}}$ n'est pas close par extension, l'algorithme peut quand même être adapté et il y a espoir d'améliorer sensiblement la complexité, au moins en espace, des algorithmes de [Del98, Del01].

De même, la variété $\mathbf{G}_{\mathrm{nil}}$ n'est pas close par extension. Cependant, dans [MSW98] il est décrit comment calculer la fermeture pro-$\mathbf{G}_{\mathrm{nil}}$ d'un sous-groupe rationnel et la méthode s'adapte peut-être aussi dans ce cas.

Index des Notations

A 19
A^* 21
$A^*\mathcal{S}$ 45
$A^*\mathbf{S}$ 45
$\overline{A}$ 22
$\tilde{A}$ 22
$\mathcal{A}_H, \mathcal{A}_H^+$ 39
$\mathcal{A}^{p,q}$ 27
$\mathcal{A}^p$ 27
$\mathcal{A}_{M,F}$ 193
$\mathrm{Cl}_{A^*}, \mathrm{Cl}_F, \mathrm{Cl}_{A^*}^{\mathbf{H}}, \mathrm{Cl}_F^{\mathbf{H}}$ 71
$\mathcal{C}$ 118
$d_{\mathbf{H}}\ (x,y)$ 59
D_A^* 111
$D(S)$ 23
$e(L)$ 25
$E(S)$ 19
$F(A)$ 23
$\mathcal{F}$ 73
$\mathbf{G}, \mathbf{G}, \mathbf{G}_{\mathrm{nil}}, \mathbf{G}_{\mathrm{sol}}, \mathbf{G}_{\mathrm{com}}$ 48
$\langle H \rangle$ 18
$\mathbf{H}$ 48
$K^{-1}L, LK^{-1}$ 31
$|K|$ 18
$K_{\mathbf{H}}\ (S)$ 192
L^c 18
$L(\mathcal{A})$ 26
$m(L)$ 31
$[m]$,$[m]_{\mathcal{A}}$ 26
$M(L)$ 43
$p \cdot q$,$p \cdot_{\mathcal{A}} q$ 27
$P(L)$ 43
Pol$\mathcal{L}$ 47
Pol$\underline{\mathcal{S}}$ 47
$r_{\mathbf{H}}\ (x,y)$ 59
red$u$ 22
R_{rev} 130
$R_{S_{1/2}}$ 176
$R_{S_{3/2}}$ 183
$T_{S_{1/2}}$ 178
$T_{S_{3/2}}$ 184
u^+ 19
u^* 19
u^R 20
$u^{-1}L$,Lu^-1 31
$|u|$ 20
$\mathbf{V}$ 45
x^ω 19
X^* 19
X^+ 19
ε 20
Ⓜ 192
ν 24
π 23
τ 24
$\sqcup\!\sqcup$ 22
$\bar{\cdot}$ 22
$\sim_L$ 43
$\square$ .... marque la fin d'une preuve.
$\diamondsuit$ .... marque la fin d'un algorithme.
$\triangledown$ .... marque la fin d'un exemple.

Index des algorithmes

BenoisSakarovitch 38
CalExpST1/2 175
CalculNoyauH 195
Décompose 52
DériveH 104
FermetureConnexeH 80
FermetureG 89
FermetureGA* 92
FermetureGroupeH 78
FermetureSousgroupeGp 69
FermetureH 83
FermetureHA* 90
GroupeEngendré 76
MotsRéduits 37
ProjectionA* 38
Réduction 40
SatureAutomate 172
TesteFerméGA* 95

TesteFerméHA* 91

TesteG1/2 150

TesteDetLocRev 160

TesteNoyauH 196

Teste-pdense 68

TesteST1/2 172

Index

alphabet, 19
automate(s), 25
  à groupe, 44
  à groupe, 116, 142
  accessible, 30
  co-accessible, 30
  complet, 29
  complété d'un, 29
  de sous-groupe, 38, 76
  déterministe, 28
  dual, 38
  émondé, 30
  état d'un, 25
  extensif, 181
  graphe sous-jacent d'un, 34
  isomorphisme d', 31
  langage reconnu par un, 27
  localement à groupe, 152
  localement réversible, 125, 152, 157, 161
  localement trivial, 174, 180, 182
  minimal, 31
  morphisme d', 30
  produit d', 32
  réversible, 114
  saturé, 170

Benois
  Théorème de, 36
boule, 64

Cayley
  graphe de, 193
chemin, 26
  arrivée d'un, 26
  étiquette d'un, 26
  longueur d'un, 26
  origine d'un, 26
  réussi, 26
clôture polynomiale, 47
congruence
  des mots réduits, 23
  syntaxique, 43

décomposition unitaire-priéfixielle, 50
dense, 66, 100
distance, 62

état(s), 25
  équivalents, 32
  final, 25
  initial, 25
étiquette
  d'un chemin, 26
étoile
  d'une partie, 23
  hauteur d', 46
expression rationnelle, 23
  calcul d'une, 36
  langage reconnu par une, 24
  polynomiale, 174
  polynomiale saturée, 174
  taille d'une, 24

fermeture
  d'un langage rationnel, 72

graphe
  de Cayley, 193
  sous-jacent d'un automate, 34

groupe
  automate à, *voir* automate à groupe
  langage à, *voir* langage à groupe
  libre, 23
  séparant, 59
  topologique, 61

Hall
  Théorème de, 65
hauteur d'étoile, 46
hiérarchie
  de concaténation, 47
  de Straubing-Thérien, 47, 167
  des langages à groupe, 47, 141

idéal de mélange, 169
idempotent, 19
  engendré, 19
isomorphisme
  d'automate, 31

Kleene
  Théorème de, 27

langage
  rationnel(s)
    dense, 100
    fermeture d'un, 72
    ouvert, 63
    ouvert-fermé, 64
langage(s), 21
  à groupe, 63, 122, 142
  classe de, 45
  de Dyck, 111
  fermé, 63, 121
    automate minimal d'un, 64
  non rationnel, 111, 144
  préfixiel, 50
  rationnel(s), 27
    complémentaire d'un, 34
    dense, 179, 187
    inclusion de, 35
    intersection de, 32
    ouvert, 148, 179
    ouvert-fermé, 144
  réversible, 115
  sans-étoile, 46, 189
  unitaire, 49
  vide, 35
lettre, 20

miroir, 20, 117
monoïde, 19
  $\mathcal{R}$-trivial, 155
  apériodique, 46
  des transitions, 44
  langage reconnu par un, 43
  libre, 21
  noyau d'un, 192
  syntaxique, 43
monôme unitaire-préfixiel, 50
morphisme
  d'automate, 30
  relationnel, 192
mot, 20
  facteur, 21
  longueur d'un, 20
  produit, 21
  reconnu par un automate, 26, 35
  réduit, 22
  sous-mot, 20
  vide, 20
mélange, 22
  idéal de, 169

ouvert-fermé, 62

palindrome, 21
partie rationnelle, 23
  du groupe libre, 36, 73
  fermée, 72, 73
  fermeture d'une, 70
points d'accumulation, 102, 186
préfixe, 20, 21
produit
  d'automates, 32

- de Mal'cev, 192
- de mots, 21
- de mélange, 22

quotient, 31

rationnel
- langage, *voir* langage rationnel
- sous-groupe, 38

rationnelle
- partie, *voir* partie rationnelle

résiduel, 31

Ribes et Zalesskii
- Théorème de, 71

semigroupe, 19

séparer, 59

shuffle, 22

suffixe, 20, 21

suite
- de Cauchy, 108
- rationnelle, 107

topologie
- pro-**H** , 60
- pro-p, 66
- profinie, 73, 89, 91

transition, 25
- ε-transition, 27
- étiquette d'une, 25
- externe, 34
- interne, 34

transitions
- duales, 39

variété
- close par extension, 62
- de groupes finis, 48
- de langages, 45
- de monoïdes finis, 45
- positive, 46

Bibliographie

[AAB99] P. Abdulla, A. Annichini, and A. Bouajjani. Algorithmic verification of lossy channel systems: An appliction to the bounded retransmission protocol. In *5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–222, 1999.

[ABJ98] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *10th Int. Conf. on Computer Aided Verification*, Lecture Notes in Computer Science, pages 305–322, 1998.

[AD99] J. Almeida and M. Delgado. Sur certain systèmes d'equations avec contraintes dans un groupe libre. *Portugaliæ Math.*, 56:409–417, 1999.

[AD00] J. Almeida and M. Delgado. Sur certain systèmes d'equations avec contraintes dans un groupe libre - addenda. Soumis, 2000.

[AF98] A. Ambainis and R. Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *39th Annual Symposium on Foundations of Computer Science(FOCS98)*, pages 332–342. IEEE Computer Society, 1998.

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms*, pages 395–400. Addison-Wesley, 1974.

[Alm90] J. Almeida. Implicit operations on finite $\mathcal{J}$-trivial semigroups and a conjecture of I. Simon. *Journal of Pure and Applied Algebra*, 69:205–218, 1990.

[Ang82] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.

[Arf87] M. Arfi. Polynomial operations and rationnal languages. In *STACS*, volume 247 of *Lecture Notes in Computer Science*, pages 198–206, 1987.

[Arf91] M. Arfi. Opération polynomiales et hiérarchies de concaténation. *Theoretical Computer Science*, 91:71–84, 1991.

[AS65] A.W. Anissimov and F.D. Seifert. Zur algebraischen Charakteristik der durch kontext-freie Sprachen definierten Gruppen. *Elektron. Inform. and Control*, 8:56–63, 1965.

[Ash91] C.J. Ash. Inevitable sequences and a proof of the type II conjecture and some related decision procedures. In *Monash Conference on Semigroup Theory*, pages 31–42. World Scientific, 1991.

[BCP89] J. Berstel, M. Crochemore, and J.-E. Pin. Thue-Morse sequence and p-adic topology for the free monoid. *Discrete Mathematics*, 76:89–94, 1989.

[Ben69] M. Benois. Partie rationnelles du groupe libre. *Compte-Rendu de l'Accadémie des Sciences*, 269:1188–1190, 1969.

[Ber79] J. Berstel. *Transductions and context-free languages.* Teubner, 1979.

[Bir92] J.-C. Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43(4):185–190, 1992.

[BK78] J.A. Brzozowski and R. Knast. The dot-depth hierarchiy of star-free languages is infinite. *Journal of Computer and System Science*, 16:37–55, 1978.

[BLT97] Martin Beaudry, François Lemieux, and Denis Thérien. Finite loops recognize exactly the regular open languages. In *Automata, Languages and Programming, 24th International Colloquium (ICALP)*, volume 1256 of *Lecture Notes in Computer Science*, pages 110–120, 1997.

[BMT01] A. Bouajjani, A. Muscholl, and T. Touili. Permutation rewriting and algorithmic verification. Soumis, 2001.

[BP99] A. Brodsky and N. Pippenger. Characterization of 1-way quantum finite automata. Technical Report 99-03, University of British Columbia, 1999.

[Bra97] M. Branco. The kernel category and variants of the concatenation product. *International Journal of Algebra and Computation*, 7(4), 1997.

[Brz76] J.A. Brzozowski. Hierarchies of aperiodic languages. *Informatique théorique et Application/Theoretical Informatics and Applications*, 10:33–49, 1976.

[BS73] J.A. Brzozowski and I. Simon. Characterizations of locally testable languages. *Discrete Mathematics*, 4:243–271, 1973.

[BS86] M. Benois and J. Sakarovitch. On the complexity of some extended word problems defined by cancellation rules. *Information Processing Letters*, 23:281–287, 1986.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms.* The MIT Press, 1990.

[Coh70] R.S. Cohen. Star height of certain families of regular events. *Journal of Computer and System Science*, 4:281–297, 1970.

[Cow93] D. Cowan. Inverse monoids of dot-depth 2. *International Journal of Algebra and Computation*, 3:411–424, 1993.

[CZ00] J.-M. Champarnaud and D. Ziadi. New finite automaton constructions based on canonical derivatives. In *4th International Conference on Implementation Implementation and Application of Automata (CIAA)*, Lecture Notes in Computer Science, 2000. A paraître.

[Del98] M. Delgado. Abelian pointlikes of a monoid. *Semigroup Forum*, 56:339–361, 1998.

[Del01] M. Delgado. Commutative images of rational languages and the abelian kernel of a monoid. Soumis, 2001.

[Eil74] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974.

[Eil76] S. Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, 1976.

[EZ76] A. Ehrenfeucht and P. Zeiger. Complexity measures for regular expressions. *Journal of Computer and System Science*, 12:134–146, 1976.

[FP97] V. Froidure and J.-E. Pin. Algorithms for computing finite semigroups. In *Foundations of Computational Mathematics*, Lecture Notes in Computer Science, pages 112–126, 1997.

[GS00] C. Glasser and H. Schmitz. Languages of dot-depth 3/2. In *STACS*, volume 1770 of *Lecture Notes in Computer Science*, pages 555–566, 2000.

[Héa01a] P.-C. Héam. Automata for pro-**V** topologies. In *4th International Conference on Implementation Implementation and Application of Automata (CIAA)*, Lecture Notes in Computer Science, 2001. A paraître.

[Héa01b] P.-C. Héam. A lower bound for reversible automata. *Informatique théorique et Application/Theoretical Informatics and Applications*, 2001. A paraître.

[Héa01c] P.-C. Héam. Some complexity problems for languages of level 3/2 in Straubing-Thérien's hierarchy. Soumis, 2001.

[Héa01d] P.-C. Héam. Some topological properties of rational sets. *Journal of Automata, Languages and Combinatorics*, 2001. A paraître.

[Hal50] M. Hall. A topology for free groups and related groups. *Ann. of Math.*, 52:127–139, 1950.

[Hal84] T. Hall. Biprefix codes, inverse semigroups and syntactic monoids of injective automata. *Theoretical Computer Science*, 32:201–213, 1984.

[Has88] K. Hashiguchi. Algorithms for determining relative star height and star height. *Information and Computation*, 78(2):124–169, 1988.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Mathematical Society*, 2:326–336, 1952.

[HL00] B. Herwig and D. Lascar. Extending partial automorphism and the profinite topology on the free groups. *Transactions of the American Mathematical Society*, 352:1985–2021, 2000.

[HM00] C. Hagenah and A. Muscholl. Computing ϵ-free nfa from regular expressions in $O(n \log^2(n))$ time. *Informatique théorique et Application/Theoretical Informatics and Applications*, 34:257–277, 2000.

[HMPR91] K. Henckell, S. Margolis, J.-E. Pin, and J. Rhodes. Ash's type II theorem, profinite topology and Malcev products. *International Journal of Algebra and Computation*, 1:441–436, 1991.

[Hop71] J. Hopcroft. *The Theory of Machines and Computation*, pages 189–196. Accademic Press, 1971. An $n \log n$ algorithm for minimizing the states in a finite automaton.

[HR91] K. Henckel and J. Rhodes. The theorem of Knast, the **BG=PG** and type II conjectures. In *Monoids and Semigroups with Application*, World Scientific, pages 453–463, 1991.

[HU80] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1980.

[Kle56] S.C. Kleene. *Representation of events in nerve nets and finite automata*, pages 3–42. Automata Studies. Princeton University Press, 1956.

[Kna83] R. Knast. A semigroup characterization of dot-depth one languages. *Informatique théorique et Application/Theoretical Informatics and Applications*, 17:321–330, 1983.

[KW97] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *38th Annual Symposium of Computer Science*, pages 66–75, 1997.

[Lot83] M. Lothaire. *Combinatorics on words.* Cambridge Mathematical Library, 1983.

[LS00] S. Lombardy and J. Sakarovitch. Start height of reversible languages and universal automata. Soumis, 2000.

[MC00] Cristopher Moore and James P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237(1–2):275–306, 2000.

[McN67] R. McNaughton. The loop complexity of events. *Informtation and Control*, 11:167–176, 1967.

[McN74] R. McNaughton. Algebraic decision procedures for local testability. *Math. Syst. Theor.*, 8:60–76, 1974.

[MP85] S.W. Margolis and J.-E. Pin. Product of group languages. In *FCT Conference*, volume 194 of *Lecture Notes in Computer Science*, pages 445–455, 1985.

[MP87] S.W. Margolis and J.-E. Pin. Inverse semigroups and varieties of semigroups. *Journal of Algebra*, 110:306–323, 1987.

[MSW98] S. Margolis, M. Sapir, and P. Weil. Closed subgroups in pro-**V** topologies and the extension problem for inverse automata. *International Journal of Algebra and Computation*, 1998.

[MW00] J. Meakin and P. Weil. Subgroups of free groups: a contribution to the hanna neumann conjecture. Soumis, 2000.

[Ner58] A. Nerode. Linear automaton transformations. *Proc. AMS*, 9:541–544, 1958.

[Nic00] C. Nicaud. *Comportement en moyenne des automates finis et des langages rationnels*. Thèse, LIAFA Université Paris 7, 2000.

[Pin81] J.-E. Pin. Propriétés syntaxiques du produit non ambigü. In *Automata, Languages and Programming, 8th International Colloquium*, volume 85 of *Lecture Notes in Computer Science*, pages 483–499, 1981.

[Pin84] J.-E. Pin. *Varieties of formal languages*. Foundations of Computer Science, 1984.

[Pin88] J.-E. Pin. A topological conjecture of Rhodes. *Bulletin of the Australian Mathematical Society*, 38:421–431, 1988.

[Pin91] J.-E. Pin. Topologies for the free monoid. *Journal of Algebra*, 137(2):297–337, 1991.

[Pin92] J.-E. Pin. On reversible automata. In *Proceedings of Latin American Symposium on Theoretical Informatics (LATIN '92)*, volume 583 of *Lecture Notes in Computer Science*, pages 401–416, 1992.

[Pin94] J.-E. Pin. Polynomial closure of group languages and open sets of the Hall topology. In *Automata, Languages and Programming, 21st International Colloquium (ICALP)*, volume 820 of *Lecture Notes in Computer Science*, pages 424–435, 1994.

[Pin95a] J.-E. Pin. **BG=PG**, a success story. *Semigroups, Formal Languages and Groups*, pages 33–47, 1995.

[Pin95b] J.-E. Pin. A variety theorem without complementation. *Russian Mathem. (Iz. VUZ)*, 39:74–83, 1995.

[Pin96] J.-E. Pin. Logic, semigroups and automata on words. *Annals of Mathematics and Artificial Intelligence*, 16:343–384, 1996.

[PP86] D. Perrin and J.-E. Pin. First order logic and star-free sets. *Journal of Computer and System Science*, pages 393–406, 1986.

[PR91] J.-E. Pin and C. Reutenauer. A conjecture on the Hall topology for the free group. *Notices of the London Mathematical Society*, 23:356–362, 1991.

[PT93] J.-E. Pin and D. Thérien. The bideterministic concatenation product. *International Journal of Algebra and Computation*, 3:535–555, 1993.

[PW96] J.-E. Pin and P. Weil. Profinite semigroups, Mal'cev products and identities. *Journal of Algebra*, 182:604–626, 1996.

[PW97] J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30:1–39, 1997.

[PW01a] J.-E. Pin and P. Weil. Semidirect product of ordered semigroups. *Communications in Algebra*, 2001. A paraître.

[PW01b] J.-E. Pin and P. Weil. The wreath product principle for ordered semigroups. Technical report, Université Paris 7 et Bordeaux 1, 2001.

[Reu79] C. Reutenauer. Une topologie du monoïde libre. *Semigroup Forum*, 18:33–49, 1979.

[Reu81] C. Reutenauer. Sur mon article "une toplogie du monoide libre". *Semigroup Forum*, 22:93–95, 1981.

[Rho87] J. Rhodes. New techniques in global semigroup theory. *Semigroups and Their Applications*, pages 33–49, 1987.

[Rob96] M. Robson. Separating words with machines and groups. *Informatique théorique et Application/Theoretical Informatics and Applications*, 30(1):81–86, 1996.

[Rot95] J. Rotman. *An introduction to the theory of groups*. Springer, 4-ième edition, 1995.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *J. Res. and Develop*, 3:114–125, 1959.

[RT72] J. Rhodes and B. Tilsen. Improved lower bounds for the complexity of semigroups. *Journal of Pure and Applied Algebra*, 2:13–71, 1972.

[RZ93] L. Ribes and P.A. Zalesskii. On the profinite topology on a free group. *Bulletin of the London Mathematical Society*, 25:37–43, 1993.

[RZ94] L. Ribes and P.A. Zalesskii. The pro-p topology of a free group and algorithmic problems in semigroups. *International Journal of Algebra and Computation*, 4:359–374, 1994.

[Sch65] M.-P. Schützenberger. On finite monoids having only trivial subgroups. *Informtation and Control*, 8:190–194, 1965.

[Sch76] M.-P. Schützenberger. Sur le produit de concatenation non ambigü. *Semigroup Forum*, 13:47–75, 1976.

[Sch94] P. Schor. Algorithms for quantum computation: discrete log and factoring. In *35th IEEE Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[Sil96] P.V. Silva. On free inverse monoid languages. *Informatique théorique et Application/Theoretical Informatics and Applications*, 30:349–378, 1996.

[Sim75] I. Simon. Piecewise testable events. In *GI Conf.*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222, 1975.

[Sim90] I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72:65–94, 1990.

[ST85] H. Straubing and D. Thérien. Partially ordered finite monoids and a theorem of I. Simon. *Journal of Algebra*, 119:161–183, 1985.

[Sta83] J. Stallings. The topology of graphs. *Inventiones Mathematiæ*, 71:551–565, 1983.

[Ste85] J. Stern. Characterization of some classes of regular events. *Theoretical Computer Science*, 35:17–42, 1985.

[Ste00a] B. Steinberg. Finite state automata: a geometric approach. *Transactions of the American Mathematical Society*, 2000. A paraître.

[Ste00b] B. Steinberg. Fundamental groups, inverse Schützenberger automata, and monoid presentations. *Communications in Algebra*, 28:5235–5253, 2000.

[Ste00c] B. Steinberg. Inverse automata and profinite topologies on a free group. *Journal of Pure and Applied Algebra*, 2000. A paraître.

[Ste00d] B. Steinberg. Polynomial closure and topology. *International Journal of Algebra and Computation*, 10:603–624, 2000.

[Str85] H. Straubing. Finite semigroups varieties of the form $\mathbf{V}*\mathbf{D}$. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

[Str88] H. Straubing. Semigroups and languages of dot-depth two. *Theoretical Computer Science*, 58:361–378, 1988.

[Thé81] D. Thérien. Classification of finite monoids: the language approach. *Theoretical Computer Science*, 14:195–208, 1981.

[Thi68] G. Thierrin. Permutation automata. *Math. System Theory*, 83(2), 1968.

[Tho82] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Science*, 25:360–375, 1982.

[Wei00] P. Weil. Computing closures of finitely generated subgroups of the free group. In J.-C. Birget, S. Margolis, J. Meakin and M. Sapir, editors, *Int. Conference on Algorithmic Problems in Groups and Semigroups*, pages 289–307. Birkhaüser, 2000.