USTL — LIFL Laboratoire d'Informatique Fondamentale de Lille

N° d'ordre : 1631

THESE

Nouveau Régime

Présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

Pour obtenir le titre de

DOCTEUR en INFORMATIQUE

par

Denis DERENCOURT

AUTOMATES FINIS ET COMPOSITIONS DE CODES

Thèse soutenue le 5 décembre 1995, devant la Commission d'Examen

Président	:	Mireille CLERBOUT	Université de LILLE I
Rapporteurs	:	Jean BERSTEL	Université de PARIS VI
	:	Véronique BRUYERE	Université de MONS-HAINAUT
Examinateurs	:	Michel LATTEUX	Université de LILLE I
	:	Alain TERLUTTE	Université de LILLE III

Je remercie Mireille Clerbout qui m'a fait le plaisir d'accepter de présider ce jury.

Je remercie Jean Berstel et Véronique Bruyère, qui ont spontanément accepté de rapporter cette thèse.

Je remercie Michel Latteux et Alain Terlutte d'avoir encadré mes recherches et de m'avoir aidé durant ces quatres années.

Je remercie enfin toutes les personnes du laboratoire ou d'ailleurs qui, d'une manière ou d'une autre, ont pu contribuer à l'amélioration des conditions de réalisation de cette thèse.

Table des matières

Introduction

Cette thèse se décompose en trois parties concernant des domaines différents mais reliés entre eux par un outil de base en informatique théorique, les automates finis.

La théorie des automates a pour origine un article de S.C. Kleene datant de 1954 ([25]). Elle est issue de la tentative de décrire le comportement de certains systèmes (comme les réseaux de neurones par exemple) à l'aide d'un modèle de machine de taille finie et ne pouvant donc prendre qu'un nombre fini d'états. Par la suite, les automates finis ont été utilisés pour modéliser des problèmes informatiques liés aussi bien à l'aspect matériel qu'à l'aspect logiciel. Concernant l'aspect matériel, on peut citer la conception des circuits logiques séquentiels. Concernant l'aspect logiciel, il est connu que certaines fonctionnalités de l'éditeur de texte EMACS sont basées sur l'utilisation d'automates finis. De façon plus générale, toute application nécessitant la manipulation de chaînes de caractères, peut utiliser tous les outils algébriques mis à disposition par la théorie des automates finis. Ainsi, ces outils sont bien adaptés pour certaines phases de la compilation d'un programme, pour la compression d'un texte, ou pour l'analyse d'une séquence de molécules, qui peut être vue comme une chaîne de caractères. Le nombre élevé d'applications des automates finis rend intéressante l'étude mathématique de leur structure et de leurs propriétés pour eux-mêmes, indépendamment des applications possibles.

De façon générique, un automate fini est une structure comportant un nombre fini d'états reliés par des transitions. Chacune de ces transitions est étiquetée par un élément d'un certain ensemble. Le nom que l'on donnera à l'automate dépendra de la nature de cet ensemble. Ainsi, le mot *automate* sera plus spécialement réservé aux automates dont les transitions sont étiquetées par des lettres. Si les transitions sont étiquetées par des couples de mots, on parlera de *transducteur*. On parlera d'*automate avec multiplicités* si chaque transition est munie, en plus de l'élément qui l'étiquette, d'un nombre appelé sa multiplicité ou son poids.

Lorsque les transitions sont étiquetées par un seul élément, on peut utiliser l'automate comme moyen de décrire un ensemble ou de tester l'appartenance à celui-ci. Dans le cas où plusieurs éléments étiquettent les transitions, on peut considérer les différents éléments comme un n-uplet, et voir l'automate comme le moyen de décrire une relation, qui peut elle-même être utilisée dans certains cas pour décrire une fonction, en attribuant à certains éléments le rôle d'argument et aux autres le rôle de résultat d'une fonction. Ainsi, un transducteur doit son nom au fait que pour chaque couple de mots étiquetant une transition, le premier mot possède le rôle de mot d'entrée et le second le rôle de mot de sortie. L'automate décrit alors le comportement d'une transduction, c'est-à-dire une fonction qui à un mot associe un langage. De même, un automate avec multiplicités décrit une fonction qui a un mot associe un nombre.

Cette façon d'utiliser les automates avec multiplicités sera utilisée dans le premier chapitre pour décrire des fonctions de $[0,1]$ dans $\mathbb{R}$. Classiquement ([19]), on appelle K-Σ-automate un automate dont les transitions sont étiquetées par une lettre d'un alphabet Σ, et munies d'une multiplicité qui est un élément d'un semi-anneau K. Un K-Σ-automate définit donc une fonction qui à tout mot fini sur l'alphabet Σ, associe un élément de K. La théorie des séries formelles (cf. [6], [34]) se base sur de tels automates. Certains problèmes concernant le graphisme sur ordinateur ([12], [10], [3], [1]), en particulier la représentation et la manipulation d'images formées de pixels, peuvent être traités en utilisant des automates. Par un codage approprié, on peut représenter chaque pixel par un mot, et la couleur ou le niveau de gris de ce pixel par un nombre. On peut alors utiliser un automate avec multiplicités pour représenter la fonction qui, à chaque pixel, associe sa couleur ou son niveau de gris. C'est un problème lié à ce type de fonction, qui a motivé l'étude théorique dans [11], des *automates finis pondérés*, qui ont été définis comme des $\mathbb{R}$-Σ-automates acceptant des mots infinis en entrée. Un mot infini pouvant être vu comme la représentation d'un nombre réel de l'intervalle $[0,1]$, un automate fini pondéré définit donc une fonction de $[0,1]$ dans $\mathbb{R}$. On trouve dans [11] certains résultats concernant la continuité ou la dérivabilité de telles fonctions, ainsi que des algorithmes permettant, à partir d'une fonction donnée par un automate qui la représente, de construire un automate définissant la dérivée ou la primitive de la fonction de départ. Une famille d'automates permettant de représenter tout polynôme à une ou plusieurs variables, y est également décrite.

Le premier chapitre contient la poursuite de l'étude des fonctions de $[0,1]$ dans $\mathbb{R}$ définies par des automates finis pondérés. On donne en particulier une condition, due à la structure de l'automate, que doivent vérifier de telles fonctions. Ceci permet de mieux préciser la frontière entre les fonctions qui peuvent et celles qui ne peuvent pas être représentées par un automate fini pondéré. Ainsi, on peut montrer que les seules fonctions infiniment dérivables représentables par un automate sont les polynômes, ce qui permet de décider si une fonction donnée par un automate est infiniment dérivable ou non.
L'étude des fonctions continues est poursuivie, en particulier celles qui sont définies par des automates particuliers définis dans [11] et appelés *automates à niveaux*. Un automate à niveaux qui définit une fonction continue pour toute distribution initiale est dit *fortement continu*. On donne une construction effective, qui permet, à partir d'un automate à niveaux représentant une fonction continue, d'obtenir un automate à niveaux fortement continu représentant la même fonction, et qui contient le nombre minimal d'états que doit posséder un automate à niveaux pour représenter cette fonction. On peut donc considérer l'automate obtenu comme étant de forme normale, ce qui donne un algorithme permettant de décider si une fonction définie par un automate à niveaux est continue ou non.
Enfin, plusieurs exemples de fonctions continues définies par des automates ne différant que par la valeur du poids de certaines transitions sont donnés. Ces exemples montrent que des automates très proches peuvent définir des fonctions complètement différentes. De plus, un des exemples montre qu'un automate assez simple (quatre états) peut définir une fonction au comportement complexe, en l'occurrence une fonction continue partout et dérivable nulle part.

Le deuxième chapitre est consacré à l'autre type d'automate définissant des fonctions qui a été mentionné plus haut, c'est-à-dire aux transducteurs. Les transductions rationnelles, c'est-à-dire les transductions représentables par un transducteur ayant un nombre fini d'états et de transitions, sont des outils fondamentaux permettant la manipulation de mots, et plus généralement de langages. Comme dans de nombreux problèmes, il est intéressant de décom-

fini maximal est décidable. On possède une réponse à cette question pour certaines familles de codes finis. Ainsi, tout code préfixe-suffixe-composé, c'est-à-dire tout code fini obtenu par composition de codes préfixes et de codes suffixes, peut être plongé dans un code maximal fini. On sait (cf. [31]) que tout code de deux mots est préfixe-suffixe-composé, et que le code de quatre mots qui ne peut être plongé dans un code maximal ne peut a fortiori être préfixe-suffixe-composé. En vue de savoir si tout code de trois mots peut être ou non plongé dans un code maximal fini, il est naturel de se demander d'abord si tout code de trois mots est préfixe-suffixe-composé ou non. A. Restivo *et al.* ([31]) ont conjecturé que oui.

Le troisième chapitre contient une étude des codes préfixe-suffixe-composés et apporte une réfutation à cette conjecture. On donne dans la première partie du chapitre une méthode pour décomposer un code préfixe-suffixe composé en un nombre minimum de codes préfixes et suffixes. En utilisant cette méthode, il est possible de prouver que tout code préfixe-suffixe composé de n mots ($n \geq 3$) peut être exprimé comme la composition d'au plus $2n-3$ codes préfixes et suffixes. Pour tout entier n, cette limite est atteinte, c'est-à-dire qu'il existe un code préfixe-suffixe-composé de n mots qui ne peut pas être exprimé comme la composition de moins de $2n-3$ codes préfixes et suffixes. On donne dans la dernière partie du chapitre un contre-exemple à la conjecture de A. Restivo *et al.*, en construisant une famille de codes de trois mots qui ne sont pas préfixe-suffixe composés, le plus petit d'entre eux étant $\{a, aba, babaab\}$. Ce code est finiment complétable. On peut par contre conjecturer que le code de trois mots $\{a, aba, baababababaab\}$ ne peut pas être plongé dans un code maximal fini.

Le quatrième chapitre est consacré, suite à une conjecture formulée par Roman König, à certaines questions concernant les codes maximaux. Le problème initial qu'il étudiait était le suivant : soit $\mathcal{C}$ l'ensemble de tous les codes définis sur un alphabet fini quelconque. Soit la relation d'ordre définie sur $\mathcal{C}$ par $X \leq Y \iff X^* \subseteq Y^*$. Pour tout code X défini sur un alphabet fini A et tout code Y défini sur un alphabet fini B, on définit alors $X \vee Y$ comme la base du plus petit sous-monoïde libre de $(A \cup B)^*$ contenant $X \cup Y$ et $X \wedge Y$ comme la base de $X^* \cap Y^*$. Il a été montré ([26]) que l'ensemble $\mathcal{C}$ muni des opérations $\vee$ et $\wedge$ est un treillis. La question était de savoir s'il en était de même pour l'ensemble des codes rationnels maximaux de $\mathcal{C}$. Toujours d'après [26], on sait que $X \vee Y$ n'est pas un code maximal en général, mais est maximal si X et Y sont définis sur le même alphabet. Il restait donc le problème de la maximalité de $X \wedge Y$, la base de $X^* \cap Y^*$. Roman König avait conjecturé que si X et Y sont deux codes rationnels maximaux, alors la base de $X^* \cap Y^*$ est un code rationnel maximal (sur son alphabet).

Des contre-exemples à cette conjecture sont donnés, ainsi que certaines méthodes permettant de construire des familles de contre-exemples. L'existence de ces contre-exemples amène à se poser d'autres questions, par exemple celle de savoir quel est le treillis engendré par l'ensemble des codes rationnels maximaux, et en particulier, quel est l'ensemble des codes qui peuvent être obtenus comme base de l'intersection de deux monoïdes engendrés par des codes rationnels maximaux. On montre de façon constructive que tous les codes rationnels peuvent être obtenus de cette manière. L'autre question qui se pose est de trouver une condition nécessaire et suffisante que doivent vérifier deux codes rationnels maximaux X et Y pour que la base de $X^* \cap Y^*$ soit un code rationnel maximal. Une telle condition est donnée. Cette condition indique en particulier que pour que deux codes rationnels maximaux vérifient la propriété, il suffit qu'au moins l'un des deux soit un code préfixe et qu'au moins l'un des deux soit un code suffixe.

poser certaines opérations en opérations plus simples. Ainsi, les transductions rationnelles ont été caractérisées par M. Nivat ([29]) comme la composition d'un morphisme inverse, d'une intersection avec un langage rationnel, et d'un morphisme. Une autre caractérisation (cf. [24], [36]) consiste à exprimer une transduction rationnelle comme la composition d'un marquage de fin suivi d'une composition (que l'on peut supposer de longueur quatre, cf. [27], [28], [37]) de morphismes et de morphismes inverses. Certaines familles de transductions, comme par exemple celle des fonctions rationnelles, possèdent une caractérisation similaire, dans laquelle les morphismes et morphismes inverses peuvent être choisis d'un type particulier. Afin d'éviter la présence d'un marquage de fin, il est intéressant d'étudier la famille des transductions qui peuvent être obtenues par composition de morphismes et de morphismes inverses, et que l'on appelle transductions étoilées ou simples, car elles peuvent être représentées par un transducteur simple, c'est-à-dire ayant l'état initial comme unique état terminal.

L'étude menée dans le deuxième chapitre concerne les compositions de morphismes injectifs et de morphïsmes injectifs inverses. On y montre qu'une telle composition peut toujours être exprimée comme une composition de morphismes injectifs et de morphismes injectifs inverses de longueur quatre commençant par un morphisme, ou de longueur cinq commençant par un morphisme inverse. Si on se restreint à des compositions de morphismes préfixes et de morphismes préfixes inverses, on obtient un résultat un peu différent : une telle composition peut être exprimée comme une composition de morphismes préfixes et de morphismes préfixes inverses de longueur six commençant par un morphisme. Enfin, on généralise un résultat déjà connu pour les compositions de morphismes injectifs, en montrant que les compositions de morphismes injectifs et de morphismes injectifs inverses ne peuvent pas toutes être obtenues comme composition de morphismes préfixes ou suffixes et de morphismes préfixes ou suffixes inverses.

La notion de morphisme injectif est fortement liée à celle de code. Un code est un langage L tel que tout mot obtenu par produit de mots de L admet une unique factorisation en mots de L. Autrement dit, tout message codé par des mots de L admet un décodage unique. Ce sont des problèmes de transmission de message qui sont à l'origine de l'étude des codes. Les travaux menés vers le début des années 1950, en particulier par Shannon ([32]), et relatifs à la théorie de l'information et de la communication, concernaient surtout les codes de longueur fixe. La théorie des codes telle que nous la connaissons aujourd'hui concerne également les codes de longueur variable, et a pour origine l'idée qu'a eue M.P. Schützenberger ([33]) d'exprimer les problèmes relatifs aux codes en termes algébriques.

Une classe particulièrement intéressante de codes est celle des codes maximaux. Un code est dit maximal sur un alphabet A s'il n'est strictement inclus dans aucun autre code défini sur l'alphabet A. Lorsque l'alphabet n'est pas précisé, cela signifie que le code est maximal sur son propre alphabet. Tout code étant inclus dans un code maximal, la connaissance de la structure des codes maximaux nous renseigne sur la structure de tous les codes. Un problème général en théorie des codes est de savoir si un code possédant une certaine propriété peut être plongé dans un code maximal possédant la même propriété. Le problème a été résolu positivement pour des propriétés telles que "être un code préfixe", "être un code rationnel" ([18]), "être un code rationnel à délai de déchiffrage fini" ([8]), et plus récemment "être un code bipréfixe rationnel" ([35]), mais l'a été négativement pour la propriété "être fini". Une famille de codes finis ne pouvant être plongés dans un code fini maximal a en effet été construite par A. Restivo ([30]). Le plus petit code de cette famille contient quatre mots. On ne sait toujours pas si la propriété pour un code fini de pouvoir être plongé dans un code

Chapitre 0

Définitions et notations

Ce chapitre indique les notations et les définitions de base utilisées dans les chapitres suivants.

0.1 Monoïdes

Un *alphabet* est un ensemble dont les éléments sont des *lettres*. Un *mot* est une succession de lettres. La longueur d'un mot w est notée $|w|$.
La *concaténation* est une opération binaire associative qui, à deux mots $a_1 \cdots a_n$ et $b_1 \cdots b_p$, associe le mot $a_1 \cdots a_n b_1 \cdots b_p$. L'élément neutre de la concaténation est le mot vide, noté ε.

Un *semi-groupe* est un ensemble muni d'une opération binaire associative. Un *monoïde* est un semi-groupe possédant un élément neutre.
Par commodité, on peut considérer une lettre comme un mot de longueur 1, et l'alphabet A comme l'ensemble des mots de longueur 1 sur A. On peut ainsi identifier l'ensemble des mots de longueur finie sur l'alphabet A et le monoïde libre engendré par A, noté A^*. De même, l'ensemble des mots non vides de longueur finie sur A, c'est-à-dire $A^* \setminus \{\varepsilon\}$, est le semi-groupe libre engendré par A, noté A^+. La concaténation, considérée comme opération dont est muni un monoïde ou un semi-groupe, sera donc aussi appelée *produit* et éventuellement notée par un point si le contexte l'exige. De même, si un mot w est obtenu comme concaténation des mots $w_1, w_2, \ldots, w_n$, alors chaque w_i sera appelé *facteur* du produit $w_1 w_2 \cdots w_n$. Les mots w_1 et w_n serons dits respectivement *facteur gauche* et *facteur droit* de w.
L'ensemble des mots de longueur infinie sur A est noté A^ω.
Un ensemble de mots sur l'alphabet A est un *langage* sur A.
L'alphabet d'un langage X, défini comme l'ensemble des lettres apparaissant dans au moins un mot de X, se note $Alph(X)$.
L'ensemble des facteurs (resp. facteurs gauches, facteurs droits) d'un langage L de A^*, c'est-à-dire l'ensemble des mots de A^* qui sont facteurs (resp. facteurs gauches, facteurs droits) d'au moins un mot de L, se note $F(L)$ (resp. $FG(L)$, $FD(L)$).

Soit M et N deux monoïdes ayant respectivement ε_M et ε_N pour élément neutre. Un *morphisme* de M dans N est une fonction f de M dans N vérifiant

- $f(uv) = f(u)f(v)$, pour tout u, v de M,

- $f(\varepsilon_M) = \varepsilon_N$.

Les trois opérations suivantes, définies sur les sous-ensembles d'un monoïde M, sont connues sous le nom d'*opérations rationnelles*. Soit X et Y deux sous-ensembles de M.

- l'*union* de X et Y est notée $X \cup Y$ ou $X + Y$.
- le *produit* de X par Y, noté XY ou $X \cdot Y$, est l'ensemble des mots obtenus par produit d'un mot de X et d'un mot de Y : $XY = \{xy, x \in X, y \in Y\}$.
- l'*étoile* de X, notée X^* est l'ensemble des mots obtenus par un nombre fini de produits de mots de X : $X^* = \{x_1 x_2 \cdots x_n, x_i \in X\}$.

La famille des sous-ensembles rationnels d'un monoïde M est la plus petite famille de sous-ensembles de M contenant les sous-ensembles finis de M et fermée par union, produit et étoile.

En particulier, si A est un alphabet, un langage rationnel de A^* est un langage obtenu, en un nombre fini d'étapes, par union, produit et étoile à partir des langages finis de A^*.

Une partie X d'un monoïde M est dite *reconnaissable* dans M s'il existe un monoïde fini N et un morphisme ϕ de M dans N tel que $\phi^{-1}\phi(X) = X$.

0.2 Automates

Soit A un alphabet fini. Un *automate* est un objet permettant de représenter tout langage de A^*.

Par exemple, sur l'alphabet $A = \{0, 1, 2\}$, le langage $L = ((0 + 2)^*1(0 + 2)^*1)^*$, qui est l'ensemble des nombres pairs écrits en base 3, peut être représenté par l'automate de la figure 0.1. Les ronds sont des *états* et les flèches des *transitions* entre les états. Chaque transition est *étiquetée* par une lettre. Chaque état *initial* est pointé par une flèche (non étiquetée). Chaque état *terminal* est représenté par un double rond, ou parfois par une flèche sortante (non étiquetée).

Dans le cas le plus général, un automate est un quintuplet $\{Q, A, \delta, I, F\}$, où

- Q est un ensemble d'états,
- A est un alphabet,
- δ est un sous-ensemble de $Q \times A \times Q$,
- I est un sous-ensemble de Q,
- F est un sous-ensemble de Q.

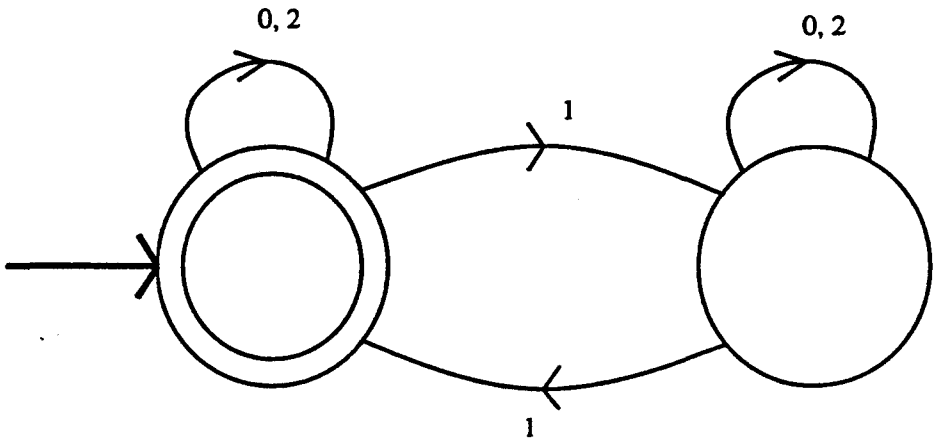


Figure 0.1 : Automate représentant le langage $L = ((0+2)^*1(0+2)^*1)^*$.

L'ensemble Q des états peut être fini ou infini.
L'alphabet A est l'alphabet sur lequel est défini le langage représenté par l'automate. Ce sont les lettres de A qui étiquettent les transitions.
L'ensemble δ représente l'ensemble des transitions de l'automate. Un triplet (p, a, q) est dans δ s'il existe dans l'automate une transition partant de l'état p, étiquetée par la lettre a, et arrivant dans l'état q. On dira parfois que la transition *lit* la lettre a.
L'ensemble I est l'ensemble des états initiaux de l'automate.
L'ensemble F est l'ensemble des états terminaux de l'automate.

Un *chemin* est une suite de transitions consécutives dans l'automate. Un chemin est étiqueté par un mot, qui est la succession des lettres étiquetant les transitions formant le chemin. Ainsi, les n transitions consécutives $(p_1, a_1, p_2), (p_2, a_2, p_3), \ldots, (p_n, a_n, p_{n+1})$, forment un chemin étiqueté par le mot $a_1 a_2 \cdots a_n$. Comme pour les transitions, on dira parfois que le chemin *lit* le mot $a_1 a_2 \cdots a_n$.
Un chemin est dit *réussi* s'il part d'un état initial et arrive dans un état terminal.
Un mot est *reconnu* par l'automate s'il existe dans l'automate un chemin réussi étiqueté par ce mot. Le langage reconnu par l'automate est l'ensemble des mots qu'il reconnait.

On peut définir pour un automate $\{Q, A, \delta, I, F\}$, une fonction, appelée fonction de transition, et que l'on notera aussi δ, qui, à un état q de Q et à une lettre a de A donnés, associe un sous-ensemble de Q, qui est l'ensemble des états atteints à partir de q en lisant a. On notera donc de façon équivalente que p est un état de $\delta(q, a)$ ou que (q, a, p) est une transition de δ.

Un automate (Q, A, δ, I, F) est *déterministe* s'il ne possède qu'un seul état initial et si pour tout état q de Q et toute lettre a de A, il existe au plus une transition de δ partant de q et étiquetée par a, ce qui signifie que $\delta(q, a)$ contient au plus un état. On pourra donc, dans le cas d'un automate déterministe, considérer la fonction de transition comme une fonction partielle de $Q \times A$ dans Q. On notera donc $\delta(q, a) = p$ plutôt que $\delta(q, a) = \{p\}$.
Un automate (Q, A, δ, I, F) est *complet* si pour tout état q de Q et toute lettre a de A, il existe au moins une transition de δ partant de q et étiquetée par a, ce qui signifie que $\delta(q, a)$ contient au moins un état.
Un état est *accessible* s'il existe un chemin dans l'automate menant d'un état initial à cet état.

Un état est *coaccessible* s'il existe un chemin dans l'automate menant de cet état à un état terminal. Un automate est émondé si tous ses états sont à la fois accessibles et coaccessibles. Un état q d'un automate déterministe reconnaissant un langage de A^* est *récurrent* si tout mot de A^* peut être lu à partir de q et mène dans un état à partir duquel on peut revenir dans l'état q en lisant un mot de A^*.

Il existe une construction effective, permettant, à partir d'un automate fini donné, de construire un automate fini déterministe et complet reconnaissant le même langage.

La définition de la reconnaissabilité d'une partie d'un monoïde est équivalente, dans le cas d'un monoïde libre à la définition suivante : un langage de A^* est dit *reconnaissable* s'il existe un automate ayant un nombre fini d'états qui reconnait ce langage.

Le théorème fondamental suivant est dû à S.C. Kleene :

Théorème 0.1 *Soit A un alphabet fini. Un langage de A^* est reconnaissable si et seulement si il est rationnel.*

Grâce à ce théorème, on peut facilement voir que l'ensemble des langages rationnels de A^* est fermé par intersection et complémentaire.

0.3 Transducteurs

Si A et B sont deux alphabets, le produit cartésien $A^* \times B^*$ est un monoïde, dont le produit est défini par $(u, v).(u', v') = (uu', vv')$, et dont les sous-ensembles sont des *relations* de $A^* \times B^*$.
Les relations rationnelles de $A^* \times B^*$ sont donc les relations de $A^* \times B^*$ qui sont obtenues, en un nombre fini d'étapes, par union, produit et étoile à partir des relations finies de $A^* \times B^*$.
Le théorème de Kleene est vrai pour les langages de A^*, mais n'est plus vrai pour les relations de $A^* \times B^*$. Toute relation reconnaissable de $A^* \times B^*$ est rationnelle, mais il existe des relations rationnelles de $A^* \times B^*$ qui ne sont pas reconnaissables. L'ensemble des relations rationnelles de $A^* \times B^*$ n'est fermé ni par intersection, ni par complémentaire.
Une relation de $A^* \times B^*$ est le graphe d'une application de A^* dans les parties de B^*. Une telle application, qui, à tout mot de A^*, associe un langage de B^*, est appelée *transduction* de A^* dans B^*. Une transduction de A^* dans B^* est rationnelle si son graphe est une relation rationnelle de $A^* \times B^*$.
On peut représenter une relation de $A^* \times B^*$ grâce à un automate dont les transitions sont étiquetées par des éléments de $A^* \times B^*$. En attribuant un rôle particulier à chaque composante des couples étiquetant les transitions, c'est-à-dire le rôle d'entrée à la première composante et le rôle de sortie à la seconde, on attribue à l'automate le rôle de *transducteur*, qui *réalise* une transduction de A^* dans B^*.
Par exemple, la relation $(((0,0)+(2,1))^*(1,0)((0,1)+(2,2))^*(1,2))^*$ est le graphe de la transduction qui, à une représentation en base 3 d'un entier n pair associe une représentation en base 3 de l'entier $n/2$ (en rajoutant éventuellement un zéro initial pour que l'argument et l'image soient deux mots de même longueur). Cette transduction est réalisée par le transducteur de la figure 0.2, qui est à rapprocher de l'automate de la figure 0.1.

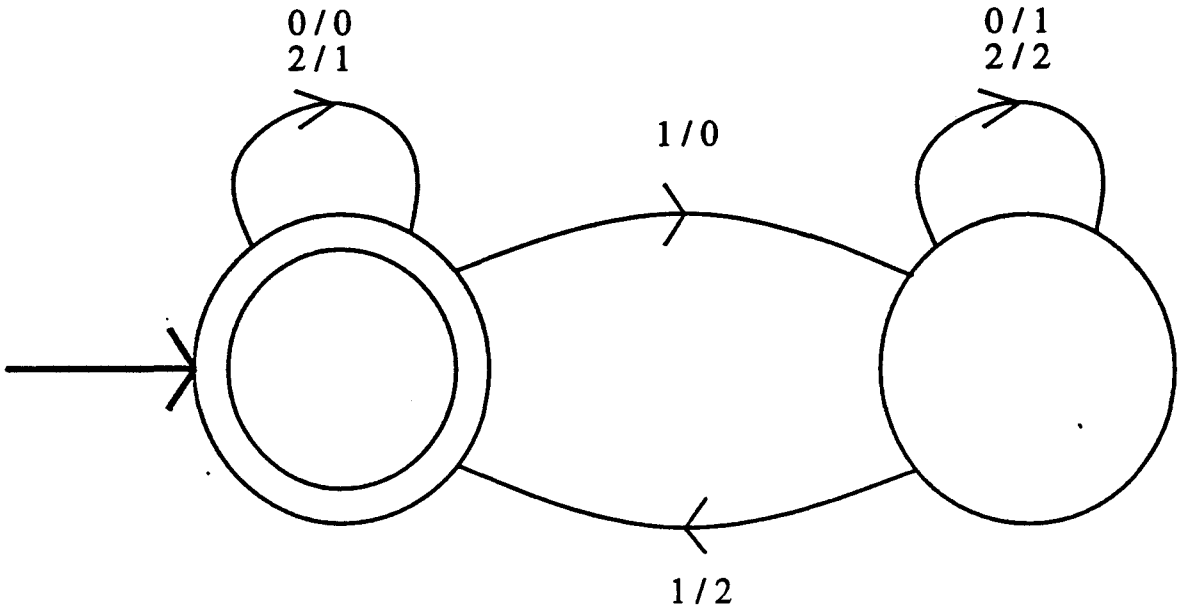


Figure 0.2 : Transducteur effectuant la division par 2 d'un nombre pair écrit en base 3.

De même qu'un langage rationnel de A^* est reconnu par un automate ayant un nombre fini d'états et de transitions (le nombre de transitions est forçément fini si le nombre d'états et le cardinal de l'alphabet A sont finis), une transduction rationnelle de A^* dans B^* est réalisée par un transducteur ayant un nombre fini d'états et de transitions (le nombre de transitions est ici potentiellement infini puisque l'ensemble des éléments pouvant étiqueter une transition est l'ensemble infini $A^* \times B^*$). On considèrera dans la suite que, si le contraire n'est pas précisé, un transducteur réalise une transduction rationnelle, et possède donc un nombre fini d'états et de transitions.
Ainsi, un transducteur est un sextuple (Q, A, B, δ, I, F) où

- Q est un ensemble (fini) d'états,
- A est un alphabet,
- B est un alphabet,
- δ est un sous-ensemble (fini) de $Q \times A^* \times B^* \times Q$,
- I est un sous-ensemble de Q,
- F est un sous-ensemble de Q.

L'alphabet A est appelé *alphabet d'entrée* et l'aphabet B *alphabet de sortie.*
L'ensemble δ représente l'ensemble des transitions du transducteur. Un quadruplet (p, u, v, q) est dans δ s'il existe dans le transducteur une transition partant de l'état p, arrivant dans l'état q, et étiquetée par le couple (u, v) de $A^* \times B^*$, c'est-à-dire, étiquetée en entrée par le mot u de A^* et étiquetée en sortie par le mot v de B^*. On dit alors que la transition *lit* le mot u et *écrit* le mot v.

L'ensemble I est l'ensemble des états initiaux du transducteur.
L'ensemble F est l'ensemble des états terminaux du transducteur.

Un *chemin* est une suite de transitions consécutives dans le transducteur. Un chemin est étiqueté par un couple de $A^* \times B^*$, qui est le produit des couples de $A^* \times B^*$ étiquetant les transitions formant le chemin. Un chemin est donc étiqueté en entrée par le mot de A^* obtenu comme produit des premières composantes des couples étiquetant les transitions formant le chemin, et il est étiqueté en sortie par le mot de B^* obtenu comme produit des deuxièmes composantes des couples étiquetant les transitions formant le chemin.
Ainsi, les n transitions consécutives $(p_1, a_1, b_1, p_2), (p_2, a_2, b_2, p_3), \ldots, (p_n, a_n, b_n, p_{n+1})$, forment un chemin étiqueté en entrée par le mot $a_1 a_2 \cdots a_n$ et en sortie par le mot $b_1 b_2 \cdots b_n$.
Comme pour une transition, on dit qu'un chemin étiqueté par le couple (u, v) de $A^* \times B^*$ *lit* le mot u et *écrit* le mot v.
Un chemin est dit *réussi* s'il part d'un état initial et arrive dans un état terminal.
Le transducteur réalise la transduction dont le graphe est l'ensemble des couples étiquetant un chemin réussi.

Soit τ une transduction de A^* dans B^*, réalisée par un transducteur T. Le *domaine* de τ, noté $Dom(\tau)$, est l'ensemble des mots de A^* ayant une image non vide par τ, c'est-à-dire l'ensemble des mots lus par un chemin réussi de T. Le *codomaine* (ou *image*) de τ, noté $Codom(\tau)$ (ou $Im(\tau)$), est l'ensemble des mots de B^* obtenus comme image par τ d'au moins un mot de $Dom(\tau)$, c'est-à-dire l'ensemble des mots écrits par un chemin réussi de T.

Le domaine (resp. le codomaine) de τ est le langage reconnu par l'automate obtenu à partir du transducteur T en ne gardant dans l'étiquette d'une transition que la première (resp. deuxième) composante du couple étiquetant cette transition. Le domaine et le codomaine d'une transduction rationnelle sont donc deux langages rationnels.

0.4 Automates finis pondérés

Les automates finis pondérés sont des automates finis dont chaque transition est munie d'un poids. Nous nous plaçons dans le cas particulier où ce poids est un nombre réel.

Un $\mathbb{R}$-A-automate (cf. [19]) est un quintuplet (Q, A, W, I, T) où :

- Q est un ensemble fini d'états,
- A est un alphabet fini,
- W est une fonction de $Q \times A \times Q$ dans $\mathbb{R}$,
- I est une fonction de Q dans $\mathbb{R}$,
- T est une fonction de Q dans $\mathbb{R}$.

La fonction W est une fonction de poids, qui associe un nombre réel à chaque transition. Un poids nul associé à une transition peut être interprété comme l'inexistence de cette transition.
La fonction I est la *distribution initiale* de l'automate. Cette fonction associe à chaque état un nombre réel. Un état est initial si son image par I est un nombre réel non nul.

La fonction T est la *distribution finale* de l'automate.
Un état est terminal si son image par T est un nombre réel non nul.

Dans la figure 0.3, on a $Q = \{q_0, q_1, q_2, q_3, q_4\}$ et $A = \{a, b\}$. Les valeurs non nulles des fonctions W, I et T sont $W(q_0, a, q_1) = 3$, $W(q1, b, q2) = 5$, $W(q0, a, q3) = 1$, $W(q3, b, q4) = 6$, $I(q0) = 2$, $T(q2) = 3$, et $T(q4) = 4$.

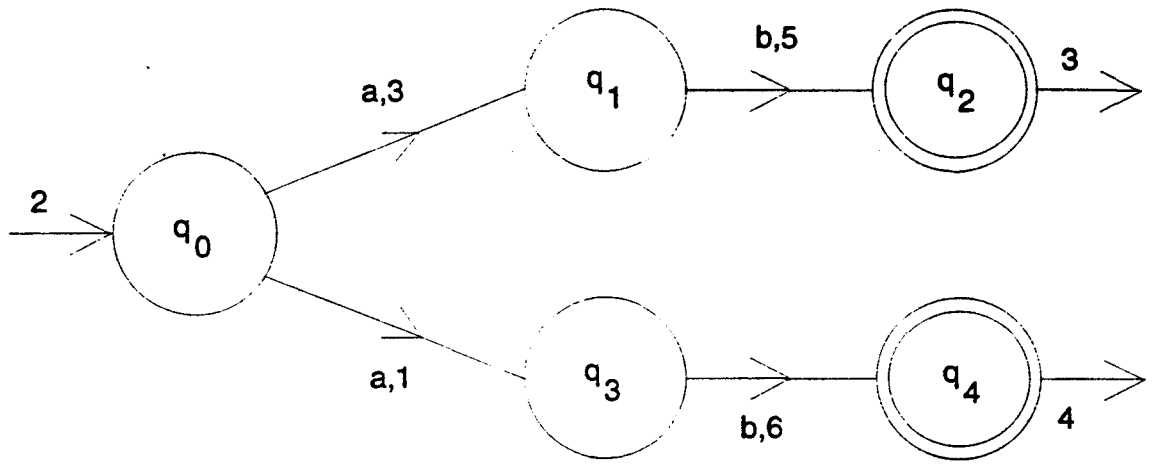


Figure 0.3 : Exemple de $\mathbb{R}$-A-automate.

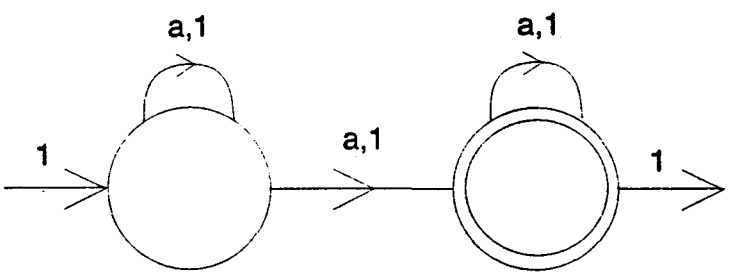


Figure 0.4 : Autre exemple de $\mathbb{R}$-A-automate.

0.5 Monoïdes de matrices

Les automates, les transducteurs et les $\mathbb{R}$-A-automates ont une structure commune : tous trois sont formés d'états et de transitions reliant ces états. Seule la nature des éléments étiquetant les transitions les différencie. On peut représenter cette structure commune à l'aide de matrices.

Plaçons nous par exemple dans le cas des $\mathbb{R}$-A-automates. Considérons un $\mathbb{R}$-A-automate (Q, A, W, I, T). On définit pour chaque lettre a de A une $Q \times Q$-matrice M_a, c'est-à-dire une matrice carrée dont les indices de ligne et de colonne sont les éléments de Q. Les éléments de chaque matrice seront les élements associés aux transitions de l'automate, c'est-à-dire ici des nombres réels. L'élément d'indice de ligne p et d'indice de colonne q de la matrice M_a est le nombre réel associé à la transition partant de l'état p, arrivant dans l'état q et étiquetée par

la lettre a, c'est-à-dire $M_a(p,q) = W(p,a,q)$. La distribution initiale I est représentée par un vecteur-ligne indicé par les éléments de Q et la distribution terminale T est représentée par un vecteur-colonne indicé par les éléments de Q. Pour un mot $w = a_1 a_2 \cdots a_n$ de A^*, considérons la matrice $M_w = M_{a_1} M_{a_2} \cdots M_{a_n}$ (Dans le cas où il existe des transitions étiquetées par le mot vide, on considèrera pour chaque lettre a de A la matrice $M_\epsilon^* M_a M_\epsilon^*$ plutôt que la matrice M_a). Le nombre réel $M_w(p,q)$ est la somme des poids de tous les chemins partant de p, arrivant dans q et étiquetés par w. Le poids du mot w est IM_wT.

Dans la figure 0.3, le poids du mot ab est $2 \cdot 3 \cdot 5 \cdot 3 + 2 \cdot 1 \cdot 6 \cdot 4 = 138$. Dans la figure 0.4, il existe pour tout entier n exactement n chemins qui lisent le mot a^n. Chacun de ces chemins a pour poids 1. Le poids du mot a^n est donc n.

On peut considérer un automate comme un $\mathcal{B}$-A-automate. Pour chaque lettre a de A, la matrice M_a est une matrice de booléens où $M_a(p,q)$ indique l'existence ou non de la transition (p,a,q) dans l'automate. Le vecteur I est un vecteur-ligne de booléens où $I(q)$ indique si q est ou non un état initial, et le vecteur Q est un vecteur-colonne de booléens où $T(q)$ indique si q est ou non un état terminal.

Un transducteur peut être vu comme un 2^{B^*}-A-automate. Pour chaque lettre a de A, la matrice M_a est une matrice de mots (en fait de langages ne contenant qu'un seul mot) de B^*, où $M_a(p,q) = w$ si (p,a,w,q) est une transition du transducteur. Le vecteur I est un vecteur-ligne où $I(q)$ est le mot vide ou l'ensemble vide selon que q est ou non un état initial, et le vecteur T est un vecteur-colonne où $T(q)$ est le mot vide ou l'ensemble vide selon que q est ou non un état terminal.

0.6 Sous-monoïdes, codes

Un *sous-semi-groupe* d'un semi-groupe M est un sous-ensemble de M stable par l'opération de M. Un *sous-monoïde* d'un monoïde M est un sous-ensemble de M stable par l'opération de M et contenant l'élément neutre de M.
Si X est un langage sur A, alors le sous-monoïde et le sous-semi-groupe engendrés par X se notent respectivement X^* et X^+.
Un sous-monoïde M d'un monoïde libre A^* est *libre* s'il existe un morphisme bijectif d'un monoïde libre B^* dans M.
Soit A un alphabet et X un langage de A^*. Une *factorisation* d'un mot w de A^* est une suite $(w_1, w_2, \cdots, w_n)$ de mots de A^* telle que $w = w_1 w_2 \cdots w_n$, et une *X-factorisation* d'un mot x de X^* est une suite $(x_1, x_2, \cdots, x_n)$ de mots de X telle que $x = x_1 x_2 \cdots x_n$.

Soit C un langage de A^*, B un alphabet, et α un morphisme de B^* dans A^* tel que $\alpha(B) = C$, et tel que l'alphabet B en bijection avec le langage C. Le langage C est un *code* sur A s'il vérifie une des conditions équivalentes suivantes :

- Tout mot de C^+ admet une seule C-factorisation.
- Le sous-monoïde C^* est un sous-monoïde libre de A^*.
- Le morphisme α est injectif.

L'ensemble minimal de générateurs d'un sous-monoïde libre M de A^* est un code, qu'on appelle la *base* de M.

Un langage P de A^* est *préfixe* (resp. *suffixe*) si et seulement si $P \cap PA^+ = \emptyset$ (resp. $P \cap A^+P = \emptyset$), c'est-à-dire, si et seulement si aucun mot de P n'est facteur gauche (resp. droit) propre d'un autre mot de P. Un langage à la fois préfixe et suffixe est dit *bipréfixe*. On peut remarquer qu'un langage préfixe (resp. suffixe) différent de $\{\varepsilon\}$ est un code.

Soit L et M deux langages de A^*. Le *quotient à gauche* (resp. *quotient à droite*) de L par M est le langage $M^{-1}L = \{u \in A^*, Mu \cap L \neq \emptyset\}$ (resp. $LM^{-1} = \{u \in A^*, uM \cap L \neq \emptyset\}$).
Un sous-monoïde N d'un monoïde M est *unitaire à droite* (resp. *unitaire à gauche*) dans M si et seulement si $N^{-1}N = N$ (resp. $NN^{-1} = N$), autrement dit, si deux mots u et v de M sont tels que u et uv (resp. u et vu) sont dans N, alors v est également dans N. Un sous-monoïde N d'un monoïde M à la fois unitaire à droite et unitaire à gauche dans M est dit *biunitaire* dans M.
La proposition suivante montre les relations existant entre les sous-monoïdes unitaires à droite (resp. unitaires à gauche, biunitaires) et les codes préfixes (resp. suffixes, bipréfixes).

Proposition 0.1 *([4, p. 46, prop. 2.5]) Un sous-monoïde de A^* est unitaire à droite (resp. unitaire à gauche, biunitaire) si et seulement si son ensemble minimal de générateurs est un code préfixe (resp. suffixe, bipréfixe). En particulier, un sous-monoïde de A^* unitaire à droite (resp. unitaire à gauche, biunitaire) est libre.*

L'opération de *composition* peut être définie pour des codes satisfaisant une certaine condition de compatibilité : les codes X et Y sont *composables* si on peut mettre l'alphabet de X en bijection avec Y. Soit $X \subseteq A^+$ et $Y \subseteq B^+$ deux codes composables. Le morphisme h tel que $h(A) = Y$, qui définit la bijection entre l'alphabet de X et Y, est un morphisme de A^* dans B^* qui est injectif puisque Y est un code. La composition de X et Y selon h, notée $X \circ_h Y$ (ou $X \circ Y$ quand il n'y a pas ambiguïté), est le code $h(X)$, c'est-à-dire le code obtenu en remplaçant chaque lettre de X par le mot correspondant de Y. On peut remarquer que $Card(X \circ Y) = Card(X)$. L'opération de composition est associative : $X \circ (Y \circ Z) = (X \circ Y) \circ Z$.
La composition de deux codes préfixes (resp. suffixes) est un code préfixe (resp. suffixe).
Si $X = Y \circ Z$ est un code préfixe (resp. suffixe), alors Y est aussi un code préfixe (resp. suffixe).

0.7 Codes maximaux

Un code C de A^* est *maximal* sur A ([4, p. 41]) s'il n'est strictement inclus dans aucun code de A^*. Lorsque l'alphabet n'est pas précisé, cela signifie que C est maximal sur son alphabet, c'est-à-dire l'ensemble des lettres apparaissant dans les mots de C.

Soit L un langage de A^*. Un mot de A^* est *complétable* (resp. *complétable à droite, complétable à gauche*) dans L s'il est facteur (resp. facteur gauche, facteur droit) d'un mot de L.
Un langage L de A^* est *dense* (resp. *dense à droite, dense à gauche*) dans A^* si tout mot de A^* est complétable (resp. complétable à droite, complétable à gauche) dans L.

Un langage L de A^* qui n'est pas dense (resp. dense à droite, dense à gauche) est dit *coupant* (resp. *coupant à droite, coupant à gauche*).
Un langage L de A^* est *complet* (resp. *complet à droite, complet à gauche*) dans A^* si le sous-monoïde L^* est dense (resp. dense à droite, dense à gauche) dans A^*, c'est-à-dire si tout mot de A^* est facteur (resp. facteur gauche, facteur droit) d'au moins un mot de L^*.

La propriété suivante nous sera très utile :

Proposition 0.2 *([4, p. 68-69, th. 5.10 et prop. 5.12]) Un code coupant de A^* est maximal sur A si et seulement si il est complet dans A^*. En particulier, tout code rationnel est coupant.*

Chapitre 1

Automates finis pondérés

Les automates finis constituent une méthode simple et fondamentale pour décrire un comportement d'entrée-sortie, en d'autres mots, pour calculer des fonctions. Une manière différente d'utiliser les automates finis pour calculer des fonctions a été présentée dans [11]. Dans cette approche, les automates finis sont utilisés pour calculer des fonctions réelles ordinaires de l'intervalle unité $[0,1]$ dans l'ensemble des nombres réels. On utilise pour cela des automates finis non déterministes ordinaires munis d'une fonction de poids, c'est-à-dire que chaque transition possède, en plus du symbole d'entrée qui l'étiquette, un nombre réel appelé son poids. De tels automates ont été appelés dans [11] des automates finis pondérés, et des $\mathbb{R}$-Σ-automates dans [19].

Un automate fini pondéré calcule une fonction réelle de $[0,1]$ dans $\mathbb{R}$ comme suit. D'abord, l'entrée x de $[0,1]$ est identifiée au mot infini binaire $bin(x)$ de $\{0,1\}^\omega$, sa représentation binaire. Ensuite, le poids associé à $bin(x)$ est calculé par l'automate. Ceci est la valeur de la fonction au point x. Pour que la définition soit correcte, certaines précautions concernant la convergence sont nécessaires.

Une classe particulièrement intéressante d'automates finis pondérés, à savoir celles des automates à niveaux, a également été définie dans [11]. Les fonctions réelles calculées par ces automates sont toujours définies. L'importance de cette classe a été démontrée dans [11].

La première section de ce chapitre est consacrée aux principales défintions et notations utilisées.

La deuxième section contient un lemme de base qui sera souvent utilisé dans les autres sections.

Dans la troisième section, nous considérons les propriétés de croissance des fonctions calculées par des automates finis pondérés. On peut remarquer, en autres choses, que, bien que l'on ne puisse réaliser exactement la fonction racine-carrée à l'aide de ces automates, on peut, a l'aide de tels automates quasiment les plus simples possibles, définir une fonction coïncidant avec la fonction racine-carrée en un nombre infini de points.

Dans la quatrième section, nous prouvons que les seules fonctions lisses, c'est-à-dire infiniment dérivables, qui peuvent être définies par un automate fini pondéré sont les polynômes, qui eux-mêmes étaient déjà connus comme pouvant être définis par des automates à niveaux (cf.

[11]).

Dans la cinquième section, nous approfondissons la connaissance que nous avons sur la façon dont les polynômes peuvent être calculés par des automates finis pondérés, en montrant qu'un polynôme de degré n nécessite un automate fini pondéré de degré au moins n, en particulier, ayant au moins $n+1$ états. Comme conséquence de ces deux résultats, nous montrons qu'on peut décider si un automate à niveaux donné définit une fonction lisse.

Dans la sixième section, nous étudions certaines propriétés relatives à la continuité des fonctions définies par un automate fini pondéré. En particulier, nous établissons une méthode pour construire des automates finis pondérés (en fait, des automates à niveaux) de plus en plus complexes calculant des fonctions continues. Cette méthode est basée sur la description d'une condition suffisante pour un automate à niveaux de définir une fonction continue. En fait notre construction produit exactement la classe des automates à niveaux fortement continus, c'est-à-dire les automates à niveaux calculant des fonctions continues pour toute distribution initiale. De plus, nous prouvons que si un automate à niveaux calcule une fonction continue pour une distribution initiale donnée, on peut construire un automate à niveaux fortement continu calculant la même fonction. Ceci donne un algorithme immédiat pour décider la continuité d'une fonction calculée par un automate à niveaux (cf. [11]).

Dans la septième section, nous appliquons la construction de la sixième section pour définir un automate à niveaux de quatre états calculant une fonction qui n'a de dérivée en aucun point. Ceci indique clairement que les automates finis pondérés sont puissants pour définirs des fonctions compliquées. Soulignons que non seulement l'automate calculant notre fonction compliquée est simple, mais aussi les calculs nécessaires à l'obtention des valeurs (ou de leur approximation) de la fonction ne sont pas difficiles ; ils sont essentiellement aussi compliqués que pour calculer les valeurs d'un polynôme de degré trois.

1.1 Définitions et notations

Soit Σ l'alphabet binaire $\{0,1\}$.
Les automates finis pondérés ont la même structure que les $\mathbb{R}$-Σ-automates. La seule différence entre les deux réside dans le fait que les automates finis pondérés admettent des mots infinis en entrée. Un automate fini pondéré a donc été défini dans [11] comme un quintuplet (Q, Σ, W, I, T) où :

- Q est un ensemble fini d'états,
- Σ est un alphabet fini,
- $W : Q \times \Sigma \times Q \to \mathbb{R}$ est une fonction de poids,
- $I : Q \to \mathbb{R}$ est la distribution initiale,
- $T : Q \to \mathbb{R}$ est la distribution finale.

On peut représenter un automate fini pondéré avec des matrices : pour chaque lettre a de Σ, on définit une $Q \times Q$-matrice W_a de réels, dans laquelle $W_a(p,q) = W(p,a,q)$ pour tout $p, q \in$

Q. De plus, I est représenté par un vecteur-ligne et T par un vecteur-colonne. Par convention, si le contraire n'est pas précisé, on suppose que $I = (1, 0, \cdots, 0)$ et $T = (0, \cdots, 0, 1)$.

Un automate fini pondéré peut être utilisé pour calculer des fonctions réelles sur l'intervalle $[0, 1]$ de la façon suivante :

Soit w un mot de $\{0,1\}^\omega$ où $w = w_1 w_2 \cdots w_n \cdots$ avec $w_i \in \{0,1\}$. Alors w est interprété comme le réel $\hat{w} = \sum_{i=1}^{\infty} w_i 2^{-i}$, et cette correspondance est biunivoque si on suppose que $w \notin \Sigma^* 01^\omega$. L'automate fini pondéré $\mathcal{A}$ définit donc les fonctions :

$$f_{\mathcal{A}} : \Sigma^\omega \to \mathbb{R}, f_{\mathcal{A}}(w) = \lim_{n \to \infty} I \cdot W_{w_1} \cdot W_{w_2} \cdots W_{w_n} \cdot T \tag{1.1}$$

$$\widehat{f_{\mathcal{A}}} : [0,1] \to \mathbb{R}, \widehat{f_{\mathcal{A}}}(x) = f_{\mathcal{A}}(w), \text{ où } \hat{w} = x \text{ et } w \notin \Sigma^* 01^\omega.$$

Ces définitions supposent l'existence de la limite (1.1). Par la suite, nous éviterons de telles considérations, soit en nous restreignant à des familles d'automates pour lesquels l'existence de la limite est garantie, ou en travaillant sous l'hypothèse que la limite existe. Une classe d'automate fini pondéré garantissant l'existence de la limite (1.1) a été présentée dans [11]. Ces automates, baptisés *automates à niveaux* ont été définis comme des automates finis pondérés vérifiant :

1. Les seules boucles dans l'automate sous-jacent de $\mathcal{A}$ sont de la forme $p \xrightarrow{a} p$.
2. $0 \leq W(p, a, p) < 1$ pour toute lettre a de Σ et tout état p de Q à partir duquel part une transition de poids non nul arrivant dans un état différent de p, et autrement $W(p, a, p) = 1$.
3. I et T sont des vecteurs de $\mathbb{R}_+^n$, où $n = Card(Q)$.
4. L'automate sous-jacent de $\mathcal{A}$ est *réduit*, c'est-à-dire n'a pas d'états inutiles.

Notons que nous modifions légèrement la définition donnée dans [11] puisque nous autorisons ici des poids négatifs dans les transitions connectantes.

Le *degré* d'un état dans un automate à niveaux $\mathcal{A}$ est le maximum des longueurs des chemins sans boucle dans $\mathcal{A}$ partant de cet état, et le degré de $\mathcal{A}$ est le plus grand degré de ses états.

Nous ne considèrerons dans la suite, sans perte de généralité, que des automates ne possédant qu'un seul état de degré 0.

Un automate fini pondéré est appelé *automate linéaire* si et seulement si pour tout entier n de $\{0, 1, \cdots, Card(Q) - 1\}$, il existe exactement un état de degré n.

1.2 Résultats de base

Le lemme suivant va beaucoup nous servir par la suite.

Lemme 1.1 *Soit $\mathcal{A}$ un automate fini pondéré définissant une fonction $\widehat{f_{\mathcal{A}}}$. Il existe un entier $t \geq 1$ et des réels $\lambda_0, \cdots, \lambda_t$ non tous nuls tels que*

$$\sum_{i=0}^{t} \lambda_i \widehat{f_{\mathcal{A}}}(\frac{x}{2^i}) = 0 \qquad \text{pour tout réel } x \text{ de } (0,1].$$

Preuve Soit x un réel de $(0,1]$ fixé et supposons que

$$\widehat{f_{\mathcal{A}}}(x) = I \cdot M_x \cdot T.$$

Si le réel x a pour représentation binaire le mot w de Σ^*, alors le réel $\frac{x}{2^n}$ a pour représentation binaire le mot $0^n w$. On a donc, pour tout entier n,

$$\widehat{f_{\mathcal{A}}}(\frac{x}{2^n}) = I \cdot W_0^n \cdot M_x \cdot T. \tag{1.2}$$

Si t est le nombre d'états de $\mathcal{A}$, alors les $t+1$ vecteurs $I, IW_0, \cdots, IW_0^t$ doivent être linéairement dépendants, i.e.

$$\sum_{i=0}^{t} \lambda_i I W_0^i = 0$$

pour des λ_i non tous nuls. Le résultat s'en déduit grâce à (1.2). □

Bien sûr, le lemme 1.1 peut également être adapté à d'autres séquences de l'argument que la séquence $(\frac{x}{2^i})_{i \geq 0}$. En effet, si on remplace W_0 par W_1 dans la preuve précédente, on obtient le résultat du lemme 1.1 pour la séquence $(\frac{x}{2^i} + 1 - \frac{1}{2^i})_{i \geq 0}$.

1.3 Croissance locale et globale

Cette section est consacrée à l'étude des différents types de croissance que l'on peut observer pour des fonctions réalisées par des automates finis pondérés. Nous considérons aussi bien la croissance locale que la croissance globale. Croissance "locale" siginifie façon de varier des valeurs d'une fonction au voisinage d'un point particulier, et est donc en étroite relation avec la dérivée (à droite ou à gauche) en un point donné. Croissance "globale" signifie croissance sur tout l'intervalle. Les deux types de croissance peuvent être basés soit sur toutes les valeurs possibles de l'argument, soit sur un ensemble discret de valeurs de l'argument. Une séquence de valeurs particulièrement intéressante est obtenue par itération de divisions par 2.
Pour une fonction f de $[0,1]$ dans $\mathbb{R}$, on appelera *séquence dichotomique* de f, une séquence $(y_i)_{i \geq 0}$ où $y_i = f(\frac{x_0}{2^i})$ pour un réel x_0 de $]0,1]$. La *dualle* de cette séquence est une séquence $(z_i)_{i \geq 0}$, définie par la condition $z_i = f(1 - \frac{1-x_0}{2^i})$.

Nous nous intéressons tout d'abord aux automates linéaires de degré 1.

Lemme 1.2 *Toute fonction continue définie par un automate linéaire de degré 1 est monotone.*

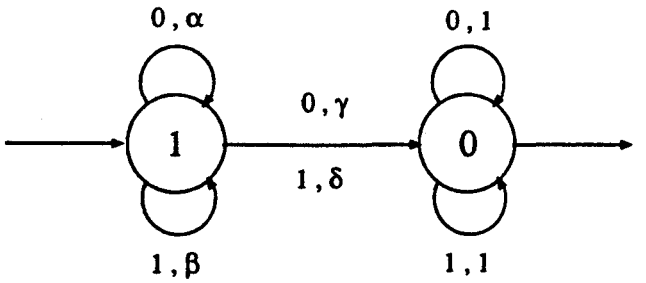


Figure 1.1 : Automate linéaire de degré 1

Preuve Considérons l'automate linéaire général de degré 1 de la figure 1.1.

Comme montré dans [11], il définit une fonction continue si et seulement si il définit une fonction continue au point 1/2, c'est-à-dire si et seulement si $\alpha+\beta = 1$ ou $\delta(1-\alpha) = \gamma(1-\beta)$. De plus, dans le dernier cas, il définit une fonction constante.

Si la dernière condition est satisfaite, le résultat est vérifié.
Si elle n'est pas satisfaite, considérons un réel $\delta > \gamma\frac{1-\beta}{1-\alpha}$.

Décomposons $\mathcal{A}$ en deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$ montrés dans la figure 1.2 (l'autre cas étant symétrique).

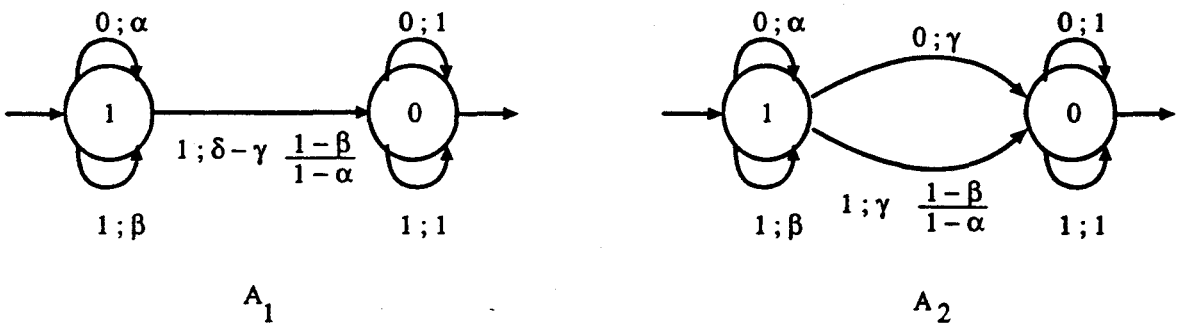


Figure 1.2 : Automates $\mathcal{A}_1$ and $\mathcal{A}_2$

Puisque $f_{\mathcal{A}} = f_{\mathcal{A}_1} + f_{\mathcal{A}_2}$ et $f_{\mathcal{A}_2}$ est une fonction constante, il suffit de prouver le lemme pour des automates de type $\mathcal{A}_1$, où le poids de la transition connectante est un réel arbitraire ε. On peut voir directement qu'un tel automate définit une fonction croissante de façon monotone, puisque

1. chaque entrée 1 ajoute quelquechose de l'état 1 au poids de l'état 0;
2. les poids de l'état 0 sont 1;
3. par continuité, $f_{\mathcal{A}_1}(01^\omega) = \alpha\frac{\varepsilon}{1-\beta} = \varepsilon = f_{\mathcal{A}_1}(10^\omega)$.

□

Il n'est pas difficile de voir que pour les automates de type $\mathcal{A}_1$, l'hypothèse de continuité $\alpha + \beta = 1$ dans le lemme 1.2 peut être affaiblie à l'hypothèse $\alpha + \beta \leq 1$.

Le lemme suivant est un outil caractérisant les séquences dichotomiques des automates linéaires de degré 1.

Lemme 1.3 *Soit $\mathcal{A}$ un automate linéaire de degré 1 tel que la transition connectante est étiquetée par le symbole d'entrée 1 seulement (cf Fig. 1.3). Alors, toute séquence dichotomique de $f_{\mathcal{A}}$ est celle d'une fonction de la forme $f(x) = Kx^q$ avec $q, K \in \mathbb{R}_+$.*

Preuve Soit x_0 un réel de $(0, 1]$ et $\mathcal{A}$ l'automate de la figure 1.3. Clairement, on a

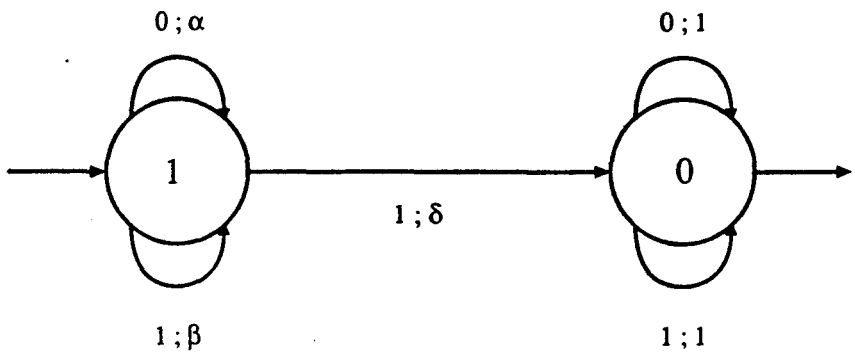


Figure 1.3 : Automate $\mathcal{A}$

$$f_{\mathcal{A}}(\tfrac{x_0}{2^i}) = \alpha^i f_{\mathcal{A}}(x_0) \qquad \text{pour } i \geq 0.$$

D'autre part,

$$f(\tfrac{x_0}{2^i}) = K(\tfrac{x_0}{2^i})^q = (2^{-q})^i f(x_0) \qquad \text{pour } i \geq 0,$$

et donc en choisissant $q = \log_2(\frac{1}{\alpha})$, le résultat s'ensuit. □

En combinant les considérations des lemmes précédents (et leurs preuves), on obtient

Théorème 1.1 *Pour tout automate linéaire $\mathcal{A}$ de degré 1 satisfaisant $f_{\mathcal{A}}(0) \leq f_{\mathcal{A}}(1)$ (resp. $f_{\mathcal{A}}(0) \geq f_{\mathcal{A}}(1)$), ses séquences dichotomiques (resp. dualles de ses séquences dichotomiques) sont celles des fonctions de la forme*

$$K_0 + K_1 x^q \tag{1.3}$$

avec $K_0, K_1, q \in \mathbb{R}_+$.

Le théorème 1.1 caractérise quelle sorte de croissance "globale discrète" peut être observée dans une fonction définie par un automate linéaire de degré 1. C'est toujours de l'ordre de x^q pour un certain $q \geq 0$. Il est également immédiat de voir que tous ces ordres de croissance, ou toutes les séquences dichotomiques de fonctions de (1.3), sont réalisables par un tel automate linéaire. De plus, d'après la caractérisation de la continuité dans [11], ils peuvent être réalisés dans la famille des automates définissant des fonctions continues. Pourtant, parmi les fonctions de (1.3), seules celles avec $q = 1$ sont effectivement des fonctions définies par des automates linéaires de degré 1 sur tout l'intervalle. Ceci se déduit directement des considérations de [11].

Exemple 1.1

Considérons l'automate de la figure 1.3. En choisissant $\alpha = 2^{-n}$, $\beta = 1 - \alpha$ et $\delta = 1 - \beta$ on obtient un automate pour lequel $f_{\mathcal{A}}(\frac{1}{2^i}) = (\frac{1}{2^i})^n$, pour $i \geq 0$. Par conséquent, $f_{\mathcal{A}}$ coïncide avec la fonction puissance-n pour les points considérés (mais pas sur tout l'intervalle).

De même, en choisissant $\alpha = \delta = \frac{1}{\sqrt{2}}$ et $\beta = 1 - \delta$ on obtient un automate réalisant une fonction continue satisfaisant $f_{\mathcal{A}}(\frac{1}{2^i}) = \sqrt{(\frac{1}{2^i})^n}$, pour $i \geq 0$.

Nous avons vu que la croissance globale pour les fonctions définies par des automates linéaires de degré 1 est assez restreinte. L'exemple suivant montre que localement, la croissance peut être extrêmement rapide.

Exemple 1.2

Considérons l'automate linéaire de la figure 1.4.

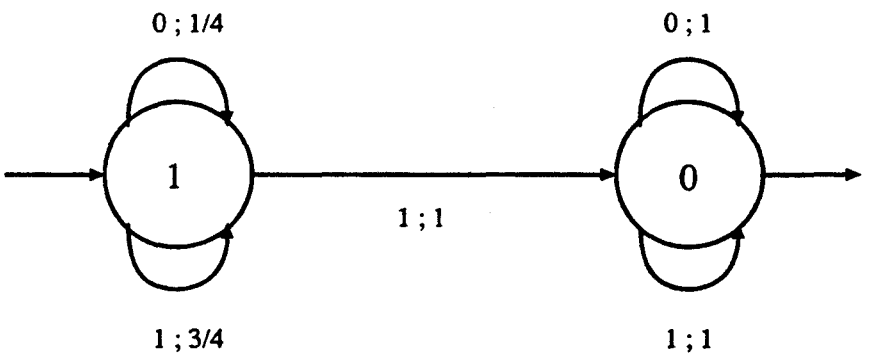


Figure 1.4 : Automate $\mathcal{A}$.

Il définit une fonction continue monotone croissante. De plus,

$$f_{\mathcal{A}}(1^\omega) = 1 \cdot \frac{1}{1-\frac{3}{4}} = 4.$$

$$f_{\mathcal{A}}(1^{n+1}0^\omega) = \sum_{i=0}^{n} (\frac{3}{4})^i = \frac{1-(\frac{3}{4})^{n+1}}{1-\frac{3}{4}} = 4 - 3(\frac{3}{4})^n.$$

Par conséquent,

$$\frac{f_{\mathcal{A}}(1^\omega) - f_{\mathcal{A}}(1^{n+1}0^\omega)}{d(1^\omega, 1^{n+1}0^\omega)} = \frac{3 \cdot (\frac{3}{4})^n}{(\frac{1}{2})^{n+1}} = 6 \cdot (\frac{3}{2})^n,$$

d'où la dérivée à gauche au point 1 est égale à ∞.

Considérons maintenant des automates finis pondérés arbitraires. Il a été montré dans [11] que les fonctions de (1.3) pour lesquelles $q = 1$, ou plus généralement tous les polynômes, sont définissables par des automates à niveaux. Par contraste avec cela, nous allons montrer que la fonction racine-carrée ne peut pas être réalisée par un automate fini pondéré. Ceci est à comparer avec la seconde partie de l'exemple 1.1.

Lemme 1.4 *la fonction $f : [0,1] \to \mathbb{R}$ définie par $f(x) = \sqrt{x}$ n'est pas réalisable par un automate fini pondéré.*

Preuve Nous donnons ici une preuve directe. Dans la suite, nous verrons comment ce résultat peut être retrouvé comme conséquence du théorème 1.2.

Supposons au contraire que $f = f_{\mathcal{A}}$ pour un automate fini pondéré $\mathcal{A}$. Soit x un réel arbitraire de l'intervalle $[0,1)$ et considérons la séquence $(x_i)_{i\geq 0}$ définie par

$$\begin{cases} x_0 &= x \\ x_{i+1} &= \frac{1}{2} + \frac{1}{2}x_i, \qquad\qquad \text{pour } i \geq 0. \end{cases}$$

Alors, clairement, $x_i = 1 - \frac{1}{2^i} + \frac{1}{2^i}x$ pour $i \geq 0$. De plus, si $f_{\mathcal{A}}(x_0) = \pi M_\omega \eta$, alors $f_{\mathcal{A}}(x_i) = \pi M_1^i M_\omega \eta$. Par conséquent, dans l'esprit du lemme 1.1, il existe un entier t (inférieur ou égal au nombre d'états de $\mathcal{A}$) et des nombres réels $\lambda_0, \cdots, \lambda_t$ non tous nuls tels que

$$\sum_{i=0}^{t} \lambda_i f_{\mathcal{A}}(x_i) = 0.$$

Ici, les λ_i sont indépendants de x de sorte que, d'après l'hypothèse $f = f_{\mathcal{A}}$, on obtient

$$\sum_{i=0}^{t} \lambda_i f(1 - \frac{1}{2^i} + \frac{1}{2^i}x) = 0 \qquad \text{pour tout réel } x \text{ de } [0,1[,$$

où au moins un λ_i est non nul. On peut écrire ceci

$$\sum_{i=0}^{t} \delta_i \sqrt{\alpha_i + x} = 0 \qquad \text{pour tout réel } x \text{ de } [0,1[,$$

où $\alpha_t > \alpha_{t-1} > \cdots > \alpha_0 = 0$ et au moins un δ_i est non nul. En prenant la dérivée $n$ième des deux côtés, on obtient

$$\sum_{i=0}^{t} \lambda_i (\delta_i + x)^{\frac{1}{2}-n} = 0 \qquad \text{pour tout réel } x \text{ de } [0,1[.$$

En particulier, en choisissant $x = 0$, on déduit l'identité

$$\sum_{i=0}^{t} \delta_i \frac{\sqrt{\alpha_i}}{\alpha_i^n} = 0 \qquad \text{pour tout } n \geq 1.$$

Mais ceci est impossible puisque au moins un δ_i est strictement positif et les α_i sont disjoints deux à deux et positifs. □

La dernière observation de cette section est que les automates finis pondérés ne peuvent pas définir de fonctions exponentielles. On a même plus que cela : la séquence dichotomique d'une fonction définie par un automate fini pondéré ne peut pas être celle d'une fonction exponentielle. On peut toutefois remarquer que pour approximer avec un précision donnée une fonction exponentielle ou toute fonction admettant une série de Taylor, il suffit de prendre un automate fini pondéré réalisant un polynôme de Taylor de la fonction considérée.

Lemme 1.5 *Il n'existe pas d'automate fini pondéré $\mathcal{A}$ tel que pour un réel x de $]0,1]$ et un réel $k > 0$, on ait*

$$f_{\mathcal{A}}(\frac{x}{2^i}) = k^{\frac{x}{2^i}} \qquad \textit{pour } i \geq 0. \tag{1.4}$$

Preuve Supposons au contraire qu'on puisse trouver $\mathcal{A}$, x et k tels que 1.4 soit vérifiée. Soit $I = \{2^{-i} | i \geq 0\}$. En appliquant le lemme 1.1 on conclut qu'il existe un entier t et des nombres réels λ_i non tous nuls tels que

$$\sum_{i=0}^{t} \lambda_i f_{\mathcal{A}}(\frac{x}{2^i}) = 0 \qquad \text{pour } x \in I.$$

D'après notre hypothèse, on peut réécrire ceci

$$\sum_{i=0}^{t} \lambda_i k^{\frac{x}{2^i}} = 0 \qquad \text{pour } x \in I,$$

ou de façon équivalente,

$$\sum_{i=0}^{t} \lambda_i (k^{\frac{x}{2^t}})^{2^{t-i}} = 0 \qquad \text{pour } x \in I.$$

Mais ceci signifie qu'une équation polynômiale de degré au plus 2^t possède une infinité de racines, ce qui est contradictoire. □

La preuve précédente montre que le lemme 1.5 est encore valide dans une forme plus forte : on peut remplacer dans le lemme "automate fini pondéré" par "automate fini pondéré de n états" et (1.4) par

(1.4') $f_{\mathcal{A}}(\frac{x}{2^i}) = k^{\frac{x}{2^i}}$ pour $i = 1, \cdots, N$,

où N est une constante qui ne dépend que de n.

Par conséquent, nous avons montré qu'il est non seulement impossible de réaliser des fonctions exponentielles avec des automates finis pondérés, mais aussi que même une séquence dichotomique d'une fonction définie par un automate fini pondéré ne peut suivre la fonction exponentielle que pendant un nombre borné d'étapes. En particulier, cela signifie que les fonctions définies par des automates finis pondérés ne peuvent pas avoir de croissance "globale discrète" exponentielle. Ceci est à comparer avec l'exemple 1.2, qui montre que la croissance locale peut être arbitrairement rapide.

1.4 Fonctions lisses

Dans la section précédente, nous avons montré que les automates finis pondérés ne peuvent pas définir des fonctions communes telles que la fonction racine-carrée ou la fonction exponentielle. D'autre part, il a été montré dans [11] que toutes les fonctions polynômiales (d'une ou plusieurs variables) peuvent être réalisées par des automates finis pondérés possédant une structure particulière, à savoir les automates à niveaux (d'un certain type). Nous montrons ici que les polynômes sont les seules fonctions lisses (d'une variable) qui peuvent être réalisées par des automates finis pondérés généraux. Ici *lisse* signifie que la fonction doit posséder toutes ses dérivées dans un intervalle ouvert contenant proprement l'intervalle $[0, 1]$. Par conséquent, la fonction exponentielle est lisse, mais la fonction racine-carrée ne l'est pas.

Théorème 1.2 *Soit f une fonction lisse et $\mathcal{A}$ un automate fini pondéré. Si*

$$f(x) = f_{\mathcal{A}}(x) \qquad \textit{pour tout réel } x \textit{ de } [0, 1], \tag{1.5}$$

alors f est un polynôme.

Preuve Supposons que (1.5) soit vraie. Utilisons de nouveau le lemme 1.1 pour conclure l'existence d'un entier t et de nombres réels $\lambda_0, \cdots, \lambda_t$ non tous nuls tels que

$$\sum_{i=0}^{t} \lambda_i f_{\mathcal{A}}(\frac{x}{2^i}) = 0 \qquad \text{pour tout réel } x \text{ de } (0,1]. \tag{1.6}$$

Puisque f est lisse, elle possède une série de Taylor

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n \qquad \text{pour } |x| \leq \delta,$$

où δ est une constante positive. Par conséquent

$$f(\frac{x}{2^i}) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!(2^i)^n} x^n \qquad \text{pour } |x| \leq \delta.$$

D'après (1.5) et (1.6), on obtient

$$\sum_{i=0}^{t} \lambda_i \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!(2^i)^n} x^n = 0 \qquad \text{pour } |x| \leq \delta,$$

ou de façon équivalente,

$$\sum_{n=0}^{\infty} (\sum_{i=0}^{t} \lambda_i \frac{f^{(n)}(0)}{n!(2^i)^n}) x^n = 0 \qquad \text{pour } |x| \leq \delta.$$

D'après l'unicité de la série de Taylor, ceci implique que

$$(\sum_{i=0}^{t} \lambda_i (\frac{1}{2^i})^n) \cdot \frac{f^{(n)}(0)}{n!} = 0 \qquad \text{pour } n \geq 0. \tag{1.7}$$

Supposons maintenant que pour $t+1$ valeurs différentes de n_j, le premier facteur du membre gauche de (1.7) soit nul, i.e.

$$\sum_{i=0}^{t} \lambda_i (\frac{1}{2^i})^{n_j} = 0 \qquad \text{pour } j = 0, \cdots, t.$$

Puisque ce système linéaire d'équations a une solution non triviale, nécessairement le déterminant du système est égal à 0, en symboles

$$\begin{vmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^t \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^t \\ \vdots & & & & \\ 1 & \alpha_t & \alpha_t^2 & \cdots & \alpha_t^t \end{vmatrix} = 0, \tag{1.8}$$

où $\alpha_j = \frac{1}{2^{n_j}}$ pour $j = 0, \cdots, t$.

Mais le membre gauche de 1.8 est le bien connu déterminant de Vandermonde, et est donc différent de 0 puisque les α_j sont disjoints deux à deux.

Il s'ensuit que dans (1.7) le second facteur est égal à zéro pour toutes sauf au plus t valeurs de n. Ceci signifie que f est un polynôme sur l'intervalle $[-\delta, \delta]$, et donc également sur l'intervalle unité considéré. □

Comme nous l'avons déjà mentionné, la fonction racine-carrée n'est pas lisse et par conséquent le lemme 1.4 n'est pas un corollaire direct du théorème 1.2. Mais il peut être conclu du théorème 1.2 comme suit. Si un automate fini pondéré réalise la fonction $\sqrt{x}$, alors un autre réalise la fonction $\sqrt{\frac{1}{2}+\frac{1}{2}x}$, qui est lisse mais n'est pas un polynôme.

1.5 Décider si une fonction est lisse

Il a été montré dans [11] qu'on peut décider si un automate à niveaux donné définit une fonction continue. Nous prouvons qu'on peut également décider si un automate à niveaux définit une fonction lisse. Pour cela, nous prouvons d'abord un résultat auxiliaire qui est intéressant en lui-même : tout automate à niveaux réalisant un polynôme de degré n doit contenir un état de degré n. Par conséquent, les automates à niveaux construits dans [11] pour les polynômes sont optimaux : ils sont de taille minimale dans le sens de la définition suivante.

La *taille* d'un automate à niveaux $\mathcal{A}$ de degré n est le vecteur $(t_n, \ldots, t_0)$, où t_i est le nombre d'états de degré i dans $\mathcal{A}$. Nous définissons la relation "inférieur ou égal à", en symboles $\prec$, dans la famille des automates à niveaux, comme suit :

On a $\mathcal{A} \preceq \mathcal{A}'$ si et seulement si le vecteur $(t_n, \ldots, t_0)$ associé à $\mathcal{A}$ est lexicographiquement inférieur ou égal au vecteur $(t'_m, \ldots, t'_0)$ associé à $\mathcal{A}'$, autrement dit, soit $n < m$ soit $n = m$ et dans ce cas on a pour un certain $j \geq -1$ l'inégalité $t_j < t'_j$ et les égalités $t_k = t'_k$ pour tout $k > j$.

Clairement, pour chaque paire d'automates à niveaux $(\mathcal{A},\mathcal{A}')$, soit $\mathcal{A} \preceq \mathcal{A}'$ soit $\mathcal{A}' \preceq \mathcal{A}$, et ces deux relations sont vérifiées simultanément si et seulement si $\mathcal{A}$ et $\mathcal{A}'$ sont de la même taille.

Nous pouvons maintenant prouver le résultat suivant :

Théorème 1.3 *Tout automate à niveaux réalisant un polynôme de degré n avec une distribution initiale quelconque est de degré au moins n.*

Preuve Supposons que le théorème 1.3 soit faux, et soit n fixé de telle manière qu'il contredise de manière minimale l'énoncé du théorème. Alors il existe un polynôme

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \qquad \text{avec } a_n \neq 0$$

réalisé par un automate à niveaux (réduit) $\mathcal{A}_m$ de degré $m < n$. Nous choisissons P_n de telle manière que $\mathcal{A}_m$ soit de taille minimale. Ceci est, bien sûr, possible.

Assertion 1.1 *Si $q \xrightarrow{i,\alpha} q$ est une transition dans $\mathcal{A}_m$ et le degré de q est m, alors $\alpha = \frac{1}{2^n}$.*

Preuve Soit $\pi = (x_1, \cdots, x_s)$ le vecteur initial de $\mathcal{A}_m$, où sans perte de généralité, la première composante correspond à q. Puisque $\mathcal{A}_m$ est réduit, x_1 est non nul. L'automate $\mathcal{A}_m$ avec la distribution initiale $\pi M_0 - \alpha\pi$ réalise le polynôme $P_n(\frac{1}{2}i + \frac{x}{2}) - \alpha P_n(x)$. Mais la première composante du vecteur $\pi M_0 - \alpha\pi$ est nulle, de sorte que dans l'automate $\mathcal{A}_m$, l'état q peut

être supprimé. D'où, par le choix de P_n, le degré du polynôme $P_n(\frac{1}{2}i + \frac{1}{2}x) - \alpha P_n(x)$ doit être au plus $n-1$. Par conséquent, on a $\frac{1}{2^n}a_n - \alpha a_n = 0$, et l'assertion s'ensuit.

En vue d'achever la preuve, nommons les états de $\mathcal{A}_m$ de telle façon que exactement les t premières composantes dans π correspondent aux états de degré m. D'où, par l'assertion 1.1, dans les deux vecteurs $\pi M_0 - \frac{1}{2^n}\pi$ et $\pi M_1 - \frac{1}{2^n}\pi$, seules les composantes correspondant aux états de degré inférieur à m dans $\mathcal{A}_m$ ont des valeurs non nulles. Par conséquent, les fonctions réalisées par $\mathcal{A}_m$ avec ces distributions initiales peuvent être réalisées par un automate à niveaux de degré au plus $m-1$. Mais ces fonctions sont

$$\begin{aligned} P_n(\frac{x}{2}) - \frac{1}{2^n}P_n(x) &= \frac{a_{n-1}}{2^n}x^{n-1} + \cdots \\ P_n(\frac{1}{2} + \frac{1}{2}x) - \frac{1}{2^n}P_n(x) &= (\frac{n}{2^n}a_n + \frac{a_{n-1}}{2^n})x^{n-1} + \cdots \end{aligned}$$

La minimalité de n implique que ces polynômes sont de degré au plus $n-2$. D'où $a_{n-1} = 0 = a_n$, ce qui est contradictoire avec le choix de P_n. □

Le théorème 1.3 a une conséquence immédiate.

Corollaire 1.1 *Tout automate à niveaux réalisant un polynôme de degré n contient au moins $n+1$ états.*

Le théorème 1.3 a également un autre corollaire intéressant.

Corollaire 1.2 *Soit $\mathcal{A}_n$ un automate linéaire de degré n réalisant un polynôme de degré n. Alors les deux boucles dans l'état de degré i sont pondérées par $1/2^i$ pour $i = 0, \cdots, n$.*

Preuve Se déduit directement du théorème 1.3 et des considérations prouvant l'assertion 1.1. □

Notons que le corollaire 1.2 est en coïncidence avec l'automate construit dans [11] pour les polynômes. Notons également que dans le corollaire 1.2, il est nécessaire de supposer que les degrés de $\mathcal{A}_n$ et du polynôme coïncident, c'est-à-dire que $\mathcal{A}_n$ est de degré minimal. L'exemple suivant le montre.

Exemple 1.3

Considérons l'automate $\mathcal{A}_2$ montré dans la figure 1.2. Il réalise un polynôme de degré 0, les poids sur les boucles de l'état 1 pouvant être arbitraires. Bien sûr, l'automate de degré minimal réalisant la fonction constante est l'automate à niveaux ayant un seul état et dont les deux seules transitions sont des boucles (l'une étiquetée par 0, l'autre par 1) toutes deux pondérées par 1. Des exemples non triviaux supplémentaires montrant la nécessité de la condition précédente sont obtenus en "intégrant" le précédent automate $\mathcal{A}_2$, cf. [11].

Nous pouvons maintenant prouve notre résultat de décidabilité.

Théorème 1.4 *On peut décider si un automate à niveaux donné réalise une fonction lisse.*

Preuve Soit $\mathcal{A}$ un automate donné. Par le théorème 1.2, le problème du théorème 1.4 est équivalent à celui de décider si $f_{\mathcal{A}}$ est un polynôme. Pour cela, on calcule d'abord n, le degré de $\mathcal{A}$. Ensuite, par le théorème 1.3, nous savons que, si $f_{\mathcal{A}}$ est un polynôme, il est de degré au plus n, c'est-à-dire qu'il est de la forme

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0.$$

Les coefficients inconnus sont déterminés de manière unique en calculant $f_{\mathcal{A}}$ en $n+1$ points différents. Ceci est de nouveau dû au déterminant de Vandermonde. Soit P le polynôme ainsi obtenu. La fonction $f_{\mathcal{A}}$ est un polynôme si et seulement si elle est égale à P, ce qui peut être décidé en calculant un automate à niveaux $\mathcal{A}_P$ pour P, par exemple avec les méthodes de [11], et en testant ensuite l'équivalence de $\mathcal{A}$ et $\mathcal{A}_P$, cf. de nouveau [11]. □

1.6 Construction d'automates calculant des fonctions continues

Dans cette section nous étudions les conditions dans lesquelles un automate à niveaux définit une fonction continue.

Clairement les automates à niveaux à deux états sont des automates linéaires. D'où, au point de départ de nos considérations, nous rappelons que la continuité de la fonction définie par un tel automate (montré dans la figure 1.1) est caractérisée par la condition

$$\alpha + \beta = 1, \text{ ou } (1-\beta)\gamma = (1-\alpha)\delta, \tag{1.9}$$

où α et β sont les poids des boucles dans l'état de degré 1, cf. [11].

Soit $\mathcal{A}$ un automate à niveaux arbitraire et α et β des nombres réels de $[0,1[$ fixés. Appelons Q l'ensemble des états de $\mathcal{A}$ et n la cardinalité de Q. Pour chaque état q de Q, appelons $\mathcal{A}_q$ le sous-automate de A constitué des états de Q qui sont accessibles de q et des transitions de $\mathcal{A}$ qui relient ces états.

Nous définissons une famille $\mathcal{A}(\alpha,\beta)$ d'automates, dont les éléments peuvent être vus comme des extensions de $\mathcal{A}$, comme suit : chaque automate $\mathcal{A}_{ext}$ de $\mathcal{A}(\alpha,\beta)$ contient, en plus de tous les états et toutes les transitions de $\mathcal{A}$, un nouvel état initial r accompagné des transitions $r \xrightarrow{0,\alpha} r, r \xrightarrow{1,\beta} r$ et $r \xrightarrow{i,W(i,q)} q$ pour toute lettre i de l'alphabet $\{0,1\}$ et tout état q de Q. Ici, les $W(i,q)$ sont des nombres réels arbitraires. Il s'ensuit que chaque $\mathcal{A}_{ext}$ de $\mathcal{A}(\alpha,\beta)$ est complètement spécifié par le vecteur de dimension $2n$

$$(W(0,q), W(1,q)) \in \mathbb{R}^{2n}. \tag{1.10}$$

Comme nous avons dit, nous considérons r comme le seul état initial de $\mathcal{A}_{ext}$, et par convention, nous supposons que chaque $\mathcal{A}_{ext}$ est réduit. Par conséquent, $\mathcal{A}_{ext}$ est de degré $n+1$ si et seulement si il existe une lettre d'entrée j et un état q de $\mathcal{A}$ de degré n tels que $W(j,q) \neq 0$. Cette construction est très générale : chaque automate à niveaux peut être obtenu de cette manière à partir de l'automate à niveaux à un état.

Avec la terminologie ci-dessus, nous allons montrer

Théorème 1.5 *Soit $\mathcal{A}, \alpha$ et β fixés comme ci-dessus et supposons que pour chaque état q, $\mathcal{A}_q$ definisse une fonction continue. Alors $\mathcal{A}_{ext}$ à partir de $\mathcal{A}(\alpha,\beta)$ representé par (1.10) définit une fonction continue si et seulement si une des conditions suivantes est vérifiée :*

(i) $\quad \alpha+\beta=1$ *et pour tout état q de Q, $f_{\mathcal{A}_q}(0^\omega)=f_{\mathcal{A}_q}(1^\omega)$.*

(ii) $$\sum_{q\in Q}\lambda_q W(0,q)+\sum_{q\in Q}\mu_q W(1,q)=0. \tag{1.11}$$

pour des nombres réels λ_q et μ_q fixés de telle manière qu'au moins un d'entre eux soit non nul.
Dans le cas (i), les $W(i,q)$ sont arbitraires. Dans le cas (ii), $(W(0,q),W(1,q))$ appartient à un hyperplan fixé de $\mathbb{R}^{2n}$.

Preuve Soit un automate arbitraire $\mathcal{A}_{ext}$ de $\mathcal{A}(\alpha,\beta)$ représenté par les $2n$ composantes du vecteur $(W(0,q),W(1,q))$. Nous supposons d'abord que $\widehat{f_{\mathcal{A}_{ext}}}$ est continue. Il en est ainsi au point $\frac{1}{2}$, et donc nécessairement

$$f_{\mathcal{A}_{ext}}(01^\omega)=f_{\mathcal{A}_{ext}}(10^\omega). \tag{1.12}$$

Le membre gauche de (1.12) peut être écrit comme

$$f_{\mathcal{A}_{ext}}(01^\omega)=\alpha[\frac{1}{1-\beta}\sum_{q\in Q}W(1,q)f_{\mathcal{A}_q}(1^\omega)]+\sum_{q\in Q}W(0,q)f_{\mathcal{A}_q}(1^\omega).$$

Clairement, une formule similaire est valable pour le membre droit de (1.12). On obtient donc (1.11) avec :

$$\lambda_q=f_{\mathcal{A}_q}(1^\omega)-\frac{\beta}{1-\alpha}f_{\mathcal{A}_q}(0^\omega),$$
$$\mu_q=\frac{\alpha}{1-\beta}f_{\mathcal{A}_q}(1^\omega)-f_{\mathcal{A}_q}(0^\omega).$$

Notons que les λ_q et les μ_q peuvent être tous nuls. Ceci se produit si et seulement si $\alpha+\beta=1$ et $f_{\mathcal{A}_q}(0^\omega)=f_{\mathcal{A}_q}(1^\omega)$ (cas (i)). Le vecteur $(W(0,q),W(1,q))$ peut alors être choisi arbitrairement.
Supposons maintenant les λ_q et les μ_q ainsi fixés. Nous devons montrer que l'automate $\mathcal{A}_{ext}$ représenté par un vecteur $(W(0,q),W(1,q))$ satisfaisant (1.11) définit bien une fonction continue. Pour cela, considérons séparément trois cas :

Cas 1.1 *Continuité au point $\frac{1}{2}$.*

Par le choix des λ_q et des μ_q, $\mathcal{A}_{ext}$ satisfait (1.12). De plus, en utilisant des arguments similaires à ceux dans [11], ceci implique la continuité au point $\frac{1}{2}$. (Notons que nous avons à modifier légèrement les considérations de [11], puisque nous autorisons dans un automate à niveaux des poids négatifs dans les transitions connectantes).

Cas 1.2 *Continuité aux points ayant une représentation binaire finie,*

c'est-à-dire deux représentations $w01^\omega$ et $w10^\omega$ pour un mot w de Σ^*. Vérifier la continuité en ce point se réduit à vérifier la continuité de $\widehat{f_{\mathcal{A}_{ext}(w)}}$ au point $\frac{1}{2}$, où $\mathcal{A}_{ext}(w)$ est l'automate $\mathcal{A}_{ext}$ avec la distribution initiale $(1,0,\cdots,0)W_w$. Mais $\widehat{f_{\mathcal{A}_{ext}(w)}}$ est clairement continue au point $\frac{1}{2}$, puisque, par le cas 1.1, $\widehat{f_{\mathcal{A}_{ext}}}$ l'est et chaque $\widehat{f_{\mathcal{A}_q}}$ est continue (même dans tout l'intervalle) par nos hypothèses.

Cas 1.3 *Continuité aux points ayant seulement une représentation binaire infinie.*

De nouveau, comme dans [11], ceci est toujours vrai pour les automates à niveaux de notre type. □

Le théorème 1.5 mérite plusieurs remarques. Premièrement, il illustre très clairement, comme cela a déjà été noté dans [11], qu'un automate à niveaux définit très peu souvent une fonction continue. On peut le voir plus concrètement dans l'exemple suivant.

Exemple 1.4

Considérons l'automate à niveaux de la figure 1.5.

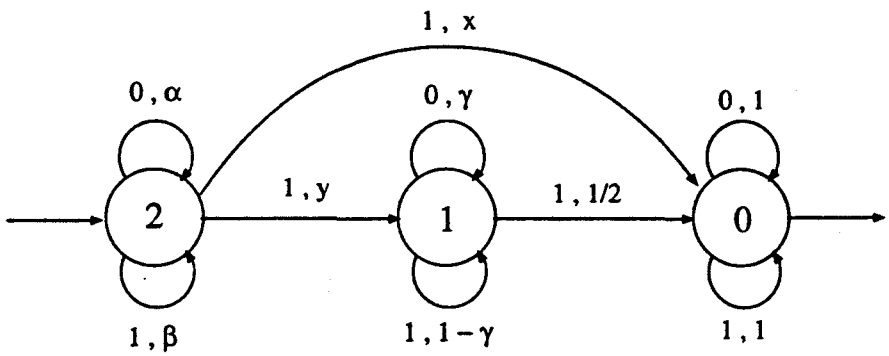


Figure 1.5 : Un automate à niveaux de degré 2

Puisque $\gamma + (1-\gamma) = 1$, l'automate $\mathcal{A}_1$ définit une fonction continue. Puisque $f_{\mathcal{A}_1}(0^\omega) = 0 \neq f_{\mathcal{A}_1}(1^\omega)$, par le théorème 1.5, il existe un unique hyperplan de $\mathbb{R}^2$, c'est-à-dire une ligne passant par l'origine, tel que $\mathcal{A}$ définisse une fonction continue si et seulement si (x,y) appartient à cette ligne. Cela signifie que le ratio x/y est unique. En particulier, si y est fixé, disons $y = \frac{1}{2}$, alors une seule valeur de x rend $\widehat{f_{\mathcal{A}}}$ continue. Pour $\gamma = \frac{1}{2}$, $\alpha = \beta = \frac{1}{4}$, cette valeur est $\frac{1}{4}$, et $\mathcal{A}$ réalise la fonction $f(x) = x^2$.

Ce qui précède conduit à l'observation intéressante suivante. Supposons que dans $\mathcal{A}$ les valeurs x et y rendant $\widehat{f_{\mathcal{A}}}$ continue soient toutes deux non nulles. Alors la décomposition la plus naturelle de $\mathcal{A}$ en deux sous-automates se fait en prenant pour $\mathcal{A}'_1$ le sous-automate excluant l'état 1 et pour $\mathcal{A}'_2$ le sous-automate excluant seulement la transition $2 \xrightarrow{1,x} 0$. Clairement, on a $\widehat{f_{\mathcal{A}}} = \widehat{f_{\mathcal{A}'_1}} + \widehat{f_{\mathcal{A}'_2}}$. Et bien que $\widehat{f_{\mathcal{A}}}$ soit continue, ni $\widehat{f_{\mathcal{A}'_1}}$ ni $\widehat{f_{\mathcal{A}'_2}}$ ne le sont. En particulier, si nous fixons les paramètres de $\mathcal{A}$ de telle manière qu'elle réalise la parabole, nous obtenons une décomposition de la parabole en la somme de deux fonctions qui sont toutes deux non continues. Un point intéressant ici est que l'automate donné pour la parabole est le

plus simple possible, et la décomposition est la seule naturelle du point de vue de la théorie des automates.

Notre seconde remarque est que le théorème 1.5 fournit une méthode simple systématique pour construire des automates de plus en plus compliqués réalisant des fonctions continues. Cette méthode est déjà illustrée dans l'exemple 1.4, et nous l'utilisons à nouveau dans la section 1.7.

Notre troisième remarque est que les conditions pour que les sous-automates $\mathcal{A}_q$ définissent des fonctions continues ne sont pas nécessaires, c'est-à-dire qu'un automate à niveaux ou même un automate linéaire peut définir une fonction continue pour la distribution initiale standard sans pouvoir être obtenu par une application récursive de la construction du théorème 1.5. On peut le voir dans l'exemple suivant :

Exemple 1.5

Considérons l'automate linéaire $\mathcal{A}$ montré dans la figure 1.6. D'après le critère décrit dans

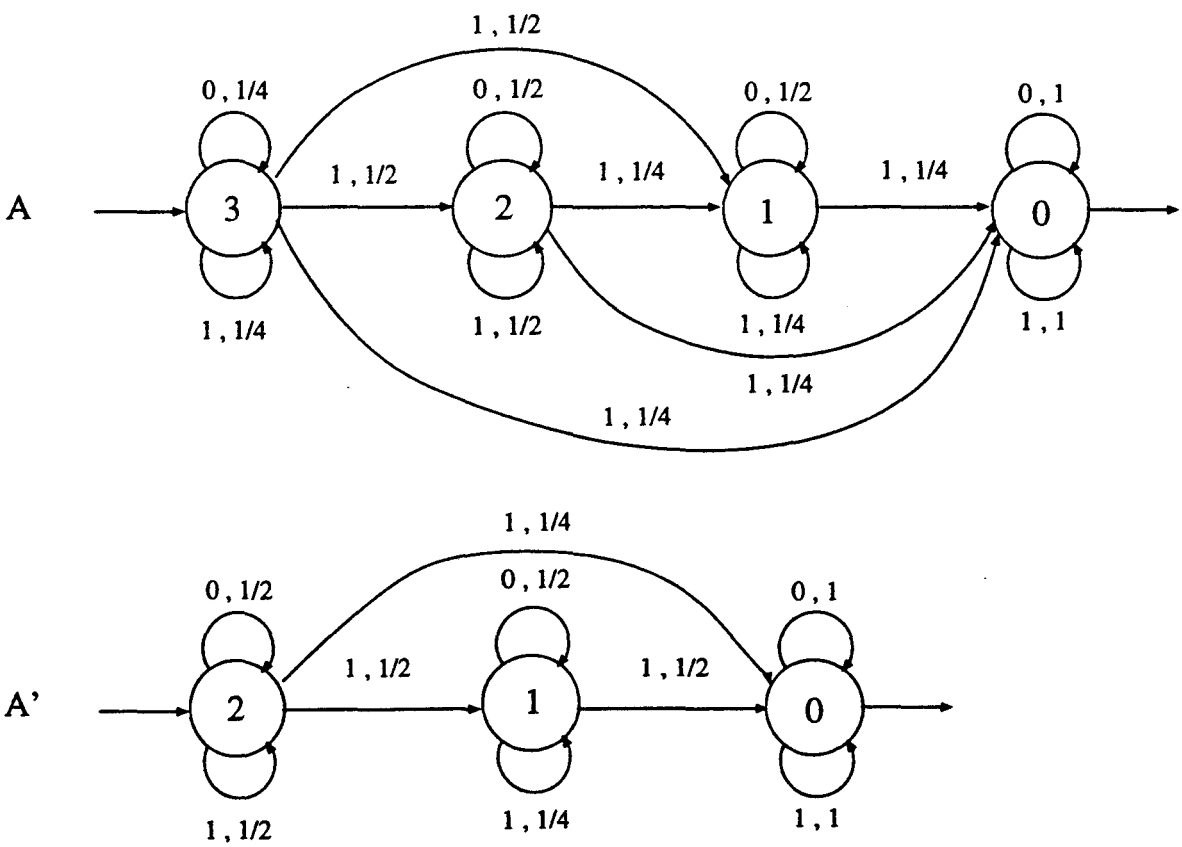


Figure 1.6 : Automates linéaires $\mathcal{A}$ et $\mathcal{A}'$

(1.9), le sous-automate $\mathcal{A}_1$ définit une fonction non continue. Il est facile de voir que pour tout mot w de Σ^*, si $P_{\mathcal{A}}(w) = (\alpha, \beta, \gamma, \delta)$ alors $\beta = \gamma$. On peut donc supprimer l'état 1 et les transitions concernant cet état, et remplacer $W_1(2,0)$ par $W_1(2,0) + W_1(1,0)$. Ce nouvel automate $\mathcal{A}'$ réalise la même fonction que $\mathcal{A}$ et il peut être obtenu par la construction de continuité. On peut vérifier que la fonction réalisée est x^2.

Nous allons maintenant prouver que la transformation précédente peut toujours être faite.

Un automate est dit *fortement continu* s'il réalise des fonctions continues pour n'importe quelle distribution initiale.

Théorème 1.6 *Soit $\mathcal{A}$ un automate à niveaux réalisant une fonction continue pour une distribution initiale I. Si $\mathcal{A}$ n'est pas fortement continu, on peut construire un automate à niveaux fortement continu ayant moins d'états que $\mathcal{A}$ et réalisant la même fonction.*

Preuve Nous allons raisonner par induction sur k, le nombre d'états de $\mathcal{A}$. Si $k = 1$, $\mathcal{A}$ est clairement fortement continu. Soit $Q = \{q_0, \cdots, q_n\}$ l'ensemble des états de $\mathcal{A}$ tel que q_0 soit le seul état de degré 0 et pour $0 \leq i \leq j \leq n$, le degré de q_i n'est pas plus grand que le degré de q_j. Considérons $D = \{P_{\mathcal{A}}(w)/w \in \Sigma^*\}$. Pour la distribution initiale $d = P_{\mathcal{A}}(w)$, $\mathcal{A}$ réalise une fonction continue, $\widehat{f_{\mathcal{A}}}(\frac{x}{2^{|w|}} + \hat{w})$. C'est encore vrai pour toute distribution initiale dans $E = \{\lambda_1\alpha_1 + \cdots + \lambda_p\alpha_p/\lambda_i \in I\!R, d_i \in D\}$, l'espace linéaire engendré par D. Si $E = I\!R^{n+1}$, nous obtenons le résultat. Sinon, il existe un vecteur colonne $\mu = (\mu_0, \cdots, \mu_n) \neq 0$, tel que pour tout d de E on a $\mu d = 0$.
Soit i_0 le plus petit entier tel que $\mu_{i_0} \neq 0$. On peut supposer que $\mu_{i_0} = 1$. Alors, pour tout $d = (d_0, \cdots, d_n)$ de E, $d_{i_0} = -(\mu_{i_0+1}d_{i_0+1} + \cdots + \mu_n d_n)$.
Si $i_0 = 0$, nous allons montrer que $f_{\mathcal{A}}$ est la fonction constante 0. En effet, puisque pour les états de degré > 0 les poids des boucles sont plus petits que 1, pour tout $\varepsilon > 0$, il existe un entier n tel que si $|w| > n, d = P_{\mathcal{A}}(w) = (d_0, d_1, \cdots, d_n)$ alors $|d_1|, \cdots, |d_n| < \varepsilon$. D'où $|d_0| \leq \varepsilon(\mu_1 + \cdots + \mu_n)$, de sorte que $f_{\mathcal{A}}$ est la fonction zéro.
Si $i_0 \neq 0$, nous allons supprimer l'état q_{i_0} et les transitions concernant q_{i_0}. Ensuite nous devons modifier les poids de certaines transitions connectantes de telle façon que cette automate $\mathcal{A}'$ réalise la même fonction.
Nous posons $Q' = Q - \{q_{i_0}\}$ et nous définissons la $Q' \times Q'$ matrice W'_a, pour $a = 0, 1$, par $W'_a(q_i, q_j) = W_a(q_i, q_j) - \mu_i W_a(q_{i_0}, q_j)$. Il est facile de vérifier que $\mathcal{A}'$ est un automate à niveaux de n états réalisant $f_{\mathcal{A}}$ et nous pouvons terminer notre construction par l'hypothèse d'induction. □

1.7 Un exemple : une fonction dérivable nulle part

Dans cette section, nous allons considérer un automate à niveaux $\mathcal{A}(t)$ montré dans la figure 1.7. Nous allons d'abord prouver que pour toute valeur de t et $x(t)$, $\mathcal{A}(t)$ définit une fonction continue. Ensuite, pour certaines valeurs particulières de t, nous regardons l'ensemble des points où $\mathcal{A}(t)$ possède une dérivée. En particulier, si $t = \frac{2}{3}$, nous obtenons un automate à niveaux réalisant une fonction continue n'ayant de dérivée en aucun point de l'intervalle $[0, 1]$.

Assertion 1.2 *$\mathcal{A}(t)$ est fortement continu.*

Preuve Soit $g : \Sigma^\omega \to I\!R$ la fonction réalisée par l'automate $\mathcal{A}(t)$ pour une distribution arbitraire $(\alpha, \beta, \beta', \gamma)$. Alors, $g = \alpha f_{\mathcal{A}(t)} + \beta f_{\mathcal{A}_1} + \beta' f_{\mathcal{A}_{1'}} + \gamma$ où $\mathcal{A}_1$ (resp. $\mathcal{A}_{1'}$) est le sous-automate contenant seulement les états 1 et 0 (resp. $1'$ et 0). D'après [11], $\mathcal{A}(t)$ est

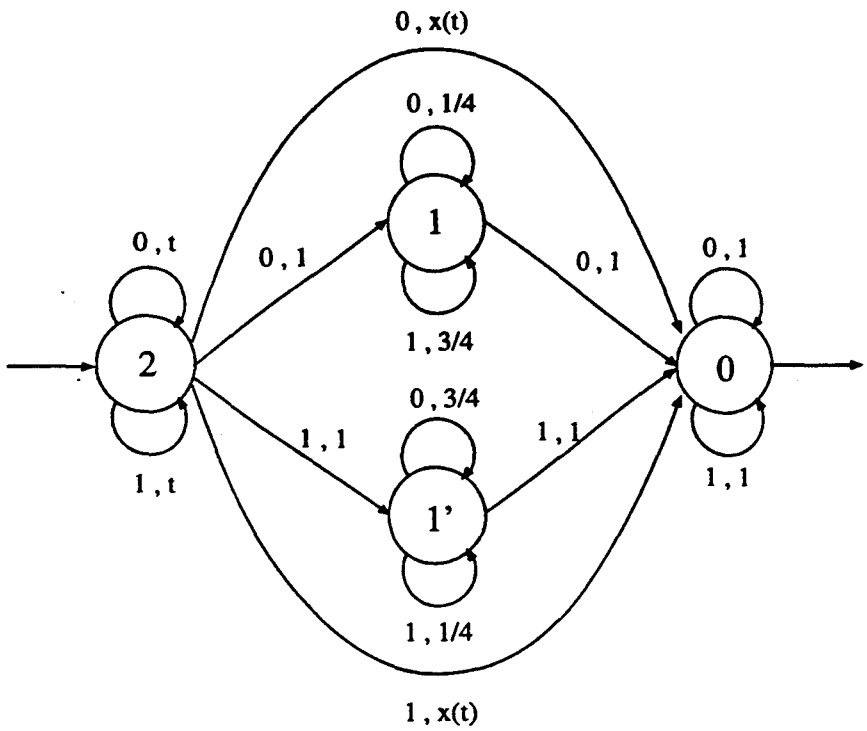


Figure 1.7 : Automate à niveaux $\mathcal{A}(t)$

fortement continue si et seulement si $g(10^\omega) = g(01^\omega)$. Cette condition est vérifiée puisque $f_{\mathcal{A}_1}(10^\omega) = f_{\mathcal{A}_1}(01^\omega)$, $f_{\mathcal{A}_{1'}}(10^\omega) = f_{\mathcal{A}_{1'}}(01^\omega)$ et $f_{\mathcal{A}(t)}(10^\omega) = f_{\mathcal{A}(t)}(01^\omega)$. En effet, $\mathcal{A}_1$ et $\mathcal{A}_{1'}$ sont fortement continus et l'égalité $f_{\mathcal{A}(t)}(10^\omega) = f_{\mathcal{A}(t)}(01^\omega)$ découle de la symétrie de $\mathcal{A}(t)$. □

Bien que $\mathcal{A}(t)$ soit fortement continu pour toute valeur de t, le sous-automate $\mathcal{A}'$ (resp. $\mathcal{A}''$) contenant seulement les états 2, 1 et 0 (resp. 2, 1' et 0) ne définit une fonction continue que pour une valeur particulière de $x(t)$. La condition pour $x(t)$ est déterminée par $f_{\mathcal{A}'(t)}(10^\omega) = f_{\mathcal{A}'(t)}(01^\omega)$ c'est-à-dire par

$$t[\frac{1}{1-t}x(t) + \frac{1}{1-t} \cdot 1 \cdot \frac{4}{3}] = x(t),$$

ou de façon équivalente,

$$(1-2t)x(t) = \frac{4}{3}t. \tag{1.13}$$

Exemple 1.6

Considérons $\mathcal{A}(t)$ pour $t = \frac{3}{4}$. L'égalité (1.13) donne $x(t) = -2$. Nous allons vérifier que pour tout mot fini w de Σ^*, la dérivée de la fonction $\widehat{f_{\mathcal{A}(\frac{3}{4})}}$ n'existe pas au point $\widehat{w0^\omega}$ et est nulle au point $\widehat{w(01)^\omega}$. Remarquons d'abord que pour toute distribution initiale $(y, 2y, 2y, z)$, $\mathcal{A}(\frac{3}{4})$ réalise une fonction constante. Ensuite, on a

$$\begin{aligned}(\alpha,\beta,\gamma,\delta)W_{0^n} &= (\alpha(\tfrac{3}{4})^n, 2\alpha(\tfrac{3}{4})^n - 2\alpha(\tfrac{1}{4})^n + \beta(\tfrac{1}{4})^n, \gamma(\tfrac{3}{4})^n, z) \\ &= (y, 2y, 2y, z) + (0, (\beta-2\alpha)(\tfrac{1}{4})^n, (\gamma-2\alpha)(\tfrac{3}{4})^n, 0)\end{aligned}$$

pour $y = \alpha(\frac{3}{4})^n$ et pour une certaine valeur de z. Par conséquent, si pour un mot w de Σ^*, la distribution $P_{\mathcal{A}(\frac{3}{4})}(w)$ est $(\alpha, \beta, \gamma, \delta)$, alors

$$f_{\mathcal{A}(\frac{3}{4})}(w0^n0^\omega) - f_{\mathcal{A}(\frac{3}{4})}(w0^n1^\omega) = ((\frac{1}{4})^n\beta - (\frac{1}{4})^n 2\alpha) \cdot \frac{4}{3} - ((\frac{3}{4})^n\gamma - (\frac{3}{4})^n 2\alpha) \cdot \frac{4}{3}.$$

Il découle directement par induction que $\gamma < 2\alpha$, de telle sorte que la limite

$$\lim_{n\to\infty} \frac{|f_{\mathcal{A}(\frac{3}{4})}(w0^n0^\omega) - f_{\mathcal{A}(\frac{3}{4})}(w0^n1^\omega)|}{|\widehat{w0^n0^\omega} - \widehat{w0^n1^\omega}|} = \lim_{n\to\infty} \frac{\frac{4}{3}(2\alpha - \gamma)}{(\frac{1}{2})^{|w|}} \cdot \frac{(\frac{3}{4})^n}{(\frac{1}{2})^n} = \infty,$$

prouvant que $f'_{\mathcal{A}(\frac{3}{4})}(\widehat{w0^\omega})$ n'existe pas.

Similairement,

$$\begin{aligned}
& (\alpha, \beta, \gamma, \delta)W_{(01)^n} \\
= \; & (\alpha(\frac{3}{4})^{2n}, 2\alpha(\frac{3}{4})^{2n}(1 - (\frac{1}{3})^n) + \beta(\frac{3}{16})^n, 2\alpha(\frac{3}{4})^{2n}(1 - (\frac{1}{3})^n) + \gamma(\frac{3}{16})^n, y) \\
= \; & (y, 2y, 2y, z) + (0, \beta', \gamma', 0) \\
& \text{pour une certaine valeur } z \text{ et pour } y = \alpha(\frac{3}{4})^{2n}, \\
& \beta' = \beta(\frac{3}{16})^n - 2\alpha(\frac{3}{4})^{2n}(\frac{1}{3})^n = (\beta - 2\alpha)(\frac{3}{16})^n \\
& \text{et } \gamma' = (\gamma - 2\alpha)(\frac{3}{16})^n.
\end{aligned}$$

La fonction réalisée par l'automate $\mathcal{A}(\frac{3}{4})$ pour la distribution initiale $(0, \beta', \gamma', 0)$ est bornée par $4|\beta' + \gamma'| \leq 4(4\alpha + \beta + \gamma)(\frac{3}{16})^n$. Par conséquent, si, pour un mot w de Σ^*, on a $P_{\mathcal{A}(\frac{3}{4})}(w) = (\alpha, \beta, \gamma, \delta)$ alors pour tout mot w' de Σ^ω, on a

$$|f_{\mathcal{A}(\frac{3}{4})}(w(01)^n(01)^\omega) - f_{\mathcal{A}(\frac{3}{4})}(w(01)^n w')| \leq 2 \cdot 4 \cdot (4\alpha + \beta + \gamma)(\frac{3}{16})^n.$$

Puisque pour $w' \in 1\Sigma^\omega \cup 00\Sigma^\omega$, on a

$$|w(01)^n(01)^\omega - w(01)^n w'| \geq \frac{1}{12} \cdot (\frac{1}{2})^{|w|+2n},$$

nous voyons que

$$\widehat{f_{\mathcal{A}(\frac{3}{4})}}'(w(01)^\omega) = 0.$$

Le graphe de la fonction $\widehat{f_{\mathcal{A}(\frac{3}{4})}}$ avec $x = -2$ est montré dans la figure 5.

Exemple 1.7

Pour $t = \frac{1}{4}$, l'égalité (1.13) donne $x(t) = \frac{2}{3}$ et on peut tirer des conclusions similaires à celles de l'exemple 1.6. En effet, le graphe de $\widehat{f_{\mathcal{A}(\frac{1}{4})}}$ montré dans la figure 6 est en un certain sens dual de celui de $\widehat{f_{\mathcal{A}(\frac{3}{4})}}$.

Exemple 1.8

Pour $t = \frac{2}{3}$, l'égalité (1.13) donne $x(t) = -\frac{8}{3}$. le graphe de la fonction $\widehat{f_{\mathcal{A}(\frac{2}{3})}}$ est montré dans la figure 7.
Le comportement de $\widehat{f_{\mathcal{A}(\frac{2}{3})}}$ est plus compliqué que ceux des fonctions $\widehat{f_{\mathcal{A}(\frac{3}{4})}}$ et $\widehat{f_{\mathcal{A}(\frac{1}{4})}}$. Nous allons montrer que, bien que, par construction, $\widehat{f_{\mathcal{A}(\frac{2}{3})}}$ est continue sur tout l'intervalle $[0,1]$, elle n'a de dérivée en aucun point de cet intervalle. Pour cela, fixons un point dans $[0,1]$, et supposons qu'il a une représentation binaire w. Notons w_n le préfixe de w de longueur n. Nous considérons les mots infinis suivants :

$$\begin{aligned} w_0(n) &= w_n 0^\omega, \\ w_1(n) &= w_n 10^\omega, \\ w_2(n) &= w_n 110^\omega, \\ w_3(n) &= w_n 1^\omega. \end{aligned}$$

Alors, clairement,

$$|\hat{w} - \widehat{w_i(n)}| \leq \frac{1}{2^n} \qquad \text{pour } i = 0,1,2,3. \tag{1.14}$$

Considérons la distribution donnée par w_n sur $\mathcal{A}(\frac{2}{3})$, disons

$$P_{\mathcal{A}(\frac{2}{3})}(w_n) = (\alpha_n, \beta_n, \gamma_n, \delta_n).$$

Ceci permet de calculer les valeurs $f_{\mathcal{A}(\frac{2}{3})}(w_i(n))$. En effet,

$$\begin{aligned} f_{\mathcal{A}(\frac{2}{3})}(w_0(n)) &= \alpha_n[3 \cdot (\frac{4}{3} - \frac{8}{3})] + \frac{4}{3}\beta_n + \delta_n \\ &= -4\alpha_n + \frac{4}{3}\beta_n + \delta_n, \end{aligned}$$

et similairement,

$$\begin{aligned} f_{\mathcal{A}(\frac{2}{3})}(w_1(n)) &= -\frac{16}{3}\alpha_n + \beta_n + \gamma_n + \delta_n, \\ f_{\mathcal{A}(\frac{2}{3})}(w_2(n)) &= -\frac{47}{9}\alpha_n + \frac{3}{4}\beta_n + \frac{5}{4}\gamma_n + \delta_n, \\ f_{\mathcal{A}(\frac{2}{3})}(w_3(n)) &= -4\alpha_n + \frac{4}{3}\gamma_n + \delta_n. \end{aligned}$$

Ici la valeur de α_n est facile à déterminer :

$$\alpha_n = (\frac{2}{3})^n,$$

indépendamment de w.

Considérons maintenant les deux différences

$$f_{\mathcal{A}(\frac{2}{3})}(w_0(n)) - f_{\mathcal{A}(\frac{2}{3})}(w_3(n)) = \frac{4}{3}(\beta_n - \gamma_n),$$

et

$$f_{\mathcal{A}(\frac{2}{3})}(w_1(n)) - f_{\mathcal{A}(\frac{2}{3})}(w_2(n)) = -\frac{1}{9}\alpha_n + \frac{1}{4}(\beta_n - \gamma_n).$$

Il s'ensuit que pour chaque entier n, la valeur absolue d'au moins une de ces différences est au moins $\frac{1}{18}\alpha_n$. Par là même, pour chaque entier n, il existe un indice i_n de $\{0,1,2,3\}$ tel que

$$|f_{\mathcal{A}(\frac{2}{3})}(w) - f_{\mathcal{A}(\frac{2}{3})}(w_{i_n}(n))| \geq \frac{1}{36}\alpha_n. \tag{1.15}$$

Puisque i_n prend ses valeurs sur un ensemble fini, (1.15) est en fait vraie pour un i_0 fixé et pour une infinité de valeurs de n. C'est-à-dire qu'il existe un sous-ensemble infini I de $\mathbb{N}$ et une valeur i_0 tels que

$$|f_{\mathcal{A}(\frac{2}{3})}(w) - f_{\mathcal{A}(\frac{2}{3})}(w_{i_0}(n))| \geq \frac{1}{36}(\frac{2}{3})^n \qquad \text{pour } n \in I.$$

En combinant ceci avec (1.14), on obtient

$$\frac{|f_{\mathcal{A}(\frac{2}{3})}(w) - f_{\mathcal{A}(\frac{2}{3})}(w_{i_0}(n))|}{|\hat{w} - \widehat{w_{i_0}(n)}|} \geq \frac{1}{36}(\frac{4}{3})^n \qquad \text{pour } n \in I.$$

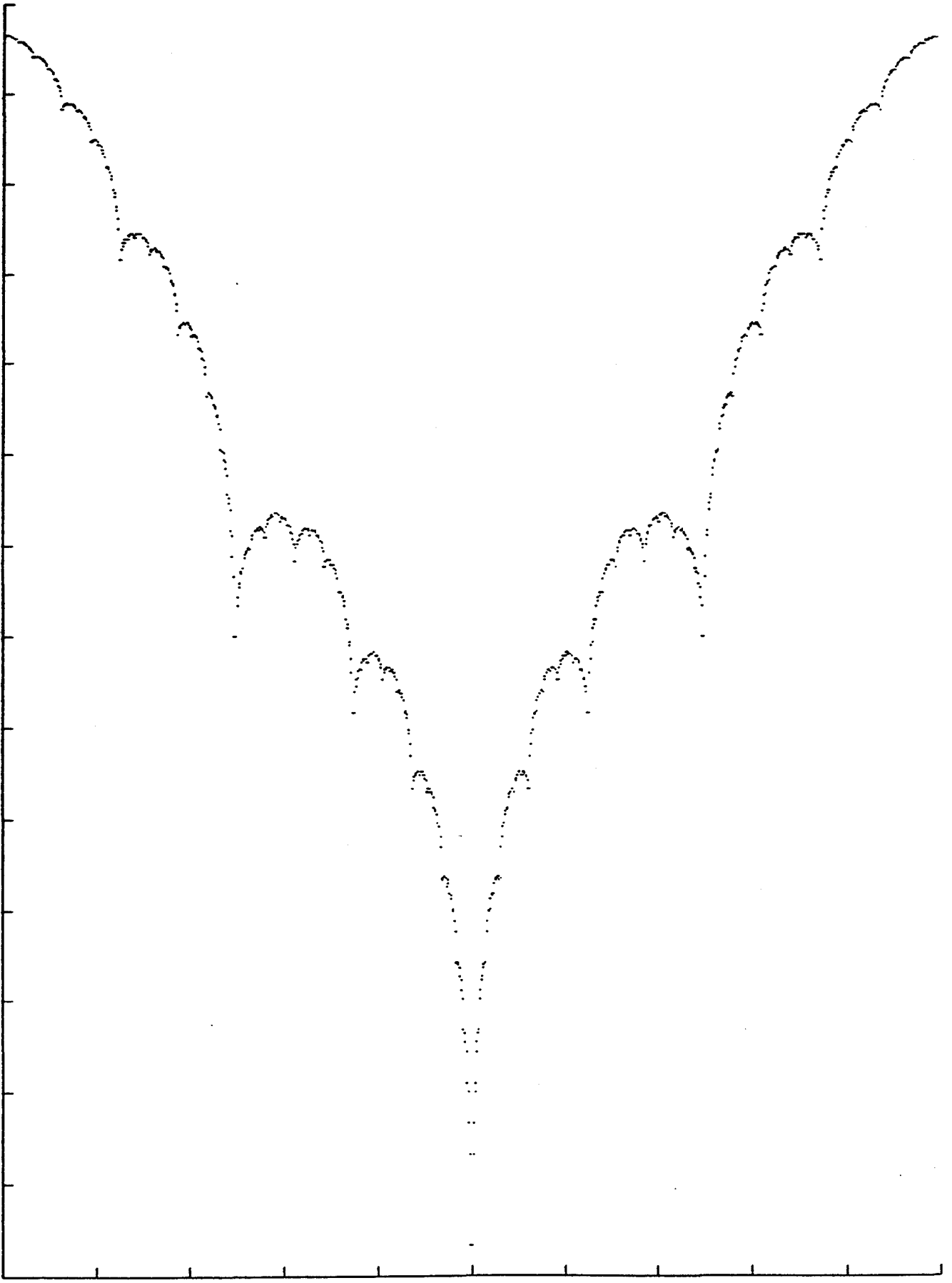
Puisque I est infini, cela prouve que $\widehat{f_{\mathcal{A}(\frac{2}{3})}}$ ne possède aucune dérivée (ni même une dérivée finie à droite ou à gauche) au point présenté par w.

Conclusion

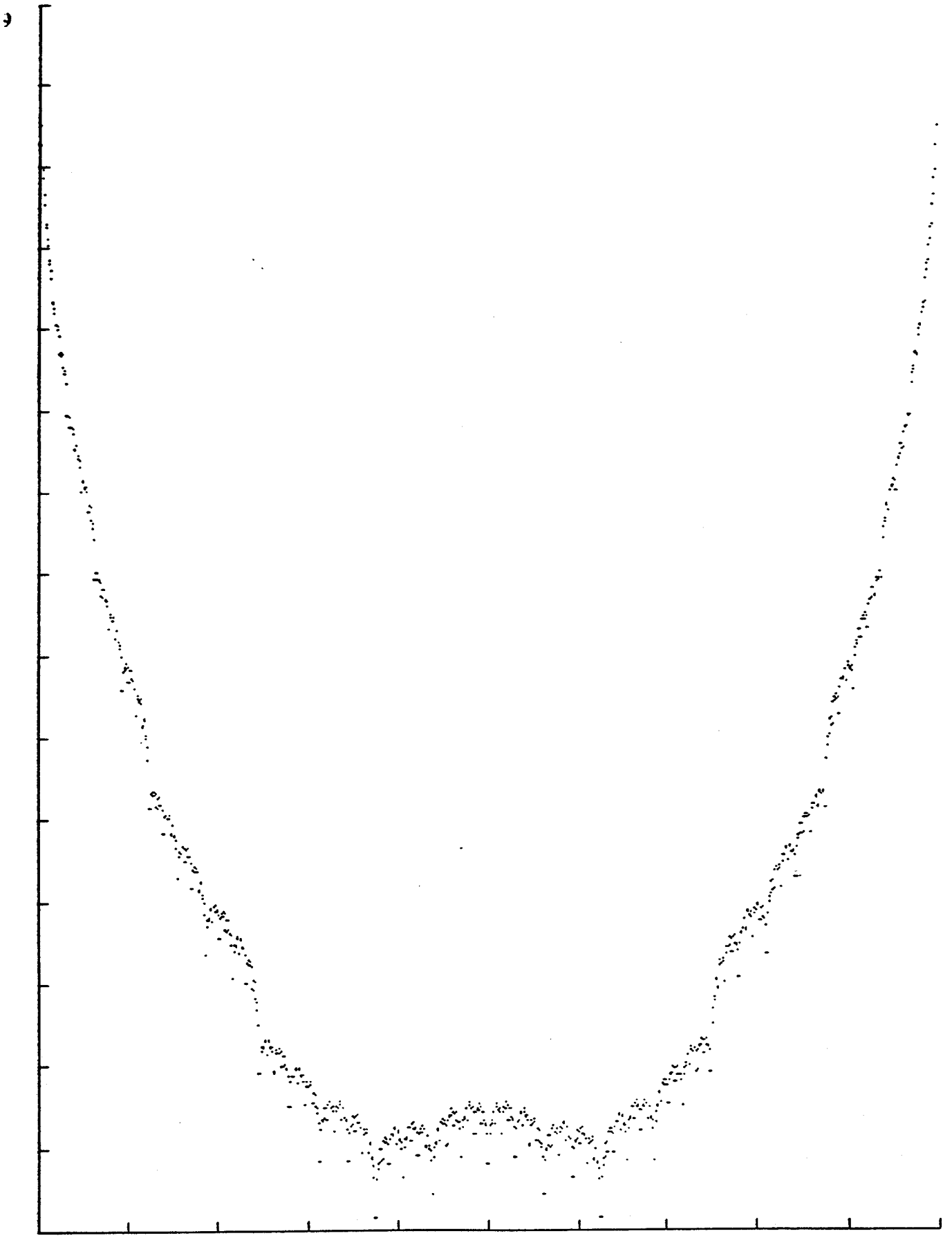
L'étude du type de croissance qui peut ou ne peut pas être réalisé par un automate fini pondéré montre que peu de fonctions ayant un bon comportement au sens traditionnel du terme peuvent êtres réalisées par de tels automates. En fait, la plupart des fonctions qui peuvent être réalisées sont de type fractal. Ceci est dû à la structure des automates, qui est beaucoup mieux adaptée pour exprimer la notion de récurrence que celle de continuité. Les exemples 1.6 à 1.8 montrent qu'un petit changement dans les poids d'un automate change radicalement le comportement de la fonction qu'il réalise. L'exemple 1.8 montre que même une fonction dont le comportement peut paraître aléatoire, peut être réalisée avec un automate très simple, ici, un automate ne contenant que quatre états, c'est-à-dire autant d'états qu'un automate définissant un pôlynome de degré trois. Ceci implique que calculer les valeurs d'une fonction au comportement complexe (ou leur approximation avec une précision donnée) n'est parfois pas plus compliqué que calculer les valeurs d'un polynôme de degré trois.

Certaines questions naturelles restent ouvertes. On peut en particulier se demander si le corollaire 1.2 et le théorème 1.4 peuvent être étendus aux automates finis pondérés généraux. De même, le problème de décider si un automate fini pondéré donné définit une fonction continue reste ouvert.

0.0 Figure 5 : t = 3/4 ; x(t) = -2 1.0



0 Figure 6 : t = 1/4 ; x(t) = 2/3 1.0



0.0 Figure 7 ; t = 2/3 ; x(t) = -8/3 1.0

Chapitre 2

Composition de morphismes injectifs et de morphismes injectifs inverses

De nombreux travaux tendent à définir des opérations à l'aide d'autres opérations plus simples. Une caractérisation fondamentale d'une transduction rationnelle est son expression en termes d'intersections avec un langage rationnel, de morphismes et de morphismes inverses [29]. D'autres caractérisations ont été trouvées, utilisant uniquement un marquage de fin suivi par une composition de morphismes et de morphismes inverses [24, 36]. Dans [21, 22], un résultat similaire est établi pour les fonctions rationnelles (resp. fonctions sous-séquentielles gauches). Dans ces compositions, le morphisme inverse est l'inverse d'un morphisme injectif (resp. préfixe).

Certains de ces résultats ont été établis grâce aux études des transducteurs simples, dans lesquels l'état initial est l'unique état final. Les transductions réalisées par ces transducteurs sont des compositions faites uniquement de morphismes et de morphismes inverses. Ces compositions peuvent être supposées de longueur quatre [27, 28, 37].

Dans ce chapitre, nous restreignons notre étude à la composition de morphismes injectifs (préfixes) et morphismes injectifs (préfixes) inverses. L'ensemble des morphismes injectifs est la famille des morphismes qui donne, par composition de ses éléments ou de leur inverse, des transductions fonctionnelles. Dans le cas des morphismes injectifs, les compositions peuvent être supposées de longueur quatre : toute composition de morphismes injectifs et de morphismes injectifs inverses est réalisée par un morphisme injectif, suivi d'un morphisme injectif inverse, un morphisme injectif et un morphime injectif inverse. Par contre, dans le cas des compositions de morphismes préfixes et de morphismes préfixes inverses, les compositions de longueur quatre ne représentent pas la totalité de la famille de ces compositions. La longueur cinq est nécessaire et la longueur six est suffisante (en commençant avec un morphisme préfixe).

2.1 Définitions et notations

Rappelons qu'un *transducteur* T est un sextuple $(Q, \Sigma, \Delta, \delta, S, F)$ où Q est un ensemble fini d'états, Σ l'alphabet d'entrée, Δ l'alphabet de sortie, δ l'ensemble fini des transitions inclus dans $Q \times \Sigma^* \times \Delta^* \times Q$, $S \subseteq Q$ l'ensemble des états initiaux et $F \subseteq Q$ l'ensemble des états terminaux.

Nous appelerons I_T (resp. W_T) le morphisme de δ^* dans Σ^* (resp. de δ^* dans Δ^*) associant à chaque chemin du transducteur T le mot lu (resp. écrit) par ce chemin. Les morphismes I_T et W_T sont donc définis sur les transitions de T par $(q, x, y, p)I_T = x$ et $(q, x, y, p)W_T = y$. Par convention, il existe pour tout état q un chemin vide allant de q vers lui-même. L'ensemble de tous les chemins (resp. chemins réussis) de T forme un ensemble régulier de δ^*.

Soit T un transducteur défini comme ci-dessus. On dit que T *réalise* la *transduction rationnelle* τ de Σ^* dans Δ^* définie par son graphe $\hat{\tau} = \{(\pi I_T, \pi W_T)/\pi \in A_T\}$ (cf. [19, 29]).

On note $x\tau$ l'ensemble $\{y/(x, y) \in \hat{\tau}\}$.

Le transducteur T est dit *non ambigu* si pour tout mot x de Σ^*, il existe au plus un chemin réussi de T lisant x.

Le transducteur T réalise une *fonction rationnelle* τ si τ est une fonction partielle de Σ^* dans Δ^*, c'est-à-dire, si tout mot x de Σ^* a pour image par τ un langage de Δ^* contenant au plus un mot.

Clairement, tout transducteur non ambigu réalise une fonction rationnelle. Réciproquement, toute fonction rationnelle peut être réalisée par un transducteur non ambigu.

Un *a-gsm gauche* T est un transducteur possédant un unique état initial, dont chaque transition lit une lettre de Σ (et écrit un mot de Δ^*), et tel que pour tout état q_i de Q et toute lettre x de Σ, il existe au plus une transition partant de q_i et lisant x. Un a-gsm gauche réalise une fonction rationnelle déterministe.

La fonction de transition d'un a-gsm gauche sera notée par un point. Ainsi l'égalité $q_i \cdot x = q_j$ signifie que q_j est l'(unique) état atteint à partir de q_i en lisant x.

Une transduction rationnelle τ est dite *sous-séquentielle gauche*, s'il existe un a-gsm gauche T_1 réalisant τ_1 et une fonction partielle ρ de F dans Δ^* telle que pour tout mot x de Σ^*, $x\tau = x\tau_1 \ (q_0 \cdot x)\rho$. Un transducteur sous-séquentiel gauche T est un 6-uple $(Q, \Sigma, \Delta, \delta, q_0, \rho)$ où $(Q, \Sigma, \Delta, \delta, q_0, Dom(\rho))$ est un a-gsm gauche et ρ une fonction partielle de Q dans Δ^*.

Une transduction τ de Σ^* dans Δ^* est *fidèle* si, pour tout mot y de Δ^*, $y\tau^{-1}$ est un langage fini de Σ^*.

Un transducteur T est *simple* si l'unique état initial est aussi l'unique état final. Une transduction est *simple* si elle peut être réalisée par un tranducteur simple.

Un morphisme $\alpha : \Sigma^* \to \Delta^*$ est dit *non effaçant* (resp. *alphabétique*, *strictement alphabetique*), si l'image par α de toute lettre de Σ est un mot de Δ^* de longueur supérieure ou égale à 1 (resp. inférieure ou égale à 1, exactement égale à 1).

Un morphisme $\alpha : \Sigma^* \to \Delta^*$ est dit *préfixe* (resp. *suffixe*), si l'image de l'alphabet Σ par α

est un langage préfixe (resp. suffixe).

De plus, si Σ et Δ sont deux alphabet disjoints, alors un morphisme $\alpha : \Sigma^* \to (\Sigma \cup \Delta)^*$ est *uniforme* (resp. *uniforme à gauche*), s'il existe un mot u de Δ^* avec $a\alpha = au$ (resp. $a\alpha = ua$) pour toute lettre a de Σ.

Soit H (resp. H_a, H_{sa}, H_{ne}, H_u, H_{lu}, H_i, H_p, H_s) la famille des morphismes (resp. morphismes alphabétiques, strictement alphabétiques, non effaçants, uniformes, uniformes à gauche, injectifs, préfixes, suffixes). Pour les ensembles de morphismes ci-dessus, on note H_x^{-1} l'ensemble des inverses des morphismes de H_x.

Un *marquage à droite* μ_m est une application qui ajoute un symbole spécial m à la fin de chaque mot, c'est-à-dire, μ_m de Σ^* dans $(\Sigma \cup \{m\})^*$ est défini par $x\mu_m = xm$, pour tout mot x de Σ^*. On note M_r la famille des marquages à droite.

Il est souvent possible d'établir une relation entre les mots d'entrée et les chemins d'un transducteur T en utilisant la composition d'un morphisme uniforme h_u et d'un morphisme inverse h^{-1}. Le morphisme uniforme ajoute à chaque lettre une représentation des états. Les morphismes peuvent être soit

$$\begin{aligned} xh_u &= xq_Nq_{N-1}\cdots q_1q_0 \\ (q_i, x, v, q_j)h &= q_{i-1}\cdots q_1q_0xq_Nq_{N-1}\cdots q_j \end{aligned}$$

soit

$$\begin{aligned} xh_u &= x\alpha^{2^N} \\ (q_i, x, v, q_j)h &= \alpha^{2^i-1}x\alpha^{2^N-2^j+1} \end{aligned}$$

Nous utiliserons dans les preuves la première méthode.

Dans toutes les preuves, nous considèrerons que les transducteurs sont choisis émondés (ne comportant que des états accessibles et coaccessibles).

2.2 Résultats précédents

Rappelons les récents résultats sur ce sujet. Nous portons une attention particulière aux résultats concernant les transductions simples.

Clairement, les compositions de marquages, de morphismes et de morphismes inverses sont des transductions rationnelles. La réciproque est également vraie, cf. [24, 36].

Pour les transductions rationnelles, ce qui suit a été prouvé dans [27, 28, 37].

Théorème 2.1 *L'ensemble des transductions rationnelles est égal à*

$$M_rH^{-1}HH^{-1}H = (M_r + H + H^{-1})^* = M_rHH^{-1}HH^{-1}.$$

L'ensemble des transductions rationnelles simple est égal à

$$(H + H^{-1})^* = H^{-1}HH^{-1}H = HH^{-1}HH^{-1}.$$

Les résultats suivants ont été prouvés pour des fonctions rationnelles dans [21, 22].

Théorème 2.2 *L'ensemble des fonctions rationnelles est égal à*

$$M_r H_i H_i^{-1} H = (M_r + H + H_i^{-1})^*.$$

L'ensemble des fonctions sous-séquentielles gauches est égal à

$$M_r H_p H_p^{-1} H = (M_r + H + H_p^{-1})^*.$$

Le résultat a été amélioré dans [23].

Théorème 2.3 *L'ensemble des fonctions rationnelles est égal à*

$$M_r H_u H_p^{-1} H_s^{-1} H_a.$$

L'ensemble des fonctions sous-séquentielles gauches est égal à

$$M_r H_u H_p^{-1} H_a.$$

L'étude des compositions de morphismes est liée à celle des transductions simples. Concernant les transductions simples, les résultats suivants ont été prouvés dans [21, 22].

Théorème 2.4 *L'ensemble des transductions simples non ambigües est égal à*

$$U_* = H_i H_i^{-1} H = (H + H_i^{-1})^*.$$

L'ensemble des transductions déterministes simples est égal à

$$D_* = H_p H_p^{-1} H = (H + H_p^{-1})^*.$$

2.3 Compositions de morphismes injectifs et de morphismes injectifs inverses

Nous allons caractériser l'ensemble des bijections rationnelles simples entre monoïdes libres comme étant égal aux compositions de morphismes injectifs et de morphismes injectifs inverses. Cet ensemble est également celui des transductions rationnelles injectives simples non ambigües.

En effet, le domaine et le codomaine d'une transduction rationnelle injective simple non ambigüe sont tous deux des monoïdes libres. Puisque la transduction τ appartient à U_*, le domaine est un monoïde libre [21]. Il a une base unique qui est un code C. Le codomaine est $C^*\tau$. Puisque la transduction est une fonction simple, le codomaine est aussi égal à $(C\tau)^*$. Puisque la transduction est injective et que C est un code, $(C\tau)^*$ est un monoïde libre.

Par conséquent, une transduction rationnelle injective simple non ambigüe est une bijection entre monoïdes libres. Réciproquement, une bijection simple entre monoïdes libres est une transduction rationnelle injective simple non ambigüe.

Rappelons d'autre part que

- Il existe des bijections rationnelles simples pour lesquelles le domaine et le codomaine ne sont pas des monoïdes libres, comme par exemple la transduction réalisant l'intersection avec le langage $(a + ab + ba)^*$.

- Il existe des transductions rationnelles simples non ambigües pour lesquelles le codomaine n'est pas un monoïde libre, comme par exemple le morphisme défini par $xh = a, yh = ab, zh = ba$.

- Il existe des bijections rationnelles pour lesquelles le domaine et le codomaine sont des monoïdes libres, mais qui ne sont pas simples, comme par exemple la transduction définie par $\varepsilon\tau = \varepsilon, a\tau = a^2, a^2\tau = a$, et pour $i > 2$, $a^i\tau = a^i$.

Le premier lemme montre une construction qui apparait souvent dans ce chapitre. Cette construction a été utilisée entre autres dans [27] et [21].

Lemme 2.1 *Soit T un transducteur simple non ambigu. Il existe une bijection rationnelle simple appartenant à $H_u H_i^{-1}$ entre les mots d'entrée et les chemins correspondants.*

Preuve Soit $T = (Q, X, Y, \delta, \{q_0\}, \{q_0\})$ un transducteur simple non ambigu. Le transducteur peut être choisi tel que chaque transition lit une lettre, c'est-à-dire tel que δ est inclus dans $Q \times X \times Y^* \times Q$ (cf. [21]). Donc aucune transition ne lit le mot vide. On peut également supposer le transducteur émondé.

Soit h_u le morphisme uniforme défini sur X par $xh_u = xq_N q_{N-1} \dots q_1 q_0$ où $N = \|Q\| - 1$.

Soit h_1 le morphisme défini sur δ par $(q_i, x, v, q_j)h_1 = q_{i-1} \dots q_0 x q_N \dots q_j$.

Puisque le transducteur est non ambigu, pour chaque mot dans le domaine, il existe un unique chemin réussi. Puisque le transducteur est simple, chaque chemin réussi commence et termine dans l'état q_0.

Si un mot $wh_u = w_1 q_N \dots q_1 q_0 w_2 q_N \dots q_1 q_0 w_3 \dots\dots w_{|w|} q_N \dots q_1 q_0$ appartient au domaine de h_1^{-1}, son image $(wh_u)h_1^{-1}$ correspond au chemin réussi de w dans le transducteur T. Réciproquement, si un mot w appartient au domaine du transducteur, le chemin réussi π_w du mot w a une image par h_1 qui appartient à $X^* h_u$.

Soit $q_{i_0} .. q_0 x_1 .... x_n q_N .. q_{i_n}$ appartenant à $(\delta h_1)^*$. Si deux lettres x_j, x_{j+1} ne sont pas séparées par $q_N \dots q_1 q_0$, toute factorisation a une seule possibilité entre x_j et x_{j+1}. Si chaque x_j, x_{j+1} sont séparées par $q_N \dots q_1 q_0$, la non ambiguïté implique l'existence d'une factorisation unique. Le morphisme h_1 est donc injectif. □

Le même raisonnement reste valable quand T est un transducteur simple déterministe. On a donc :

Lemme 2.2 *Soit T un tranducteur simple déterministe. Il existe une bijection rationnelle simple appartenant à $H_u H_p^{-1}$ entre les mots d'entrée et les chemins réussis correspondants.*

On peut maintenant prouver la proposition suivante :

Proposition 2.1 *L'ensemble des bijections rationnelles simples entre monoïdes libres est égal à* $(H_i + H_i^{-1})^* = H_u H_i^{-1} H_i H_u^{-1}$.

Preuve Nous prouvons d'abord que l'ensemble des bijections rationnelles simples entre monoïdes libres est inclus dans $H_u H_i^{-1} H_i H_u^{-1}$.

Soit τ une bijection rationnelle simple entre un monoïde inclus dans X^* et un autre inclus dans Y^*. Puisque la transduction τ est une fonction simple ayant pour domaine un monoïde libre, elle appartient à U_* (cf. [21]). Soit $T = (Q, X, Y, \delta, \{q_0\}, \{q_0\})$ un transducteur simple non ambigu réalisant τ. Le transducteur peut être choisi tel que δ est inclus dans $Q \times X \times Y^* \times Q$ (cf. [21]). Il est également supposé émondé. Puisque τ est une bijection simple, $\varepsilon\tau^{-1} = \varepsilon$. Chaque chemin réussi non vide est donc non effaçant. On peut construire un transducteur simple non ambigu $T_2 = (Q_2, X, Y, \delta_2, \{(q_0, \varepsilon)\}, \{(q_0, \varepsilon)\})$ tel que chaque transition qui mène dans l'état (q_0, ε) est non effaçante. Ceci est fait par la construction suivante :

$$
\begin{array}{rll}
Q_2 & = Q_1 \times (\{\varepsilon\} \cup Y), & \\
((q_i, \varepsilon), x, \varepsilon, (q_j, \varepsilon)) & \in \delta_2 \text{ quand } (q_i, x, \varepsilon, q_j) & \in \delta_1, \\
((q_i, y), x, \varepsilon, (q_j, y)) & \in \delta_2 \text{ quand } (q_i, x, \varepsilon, q_j) & \in \delta_1 \text{ et } q_j \neq q_0, \\
((q_i, \varepsilon), x, v, (q_j, y)) & \in \delta_2 \text{ quand } (q_i, x, vy, q_j) & \in \delta_1 \text{ et } q_j \neq q_0, \\
((q_i, y_1), x, y_1 v, (q_j, y_2)) & \in \delta_2 \text{ quand } (q_i, x, vy_2, q_j) & \in \delta_1 \text{ et } q_j \neq q_0, \\
((q_i, y), x, yv, (q_0, \varepsilon)) & \in \delta_2 \text{ quand } (q_i, x, v, q_0) & \in \delta_1,
\end{array}
$$

Dans cette définition, on suppose que v est un mot (éventuellement vide) de Y^*, et que y, y_1 et y_2 sont des mots non vides de Y^*.

Supposons donc que $T = (Q, X, Y, \delta, \{q_0\}, \{q_0\})$ possède cette propriété.

Nous allons établir deux bijections. Le lemme 2.1 montre qu'il en existe une première entre les mots d'entrée et les chemins réussis correspondants dans le transducteur T. La seconde sera une bijection entre les mots de sortie et les chemins réussis correspondants dans le transducteur T.

Soit g_u le morphisme uniforme défini sur Y par $yg_u = yq_N \ldots q_1 q_0$ où $N = \|Q\| - 1$.

Puisque la transduction τ est une bijection, elle est fidèle. Tout chemin plus long que $\|Q\|$ est donc non effaçant. Tout chemin non effaçant est donc une succession $c_1 c_2 \cdots c_n$ de chemins consécutifs où chaque c_i est une succession d'au plus $\|Q\|$ transitions consécutives dont seule la dernière est non effaçante. On représente l'ensemble des chemins c_i par l'alphabet Γ suivant :

$\Gamma = \{[q_{i_1}, x_1 \ldots x_m, y_1 \ldots y_n, q_{i_{m+1}}] / \ 1 \leq m \leq \|Q\|, (q_{i_1}, x_1, \varepsilon, q_{i_2}) \ldots (q_{i_{m-1}}, x_{m-1}, \varepsilon, q_{i_m})$ est un chemin effaçant de T menant de q_{i_1} à q_{i_m} et $(q_{i_m}, x_m, y_1 y_2 \ldots y_n, q_{i_{m+1}})$ est une transition non effaçante de T menant de q_{i_m} à $q_{i_{m+1}}$ $\}$.

Soit g_1 le morphisme défini pour chaque lettre $[q_i, x_1 \ldots x_m, y_1 \ldots y_n, q_j]$ de Γ par

$$[q_i, x_1 \ldots x_m, y_1 \ldots y_n, q_j] g_1 = q_{i-1} \ldots q_0 y_1 q_N \ldots q_0 y_2 \cdots q_N \ldots q_0 y_n q_N \ldots q_j.$$

Puisque la transduction est une bijection et le transducteur étant non ambigu, Γg_1 définit un code. En effet, si un mot de $(\Gamma g_1)^*$ avait deux factorisations, on pourrait supposer que chaque y_i est séparé du y_{i+1} suivant par $q_N \ldots q_1 q_0$. Les deux factorisations correspondent aux chemins d'un état q_i à un état q_j. Ces chemins ont la même sortie. Puisque la transduction

est une bijection, les mots d'entrée sont égaux. La non ambiguïté implique que ce sont deux factorisations basées sur un chemin unique. La définition de Γ prenant en compte les transitions non effaçantes de δ, la factorisation doit être unique.

Le morphisme g_1 est donc injectif.

Soit g_2 le morphisme défini pour chaque lettre $[q_i, x_1 \dots x_m, y_1 \dots y_n, q_j]$ de Γ par

$$[q_i, x_1 x_2 \dots x_m, y_1 y_2 \dots y_n, q_j]g_2 = (q_i, x_1, \varepsilon, q_{i_1})(q_{i_1}, x_2, \varepsilon, q_{i_2}) \dots (q_{i_{m-2}}, x_{m-1}, \varepsilon, q_k)(q_k, x_m, y_1 y_2 \dots y_n, q_j)$$

Puisque le transducteur est non ambigu, Γg_2 définit un code. Le morphisme g_2 est donc injectif.

Si π est un chemin réussi dans T, on a $\pi W_T g_u g_1^{-1} g_2 = \pi$. Si $w g_u$ appartient à $(\Gamma g_1)^*$, le mot $w g_u g_1^{-1} g_2$ est un chemin de T écrivant w et menant de q_0 à q_0. La composition $g_u g_1^{-1} g_2$ est une bijection entre le codomaine et les chemins réussis correspondants dans T.

Il existe une bijection rationnelle appartenant à $H_u H_i^{-1}$ entre les mots d'entrée et les chemins réussis correpondants. Il existe une bijection rationnelle appartenant à $H_i^{-1} H_i H_u^{-1}$ entre les chemins réussis et les mots de sortie correspondants. Donc τ appartient à $H_u H_i^{-1} H_i H_u^{-1}$.

$(H_i + H_i^{-1})^*$ est inclus dans l'ensemble des fonctions rationnelles simples ayant un monoïde libre pour domaine. Puisque la transduction inverse appartient aussi à cet ensemble, le codomaine est un monoïde libre. Chaque transduction de $(H_i + H_i^{-1})^*$ est une bijection. Par conséquent $(H_i + H_i^{-1})^*$ est inclus dans l'ensemble des bijections rationnelles simples entre monoïdes libres, qui est inclus dans $H_u H_i^{-1} H_i H_u^{-1}$. □

Afin de compléter le résultat, nous prouvons que l'ensemble $(H_i + H_i^{-1})^*$ contient strictement $H_i^{-1} H_i H_i^{-1} H_i$. Soit Rat_p l'ensemble des langages rationnels préfixes. L'intersection avec un langage rationnel R de Rat_p^* est toujours une bijection rationnelle simple entre monoïdes libres, mais elle n'appartient pas toujours à $H_i^{-1} H_i H_i^{-1} H_i$.

Lemme 2.3 *La famille des transductions réalisant l'intersection avec un langage de Rat_p^* n'est pas incluse dans $H_i^{-1} H_i H_i^{-1} H_i$.*

Preuve Le langage a^*b est un langage préfixe rationnel. Supposons que l'intersection avec le langage $(a^*b)^*$ soit une transduction τ pouvant se mettre sous la forme $h_1^{-1} h_2 h_3^{-1} h_4$, où chaque morphisme h_i est injectif.

Soit X l'alphabet sur lequel est défini le morphisme h_1 et $X_1 = X h_1$. Tout mot de $(a^*b)^*$, et en particulier tout mot de a^*b possède une X_1–factorisation unique. Soit n la longueur du plus long mot de X_1. Le premier facteur de la X_1–factorisation de $a^n b$ est une puissance de a, disons a^r. Le morphisme h_1 étant injectif, l'ensemble X_1 ne peut contenir qu'une puissance de a. Pour tout entier i de $\{0, 1, \dots, r-1\}$, la X_1–factorisation du mot $a^{r+i}b$ ne peut donc être que $a^{r+i}b = a^r \cdot a^i b$. Puisque tout mot de $(a^*b)^*$ peut être obtenu comme produit de mots de $\{b, ab, \dots, a^{r-1}b, a^r\}$, l'ensemble X_1 ne contient aucun autre mot. On peut donc définir X et h_1 par $X = \{x_0, x_1, \dots, x_{r-1}, x_a\}$, $x_a h_1 = a^r$ et $x_i h_1 = a^i b$ pour tout entier i de $\{0, 1, \dots, r-1\}$.
Définissons le morphisme h_2 sur l'alphabet X par $x_a h_2 = u$ et $x_i h_2 = u_i$ pour tout entier i

de $\{0,1,\ldots,r-1\}$.
Symétriquement, le morphisme h_4 est défini pour un certain entier s sur l'alphabet $Y = \{y_0, y_1, \ldots, y_{s-1}, y_a\}$ par $y_a h_4 = a^s$ et $y_i h_4 = a^i b$ pour tout entier i de $\{0,1,\ldots,s-1\}$.
Définissons le morphisme h_3 sur l'alphabet Y par $y_a h_3 = v$ et $y_i h_3 = v_i$ pour tout entier i de $\{0,1,\ldots,s-1\}$.
Des égalités

$$\begin{aligned} a^{rs+i}bh_1^{-1}h_2h_3^{-1}h_4 &= a^{rs+i}b \\ a^{rs+i}bh_1^{-1}h_2 &= u^s u_i \\ a^{rs+i}bh_4^{-1}h_3 &= v^r v_i \end{aligned}$$

on déduit que $u^s u_i = v^r v_i$.
Des égalités

$$\begin{aligned} a^{2rs+i}bh_1^{-1}h_2h_3^{-1}h_4 &= a^{2rs+i}b \\ a^{2rs+i}bh_1^{-1}h_2 &= u^{2s} u_i \\ a^{2rs+i}bh_4^{-1}h_3 &= v^{2r} v_i \end{aligned}$$

on déduit que $u^{2s} u_i = v^{2r} v_i$.
On a donc $u^s = v^r$ et puisque

$$\begin{aligned} a^{rs}h_1^{-1}h_2 &= u^s \\ a^{rs}h_4^{-1}h_3 &= v^r \\ a^{rs}h_1^{-1}h_2h_3^{-1}h_4 &= a^{rs} \end{aligned}$$

le mot a^{rs} appartient au domaine de τ, ce qui est contradictoire. □

Corollaire 2.1 *$H_i^{-1}H_iH_i^{-1}H_i$ est strictement inclus dans $H_iH_i^{-1}H_iH_i^{-1} = (H_i + H_i^{-1})^*$.*

2.4 Compositions de morphismes préfixes et de morphismes préfixes inverses

Nous allons maintenant prouver que la hiérarchie est plus compliquée pour les morphismes préfixes. Notre résultat positif indique que $(H_p + H_p^{-1})^*$ est égal à $H_u H_p^{-1} H_p H_p^{-1} H_p H_u^{-1}$. De plus, cet ensemble coïncide avec $\mathcal{D}_{*-1}$, l'ensemble des transductions simples et déterministes telles que les transductions inverses sont également simples et déterministes.

Nous savons que l'ensemble des fonctions simples déterministes est égal à $H_p H_p^{-1} H$ et est clos par composition [22]. Ainsi les morphismes préfixes et inverses de morphismes préfixes sont des fonctions déterministes simples. L'ensemble $(H_p + H_p^{-1})^*$ est clos par inversion. On a donc le lemme suivant :

Lemme 2.4 $(H_p + H_p^{-1})^* \subseteq \mathcal{D}_{*-1}$

Le lemme suivant va nous servir de point de départ pour montrer que l'inclusion inverse est également vraie.

Lemme 2.5 $\mathcal{D}_{*\text{-}1} \subseteq H_u H_p^{-1} H_{sa}$

Preuve Puisque les égalités $H_u H_p^{-1} H_{ne} = (H_{ne} + H_p^{-1})^* = H_u H_p^{-1} H_{sa}$ sont connues [23], nous avons juste à prouver que $\mathcal{D}_{*\text{-}1} \subseteq H_u H_p^{-1} H_{ne}$.

Soit τ une transduction de $\mathcal{D}_{*\text{-}1}$. Puisque τ^{-1} est une fonction simple, on a $\varepsilon\tau^{-1} = \varepsilon$. Par conséquent, le mot vide ne peut être l'image par τ d'un mot non vide. De plus, la transduction τ est fidèle car τ^{-1} est une fonction. Soit T un transducteur émondé réalisant τ ; aucun chemin plus long que le nombre d'états ne peut être effaçant.

On peut construire un transducteur déterministe et simple $T = (Q, X, Y, \delta, \{q_0\}, \{q_0\})$ dont chaque transition menant à l'état q_0 est non effaçante (cf. la preuve de la proposition 2.1, le transducteur résultant est également déterministe).

La transduction τ est réalisée par $h_u h_p^{-1} h_{ne}$ où

- le morphisme uniforme h_u est défini sur X par $x h_u = x q_N q_{N-1} \dots q_1 q_0$ où $N = \|Q\| - 1$.
- le morphisme préfixe h_p est défini par $[q_{i_1}, x_1 x_2 \dots x_m, y_1 y_2 \dots y_n, q_{i_{m+1}}] h_p = q_{i_1 - 1} .. q_0 x_1 q_N .. q_0 x_2 q_N .. q_0 \dots q_N .. q_0 x_n q_N .. q_{i_{m+1}}$ où $(q_{i_1}, x_1, \varepsilon, q_{i_2}) \dots (q_{i_{m-1}}, x_{m-1}, \varepsilon, q_{i_m})$ est un chemin effaçant dans T de q_{i_1} vers q_{i_m} et $(q_{i_m}, x_m, y_1 y_2 \dots y_n, q_{i_{m+1}})$ est une transition non effaçante dans T de q_{i_m} vers $q_{i_{m+1}}$.
- le morphisme h_{ne} est défini par $[q_i, x_1 x_2 \dots x_m, y_1 y_2 \dots y_n, q_j] h_p = y_1 y_2 \dots y_n$. □

La dernière partie de la composition $H_u H_p^{-1} H_{sa}$ est un morphisme strictement alphabétique appliqué à un langage rationnel. Nous nous intéressons donc aux transductions déterministes préservant les longueurs.

Soit τ une transduction préservant les longueurs réalisée par un transducteur déterministe $T = (Q, X, Y, \delta, \{q_0\}, Q_F)$. On peut associer à chaque état q_i de Q un délai r_i qui est la différence entre la longueur de l'entrée et la longueur de la sortie de tout chemin de T menant de l'état q_0 à cet état q_i. Les états initiaux et terminaux ont un délai 0.

Si le délai d'un état q_i de Q est r_i, alors le délai r_j de l'état q_j atteint à partir de q_i par la transition (q_i, x, v, q_j) est $r_j = r_i + 1 - |v|$.

Définition 2.1 *Une transduction déterministe préservant les longueurs est de délai d si elle peut être réalisée par un transducteur déterministe T tel que, pour tout chemin π partant de l'état initial, le mot lu et le mot écrit par π ont leurs longueurs liées par la relation* $0 \leq |\pi I_T| - |\pi W_T| \leq d$.

Supposer que le mot écrit par un chemin partant de l'état initial n'est jamais plus long que le mot lu par le même chemin n'est pas une vraie contrainte puisqu'on peut toujours construire un transducteur dont tous les états ont un délai positif ou nul.

Définition 2.2 *Pour un transducteur déterministe préservant les longueurs, le délai $d \geq 0$ est permanent si les états initiaux et terminaux sont de délai 0, et pour toute transition*

(q_i, x, v, q_j) *qui ne mène pas à un état terminal, le délai de l'état* q_j *est*

$$r_j = \begin{cases} r_i + 1 & si\ r_i < d, \\ d & si\ r_i = d. \end{cases}$$

Quand le transducteur est de délai permanent 1, les transitions appartiennent à

$$\{q_0\} \times X \times Y \times \{q_0\} \ \cup\ \{q_0\} \times X \times \varepsilon \times Q_{\setminus q_0} \ \cup\ Q_{\setminus q_0} \times X \times Y \times Q_{\setminus q_0} \ \cup\ Q_{\setminus q_0} \times X \times Y^2 \times \{q_0\}$$

Quand le transducteur est de délai permanent d, on peut regrouper les transitions de la manière suivante $\{q_0\} \times X^l \times Y^l \times \{q_0\} \ \cup\ \{q_0\} \times X^d \times \varepsilon \times Q_{\setminus q_0} \ \cup\ Q_{\setminus q_0} \times X^d \times Y^d \times Q_{\setminus q_0} \ \cup\ Q_{\setminus q_0} \times X^l \times Y^{d+l} \times \{q_0\}$ où $1 \leq l \leq d$.

Nous prouvons d'abord qu'un transducteur de délai permanent d existe pour chaque transduction de délai d.

Lemme 2.6 *Toute transduction de* $\mathcal{D}_{*-1}$ *préservant les longueurs de délai d peut être réalisée par un transducteur déterministe simple de délai permanent d.*

Preuve Soit τ une transduction préservant les longueurs réalisée par le transducteur $T = (P, Y, X, \delta, \{p_0\}, \{p_0\})$. Soit d le délai de la transduction. Si le délai est 0, il est permanent par définition. Supposons le délai supérieur à 0. On construit un nouveau transducteur $T_1 = (P_1, Y, X, \delta_1, \{(p_0, \varepsilon)\}, \{(p_0, \varepsilon)\})$ qui va décaler la sortie de d lettres jusqu'à ce que le chemin atteigne l'état (p_0, ε). Quand le délai est obtenu, la sortie d'une transition sera une lettre. Quand la transition atteint l'état (p_0, ε), les lettres stockées sont sorties.

L'ensemble δ_1 des transitions du transducteur T_1 est défini par

- $((p_i, v_i), y, \varepsilon, (p_j, v_i v)) \in \delta_1$ si $(p_i, y, v, p_{j \neq 0}) \in \delta$ et $r_i + |v_i| < d$
- $((p_i, \varepsilon), y, x, (p_j, v)) \in \delta_1$ si $(p_i, y, xv, p_{j \neq 0}) \in \delta$ et $r_i = d$
- $((p_i, xv_i), y, x, (p_j, v_i v)) \in \delta_1$ si $(p_i, y, v, p_{j \neq 0}) \in \delta$ et $r_i + |xv_i| = d$
- $((p_i, v_i), y, v_i v, (p_0, \varepsilon)) \in \delta_1$ si $(p_i, y, v, p_0) \in \delta$ et $r_i + |v_i| \leq d$.

On peut prouver par récurrence sur la longueur des chemins que les transducteurs T et T_1 réalisent la même transduction. Soit π un chemin du transducteur T menant de l'état p_0 à l'état p_i et π_1 un chemin du transducteur T_1 menant de l'état (p_0, ε) à l'état (p_i, v_i). Si les chemins π et π_1 lisent le même mot, alors les mots qu'ils écrivent vérifient $\pi W_T = (\pi_1 W_{T_1}) v_i$.

Ceci montre aussi que le délai d'un état (p_i, v_i) de T_1 est $R_i = r_i + |v_i|$ où r_i est le délai de l'état p_i dans T.

L'évolution du délai dans T était $r_j = r_i + 1 - |v|$ pour une transition (p_i, y, v, p_j). Une transition $((p_i, v_i), y, x, (p_j, v_j))$ qui ne mène pas à (p_0, ε) change donc le délai de la manière suivante :

- quand $R_i = r_i + |v_i| < d$, la transition est $((p_i, v_i), y, \varepsilon, (p_j, v_i v))$.
  On a $R_j = r_j + |v_i v| = r_i + 1 - |v| + |v_i v| = r_i + |v_i| + 1 = R_i + 1$

- quand $R_i = r_i + |v_i| = d$, la transition est
  - soit $((p_i, \varepsilon), y, x, (p_j, v))$. On a alors
    $R_j = r_j + |v| = r_i + 1 - |xv| + |v| = r_i = R_i = d$.
  - soit $((p_i, xv_i'), y, x, (p_j, v_i'v))$. On a alors
    $R_j = r_j + |v_i'v| = r_i + 1 - |v| + |v_i'v| = r_i + |v_i'| + 1 = R_i = d$.

Par conséquent, le nouveau transducteur est de délai permanent d. □

Lemme 2.7 *Soit τ_1 une transduction de $\mathcal{D}_{*\text{-}1}$ préservant les longueurs. Il existe une transduction de $\mathcal{D}_{*\text{-}1}$ préservant les longueurs τ_2 de $(X_0 \cup X_1)^*$ dans $(Y_0 \cup Y_1)^*$, où X_0 et X_1 (resp. Y_0 et Y_1) sont deux alphabets disjoints, telle que*

- *Le domaine de τ_2 est inclus dans $(X_1^* X_0)^*$,*
- *Le codomaine de τ_2 est inclus dans $(Y_1^* Y_0)^*$,*
- *τ_2, τ_2^{-1} sont de mêmes délais que τ_1, τ_1^{-1},*
- *$\tau_1 \in H_u H_p^{-1} \tau_2 H_p H_u^{-1}$.*

Preuve Soit τ_1 une transduction préservant les longueurs réalisée par le transducteur $T_1 = (Q, X, Y, \delta_1, \{q_0\}, \{q_0\})$ et dont la transduction inverse est réalisée par le transducteur $T_{\text{-}1} = (P, Y, X, \delta_{\text{-}1}, \{p_0\}, \{p_0\})$.

La transduction τ_2 sera définie sur les transitions de T_1 et $T_{\text{-}1}$. L'alphabet du domaine de τ_2 est composé de $X_0 = \{[q_i, x, v, q_0] / (q_i, x, v, q_0) \in \delta_1\}$ et $X_1 = \{[q_i, x, v, q_{j \neq 0}] / (q_i, x, v, q_{j \neq 0}) \in \delta_1\}$. L'alphabet du codomaine de τ_2 est composé de $Y_0 = \{[p_i, y, u, p_0] / (p_i, y, u, p_0) \in \delta_{\text{-}1}\}$ et $Y_1 = \{[p_i, y, u, p_{j \neq 0}] / (p_i, y, u, p_{j \neq 0}) \in \delta_{\text{-}1}\}$.

Il existe une bijection f_{δ_1} (resp. $f_{\delta_{\text{-}1}}$) entre le domaine (resp. le codomaine) de τ_1 et les chemins réussis correspondants dans T_1 (resp. $T_{\text{-}1}$). Les bijections f_{δ_1} et $f_{\delta_{\text{-}1}}$ appartiennent à $H_u H_p^{-1} \subset \mathcal{D}_{*\text{-}1}$. Définissons τ_2 par $\tau_2 = f_{\delta_1}^{-1} \tau_1 f_{\delta_{\text{-}1}}$. Alors τ_2 appartient à $\mathcal{D}_{*\text{-}1}$.

Puisque τ_1 (resp. τ_1^{-1}) est simple, l'ensemble des chemins réussis dans T_1 (resp. $T_{\text{-}1}$) est inclus dans $(X_1^* X_0)^*$ (resp. $(Y_1^* Y_0)^*$). Puisque f_{δ_1} (resp. $f_{\delta_{\text{-}1}}$) est une bijection entre le domaine (resp. codomaine) de τ_1 et les chemins réussis correspondants dans T_1 (resp. $T_{\text{-}1}$), le domaine de τ_2 est inclus dans $(X_1^* X_0)^*$ et le codomaine de τ_2 est inclus dans $(Y_1^* Y_0)^*$.

Puisque f_{δ_1} (resp. $f_{\delta_{\text{-}1}}$) est une bijection avec un domaine égal au domaine (resp. codomaine) de τ_1, la transduction τ_1 est égale à $f_{\delta_1} \tau_2 f_{\delta_{\text{-}1}}^{-1}$ et appartient à $H_u H_p^{-1} \tau_2 H_p H_u^{-1}$.

Puisque τ_1 est déterministe, la transformation f_{δ_1} (resp. $f_{\delta_{\text{-}1}}$) des mots d'entrée (resp. de sortie) en leurs transitions associées est faite avec un délai 0. La transformation inverse $f_{\delta_1}^{-1}$ (resp. $f_{\delta_{\text{-}1}}^{-1}$) a aussi un délai 0. Donc τ_2 (resp. τ_2^{-1}) a les mêmes délais que τ_1 (resp. τ_1^{-1}). □

Lemme 2.8 *Soit τ_1 une transduction de $\mathcal{D}_{*\text{-}1}$ préservant les longueurs de délai 0 de $(X_1^* X_0)^*$ dans $(Y_1^* Y_0)^*$, où X_0 et X_1 (resp. Y_0 et Y_1) sont deux alphabets disjoints. Il existe une transduction de $\mathcal{D}_{*\text{-}1}$ préservant les longueurs τ_2 de délai 0 telle que τ_2^{-1} est de délai ≤ 1 et*

$$\tau_1 \in H_p^{-1} \tau_2 H_p$$

Preuve Soit τ_1 une transduction préservant les longueurs de délai 0 appartenant à $\mathcal{D}_{*-1}$. Supposons le domaine de τ_1 inclus dans $(X_1^*X_0)^*$ et le codomaine de τ_1 inclus dans $(Y_1^*Y_0)^*$, où X_0 et X_1 (resp. Y_0 et Y_1) sont deux alphabets disjoints. La transduction τ_1 est réalisée par un transducteur $T_1 = (Q, X, Y, \delta_1, \{q_0\}, \{q_0\})$ de délai 0. La transduction τ_1^{-1} est de délai d et est réalisée par un transducteur $T_{-1} = (P, Y, X, \delta_{-1}, \{p_0\}, \{p_0\})$ de délai d. Par le lemme 2.6, on peut choisir les tranducteurs tels que les délais sont permanents. Supposons le délai de T_{-1} plus grand que 1.

Définissons h_{p_1} et h_{p_2} par

$$\begin{aligned}
[x_{i_1}, ..., x_{i_d}]\, h_{p_1} &= x_{i_1}...x_{i_d} \quad \forall x_{i_1}, ..., x_{i_d} \in X_1, \\
[x_{i_1}, ..., x_{i_l}, x_j]\, h_{p_1} &= x_{i_1}...x_{i_l}x_j \quad \forall x_{i_1}, ..., x_{i_l} \in X_1, 0 \leq l < d \text{ et } x_j \in X_0.
\end{aligned}$$

$$\begin{aligned}
[y_{i_1}, ..., y_{i_d}]\, h_{p_2} &= y_{i_1}...y_{i_d} \quad \forall y_{i_1}, ..., y_{i_d} \in Y_1, \\
[y_{i_1}, ..., y_{i_l}, y_j]\, h_{p_2} &= y_{i_1}...y_{i_l}y_j \quad \forall y_{i_1}, ..., y_{i_l} \in Y_1, 0 \leq l < d \text{ et } y_j \in Y_0.
\end{aligned}$$

Puisque les alphabets X_0 et X_1 (resp. Y_0 et Y_1) sont disjoints, les morphismes h_{p_1} et h_{p_2} sont préfixes.

La transduction τ_2 est définie par $\tau_2 = h_{p_1}\tau_1 h_{p_2}^{-1}$. La transduction $h_{p_1}^{-1}\tau_2 h_{p_2}$ est égale à $h_{p_1}^{-1}h_{p_1}\tau_1 h_{p_2}^{-1}h_{p_2}$. La composition $h_{p_1}^{-1}h_{p_1}$ est équivalente à une intersection avec le langage rationnel $(X_1^*X_0)^*(X_1^d)^*$. La composition $h_{p_2}^{-1}h_{p_2}$ est équivalente à une intersection avec le langage rationnel $(Y_1^*Y_0)^*(Y_1^d)^*$. Le domaine de τ_1 est inclus dans $(X_1^*X_0)^*$ et le codomaine de τ_1 est inclus dans $(Y_1^*Y_0)^*$. La transduction τ_1 est donc égale à $h_{p_1}^{-1}\tau_2 h_{p_2}$.

La transduction τ_2 préserve les longueurs et a un délai 0. Elle sort un d-uple (resp. l-uple) pour chaque d-uple (resp. l-uple) en entrée.

Le transducteur T_{-1} ayant un délai permanent égal à d, on peut regrouper les transitions de la manière suivante : $\{p_0\} \times Y_1^l Y_0 \times X_1^l X_0 \times \{p_0\} \cup \{p_0\} \times Y_1^d \times \varepsilon \times P_{\setminus p_0} \cup P_{\setminus p_0} \times Y_1^d \times X_1^d \times P_{\setminus p_0} \cup P_{\setminus p_0} \times Y_1^l Y_0 \times X_1^{d+l} X_0 \times \{p_0\}$ où $1 \leq l < d$. La transduction τ_2^{-1} a alors un délai permanent 1. □

Lemme 2.9 *Si τ est une transduction de $\mathcal{D}_{*-1}$ préservant les longueurs de délai 0 telle que τ^{-1} est de délai ≤ 1, alors*

$$\tau \in H_u H_p^{-1} H_p H_p^{-1} H_p H_u^{-1}$$

Preuve Soit τ une transduction de $\mathcal{D}_{*-1}$ préservant les longueurs de délai 0 telle que τ^{-1} est de délai ≤ 1. Ainsi la transduction τ^{-1} a un délai 1.

La transduction τ est réalisée par un transducteur émondé $T = (Q, X, Y, \delta, \{q_0\}, \{q_0\})$ qui est de délai 0.
La transduction τ^{-1} est réalisée par un tranducteur émondé $T_{-1} = (P, Y, X, \delta_{-1}, \{p_0\}, \{p_0\})$ qui est de délai permanent 1.

Comme vu dans le lemme 2.2, il existe une bijection entre X^* (resp. Y^*) et les chemins réussis dans T (resp. T_{-1}) realisée par $h_{u_1}h_{p_1}^{-1}$ (resp. $h_{u_2}h_{p_2}^{-1}$).

On définit alors h_{p_3} et h_{p_4} sur δ et δ_{-1} considérés comme des alphabets.

$$\begin{array}{rcll}
(q_i,x,y,q_j)h_{p_3} & = & q_{i-1}..q_0yp_n...p_0xq_m..q_j & \text{pour} \quad (q_i,x,y,q_j)\in\delta \\
\\
(p_0,y,\varepsilon,p_{j\neq 0})h_{p_4} & = & yp_n...p_j & \text{pour} \quad (p_0,y,\varepsilon,p_{j\neq 0})\in\delta_{-1} \\
(p_{i\neq 0},y,x,p_{j\neq 0})h_{p_4} & = & p_{i-1}..p_0xq_m...q_0yp_n..p_j & \text{pour} \quad (p_{i\neq 0},y,x,p_{j\neq 0})\in\delta_{-1} \\
(p_{i\neq 0},y,x_1x_2,p_0)h_{p_4} & = & p_{i-1}..p_0x_1q_m...q_0yp_n...p_0x_2q_m...q_0 & \text{pour} \quad (p_{i\neq 0},y,x_1x_2,p_0)\in\delta_{-1} \\
(p_0,y,x,p_0)h_{p_4} & = & yp_n...p_0xq_m...q_0 & \text{pour} \quad (p_0,y,x,p_0)\in\delta_{-1}
\end{array}$$

Puisque le tranducteur T_{-1} est de délai permanent 1, les transitions appartiennent à $\{p_0\}\times Y\times X\times\{p_0\}\ \cup\ \{p_0\}\times Y\times\varepsilon\times P_{\setminus p_0}\ \cup\ P_{\setminus p_0}\times Y\times X\times P_{\setminus p_0}\ \cup\ P_{\setminus p_0}\times Y\times X^2\times\{p_0\}$. Donc toutes les transitions de δ_{-1} apparaissent dans la définition de h_{p_4}.

Les tranducteurs T et T_{-1} étant déterministes, les morphismes h_{p_3} et h_{p_4} sont préfixes.

Il est facile de vérifier que, si π est un chemin du transducteur T menant de q_0 à q_0, lisant le mot $\pi I_T = x_{i_1}...x_{i_k}$ et écrivant le mot $\pi W_T = y_{i_1}...y_{i_k}$, on a

$$\pi I_T h_{u_1} h_{p_1}^{-1} h_{p_3} = y_{i_1}p_n...p_0x_{i_1}q_m...q_0\cdot\cdot\cdot\cdot\cdot y_{i_k}p_n...p_0x_{i_k}q_m...q_0$$

Si π est un chemin du transducteur T menant de q_0 à q_0, il existe dans le transducteur T_{-1} un chemin π_{-1} menant de p_0 à p_0, lisant le mot écrit par π et écrivant le mot lu par π. On a alors

$$\pi W_T h_{u_2} h_{p_2}^{-1} h_{p_4} = \pi_{-1} I_{T_{-1}} h_{u_2} h_{p_2}^{-1} h_{p_4} = y_{i_1}p_n...p_0x_{i_1}q_m...q_0\cdot\cdot\cdot\cdot\cdot y_{i_k}p_n...p_0x_{i_k}q_m...q_0$$

Réciproquement, si $x_{i_1}...x_{i_k}h_{u_1}h_{p_1}^{-1}h_{p_3}h_{p_4}^{-1}h_{p_2}h_{u_2}^{-1} = y_{i_1}...y_{i_k}$, on peut voir que $x_{i_1}...x_{i_k}h_{u_1}h_{p_1}^{-1}$ est un chemin du transducteur T menant de q_0 à q_0 et écrivant le mot $y_{i_1}...y_{i_k}$.

Donc, $\tau = h_{u_1}h_{p_1}^{-1}h_{p_3}h_{p_4}^{-1}h_{p_2}h_{u_2}^{-1}$. □

Lemme 2.10 *Soit τ une transduction de $\cap Rat_p^* H_{sa}$ telle que τ^{-1} est simple et déterministe. La transduction τ appartient à $H_uH_p^{-1}H_pH_p^{-1}H_pH_u^{-1}$.*

Preuve La transduction τ appartient à $\mathcal{D}_{*-1}$ et préserve les longueurs. Le délai de τ est 0. Le délai de τ^{-1} est d. On peut donc appliquer les lemmes précédents. En combinant les lemmes 2.7 et 2.8, il existe une transduction de $\mathcal{D}_{*-1}$ préservant les longueurs τ_1 de délai 0 telle que τ_1^{-1} est de délai ≤ 1 et on a

$$\begin{array}{rcll}
\tau & \in & H_uH_p^{-1}H_p^{-1}\tau_1H_pH_pH_u^{-1} & \textit{lemmes } 2.7,\ 2.8 \\
 & \in & H_uH_p^{-1}H_p^{-1}H_uH_p^{-1}H_pH_p^{-1}H_pH_u^{-1}H_pH_pH_u^{-1} & \textit{lemme } 2.9
\end{array}$$

Il a été prouvé dans [23] que $H_p^{-1}H_u \subset H_uH_p^{-1}$. On a donc $\tau \in H_uH_p^{-1}H_pH_p^{-1}H_pH_u^{-1}$. □

Proposition 2.2 $\mathcal{D}_{*-1} = (H_p + H_p^{-1})^* = H_uH_p^{-1}H_pH_p^{-1}H_pH_u^{-1}$.

Preuve Soit τ une transduction de $\mathcal{D}_{*-1}$. Nous avons prouvé que $\mathcal{D}_{*-1} \subseteq H_uH_p^{-1}H_{sa}$ (lemme 2.5). L'image d'un morphisme uniforme suivi par l'inverse d'un morphisme préfixe

est un langage rationnel appartenant à Rat_p^*. Il existe donc une transduction τ_1 de $\cap Rat_p^* H_{sa}$, un morphisme uniforme h_u et un morphisme préfixe h_p tels que $\tau = h_u h_p^{-1} \tau_1$. On a donc $h_p h_u^{-1} \tau = h_p h_u^{-1} h_u h_p^{-1} \tau_1$. La composition $h_p h_u^{-1} h_u h_p^{-1}$ est équivalente à une intersection avec un langage rationnel qui est le domaine de τ_1. La transduction τ_1 est donc égale à $h_p h_u^{-1} \tau$ et appartient à $\mathcal{D}_{*-1}$. On peut appliquer le lemme précédent : $\tau_1 \in H_u H_p^{-1} H_p H_p^{-1} H_p H_u^{-1}$. Puisque $\tau = h_u h_p^{-1} \tau_1$,

$$\mathcal{D}_{*-1} \subseteq H_u H_p^{-1} H_u H_p^{-1} H_p H_p^{-1} H_p H_u^{-1} \subseteq H_u H_p^{-1} H_p H_p^{-1} H_p H_u^{-1} \subseteq (H_p + H_p^{-1})^*$$

Le lemme 2.4 donne les égalités. □

Nous allons maintenant prouver que l'ensemble des compositions de morphismes préfixes et de morphismes préfixes inverses n'est inclus ni dans $H_p^{-1} H_p H_p^{-1} H_p$, ni dans $H_p H_p^{-1} H_p H_p^{-1}$.

Proposition 2.3 *$\mathcal{D}_{*-1}$ contient strictement $H_p^{-1} H_p H_p^{-1} H_p$.*

Preuve Rat_p étant l'ensemble des langages rationnels préfixes, on a $\cap Rat_p^*$ inclus dans $\mathcal{D}_{*-1}$. on peut même vérifier que $\cap Rat_p^* \subset H_u H_p^{-1} H_p H_u^{-1}$.

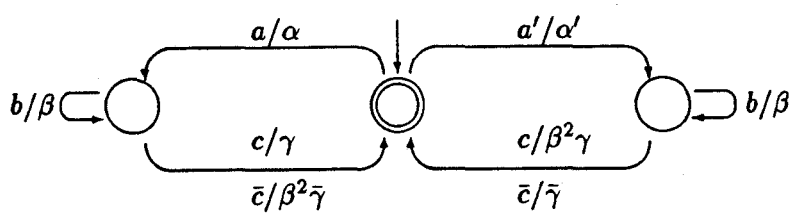
Utilisons $H_u H_p^{-1}$ pour construire les chemins réussis dans l'automate déterministe reconnaissant le langage rationnel préfixe. Utilisons $H_p H_u^{-1}$ pour faire l'inverse.

Le lemme 2.3 montre que $\cap Rat_p^*$ n'est pas inclus dans $H_i^{-1} H_i H_i^{-1} H_i$, encore moins dans $H_p^{-1} H_p H_p^{-1} H_p$. Donc $\mathcal{D}_{*-1} = (H_p + H_p^{-1})^*$ contient strictement $H_p^{-1} H_p H_p^{-1} H_p$. □

La proposition suivante montre que $(H_p + H_p^{-1})^*$ n'est pas plus dans $H_p H_p^{-1} H_p H_p^{-1}$.

Proposition 2.4 *$\mathcal{D}_{*-1}$ contient strictement $H_p H_p^{-1} H_p H_p^{-1}$.*

Preuve Soit τ la transduction définie par le transducteur suivant



Supposons la transduction τ égale à $h_1 h_2^{-1} h_3 h_4^{-1}$. Nous allons essayer de définir les différents morphismes.

$$\begin{array}{llll} ah_1 & = u & \alpha h_4 & = U \\ a'h_1 & = u' & \alpha' h_4 & = U' \\ bh_1 & = v & \beta h_4 & = V \\ ch_1 & = w_0 & \gamma h_4 & = W_0 \\ \bar{c}h_1 & = \bar{w}_0 & \bar{\gamma} h_4 & = \bar{W}_0 \end{array}$$

Les morphismes h_2 et h_3 sont définis sur un alphabet Γ.
Les mots $uv^n w_0$, $uv^n \bar{w}_0$, $u'v^n w_0$, $u'v^n \bar{w}_0$ appartiennent au domaine de h_2^{-1}.
Soit l la plus grande longueur des mots de Γh_2. Alors, pour n plus grand que $l * |v|$, il existe une factorisation $uv^n w_0 = (uv^{k_0} v_1)(v_2 v^{k-1} v_1)(v_2 v^i w_0)$ et une factorisation $u'v^n w_0 = (u'v^{k'_0} v_3)(v_4 v^{k'-1} v_3)(v_4 v^j w_0)$ telles que chaque partie des factorisations appartient à $(\Gamma h_2)^*$. L'ensemble Γh_2 étant préfixe, il existe des factorisations $(uv^{k_0} v_1)(v_2 v^{k-1} v_1)(v_2 v^i w_0)$ et des factorisations $(u'v^{k'_0} v_3)(v_4 v^{k'-1} v_3)(v_4 v^j w_0)$ pour $0 \leq i \leq k-1$ et $0 \leq j \leq k'-1$ telles que chaque partie des factorisations appartienne à $(\Gamma h_2)^*$. Selon ces factorisations, nous allons construire des morphismes g_2 et g_3 tels que $h_2^{-1} h_3 = g_2^{-1} g_3$ sur le langage $(uv^* w_0 + uv^* \bar{w}_0 + u'v^* w_0 + u'v^* \bar{w}_0)^*$. Les factorisations dépendent du début de w_0 et $\bar{w}_0$. Puisque le morphisme h_1 est préfixe, v n'est pas un préfixe de w_0 (resp. $\bar{w}_0$) mais v_1 ou v_3 pourrait.

Cette transduction étant 'symétrique' (relativement à a et a'), supposons que $|v_1| \leq |v_3|$. Il y a neuf cas

$$w_0 \stackrel{?}{=} v_1 w \stackrel{?}{=} v_3 w'$$

$$\bar{w}_0 \stackrel{?}{=} v_1 \bar{w} \stackrel{?}{=} v_3 \bar{w}'$$

1. Supposons d'abord que $w_0 = v_1 w$ et $\bar{w}_0 = v_1 \bar{w}$. ($w_0 = v_3 w'$ ou $w_0 \neq v_3 w'$, $\bar{w}_0 = v_3 \bar{w}'$ ou $\bar{w}_0 \neq v_3 \bar{w}'$).

   On définit alors g_2 et g_3 par

$$\begin{array}{llll} Xg_2 & = uv^{k_0} v_1 & Xg_3 & = uv^{k_0} v_1 h_2^{-1} h_3 \\ X'g_2 & = u'v^{k'_0} v_3 & X'g_3 & = u'v^{k'_0} v_3 h_2^{-1} h_3 \\ Yg_2 & = v_2 v^{k-1} v_1 & Yg_3 & = v_2 v^{k-1} v_1 h_2^{-1} h_3 \\ Y'g_2 & = v_4 v^{k'-1} v_3 & Y'g_3 & = v_4 v^{k'-1} v_3 h_2^{-1} h_3 \\ Z_i g_2 & = v_2 v^i v_1 w & Z_i g_3 & = v_2 v^i v_1 w h_2^{-1} h_3 \text{ pour tout } i \text{ de } \{0, \ldots, k-1\} \\ Z_{k-1} g_2 & = w & Z_{k-1} g_3 & = w h_2^{-1} h_3 \\ \bar{Z}_i g_2 & = v_2 v^i v_1 \bar{w} & \bar{Z}_i g_3 & = v_2 v^i v_1 \bar{w} h_2^{-1} h_3 \text{ pour tout } i \text{ de } \{0, \ldots, k-1\} \\ \bar{Z}_{k-1} g_2 & = \bar{w} & \bar{Z}_{k-1} g_3 & = \bar{w} h_2^{-1} h_3 \end{array}$$

   Les morphismes h_2 et h_3 étant injectifs, g_3 est un morphisme.

   Certains mots $uv^i w_0$, $uv^i \bar{w}_0$, $u'v^i w_0$, $u'v^i \bar{w}_0$, avec $0 \leq i < k_0$, ou $v_4 v^i w_0$, $v_4 v^i \bar{w}_0$, avec $0 \leq i \leq k'-1$, interviennent pour obtenir l'équivalence $h_2^{-1} h_3 = g_2^{-1} g_3$ sur le langage $(uv^* w_0 + uv^* \bar{w}_0 + u'v^* w_0 + u'v^* \bar{w}_0)^*$. Mais ces mots n'interfèrent pas avec la preuve du premier cas.

   Il est facile de vérifier que

$$\begin{array}{llll} XY^n Z_i g_3 h_4^{-1} & = XY^n Z_i g_2 h_1^{-1} \tau & = \alpha \beta^{k_0 + kn + i + 1} \gamma \text{ pour tout } i \text{ de } \{0, \ldots, k-1\} \\ XY^n Z_{k-1} g_3 h_4^{-1} & = XY^n Z_{k-1} g_2 h_1^{-1} \tau & = \alpha \beta^{k_0 + kn} \gamma \\ XY^n \bar{Z}_i g_3 h_4^{-1} & = XY^n \bar{Z}_i g_2 h_1^{-1} \tau & = \alpha \beta^{k_0 + kn + i + 3} \bar{\gamma} \text{ pour tout } i \text{ de } \{0, \ldots, k-1\} \\ XY^n \bar{Z}_{k-1} g_3 h_4^{-1} & = XY^n \bar{Z}_{k-1} g_2 h_1^{-1} \tau & = \alpha \beta^{k_0 + kn + 2} \bar{\gamma} \end{array}$$

Donc,

$$\begin{aligned} XY^nZ_ig_3 &= UV^{k_0+kn+i+1}W_0 \text{ pour tout } i \text{ de } \{0,\ldots,k-1\} \\ XY^nZ_{k-1}g_3 &= UV^{k_0+kn}W_0 \\ XY^n\bar{Z}_ig_3 &= UV^{k_0+kn+i+3}\bar{W}_0 \text{ pour tout } i \text{ de } \{0,\ldots,k-1\} \\ XY^n\bar{Z}_{k-1}g_3 &= UV^{k_0+kn+2}\bar{W}_0 \end{aligned}$$

Ces équations étant vérifiées pour tout n, on doit avoir $Yg_3 = V_2V^{k-1}V_1$. Nous distinguons deux cas.

Quand $k = 1$, on a $Yg_3 = V_2V_1$ et $XY^n\bar{Z}_0g_3 = UV^{k_0+n+2}\bar{W}_0$. Donc soit $\bar{Z}_0g_3$ appartient à $V_2V^*\bar{W}_0$ soit il est un facteur droit de $\bar{W}_0$. Si $\bar{Z}_0g_3$ appartient à $V_2V^+\bar{W}_0$, le mot Yg_3 est un préfixe de $\bar{Z}_0g_3$, ce qui signifie que $v_2v_1h_2^{-1}h_3$ est un préfixe de $\bar{w}h_2^{-1}h_3$. Les morphismes h_2 et h_3 étant préfixes, le mot v_2v_1 devrait être un préfixe de $\bar{w}$ et le mot v devrait être un préfixe de $v_1\bar{w} = \bar{w}_0$; ceci est impossible car h_1 est un morphisme préfixe. Donc $\bar{Z}_0g_3$ est un facteur droit de $V_2\bar{W}_0$. On a $X\bar{Z}_0g_3 = UV^{k_0+2}\bar{W}_0$. Donc par g_3, X génère au moins $UV^{k_0+1}V_1$, ce qui contredit $XZ_0g_3h_4^{-1} = \alpha\beta^{k_0}\gamma$.

Quand $k > 1$, on a $Yg_3 = V_2V^{k-1}V_1$ et $XY^n\bar{Z}_{k-2}g_3 = UV^{k_0+kn+k+1}\bar{W}_0$. Donc soit $\bar{Z}_{k-2}g_3$ appartient à $V_2V^*\bar{W}_0$ soit il est un facteur droit de $\bar{W}_0$. Si $\bar{Z}_{k-2}g_3$ appartient à $V_2V^kV^*\bar{W}_0$, le mot Yg_3 est un préfixe de $\bar{Z}_{k-2}g_3$, ce qui signifie que $v_2v^{k-1}v_1h_2^{-1}h_3$ est un préfixe de $v_2v^{k-2}v_1\bar{w}h_2^{-1}h_3$. Les morphismes h_2 et h_3 étant préfixes, le mot v_2v_1 devrait être un préfixe de $\bar{w}$ et le mot v devrait être un préfixe de $v_1\bar{w} = \bar{w}_0$; ceci est impossible car h_1 est un morphisme préfixe. Donc $\bar{Z}_{k-2}g_3$ est un facteur droit de $V_2V^{k-1}\bar{W}_0$. On a $X\bar{Z}_{k-2}g_3 = UV^{k_0+k+1}\bar{W}_0$. Donc par g_3, X génère au moins $UV^{k_0+1}V_1$, ce qui contredit $XZ_{k-1}g_3h_4^{-1} = \alpha\beta^{k_0}\gamma$.

2. Supposons maintenant que $w_0 = v_1w$ et $\bar{w}_0 \neq v_1\bar{w}$. ($w_0 = v_3w'$ ou $w_0 \neq v_3w'$).

   On a juste à changer la définition de $\bar{Z}_ig_2$ et $\bar{Z}_ig_3$.

   $\bar{Z}_ig_2 = v_2v^i\bar{w}_0$ et $\bar{Z}_ig_2 = v_2v^i\bar{w}_0h_2^{-1}h_3$ pour tout i de $\{0,\ldots,k-1\}$.

   Donc, $XY^n\bar{Z}_ig_3h_4^{-1} = XY^n\bar{Z}_ig_2h_1^{-1}\tau = \alpha\beta^{k_0+kn+i+3}\bar{\gamma}$ pour tout i de $\{0,\ldots,k-1\}$, et $XY^n\bar{Z}_ig_3 = UV^{k_0+kn+i+3}\bar{W}_0$ pour tout i de $\{0,\ldots,k-1\}$.

   On a donc $X\bar{Z}_{k-1}g_3 = UV^{k_0+k+2}\bar{W}_0$.

   Le mot $\bar{Z}_{k-1}g_3$ doit être un facteur droit de $V_2V^{k-1}\bar{W}_0$. Donc X génère par g_3 au moins $UV^{k_0+2}V_1$, ce qui contredit $XZ_0g_3h_4^{-1} = \alpha\beta^{k_0}\gamma$ quand $k = 1$ et $XZ_0g_3h_4^{-1} = \alpha\beta^{k_0+1}\gamma$ quand $k > 1$.

3. Supposons maintenant que $w_0 \neq v_1w$ et $\bar{w}_0 = v_1\bar{w} = v_3\bar{w}'$.

   En utilisant les transformations de $X'Y'^nZ'_i$ et $X'Y'^n\bar{Z}'_i$, on obtient une contradiction similaire à celle du cas précédent. La lettre Y' doit générer par g_3 un mot $V_4V^{k'-1}V_3$. En utilisant l'image de $X'\bar{Z}'_{k'-1}$, on obtient que X' génère par g_3 au moins $U'V'^{k'_0+2}V_3$, ce qui contredit $X'\bar{Z}'_0g_3h_4^{-1} = \alpha'\beta'^{k'_0}\bar{\gamma}$ quand $k' = 1$ et $X'\bar{Z}'_0g_3h_4^{-1} = \alpha'\beta'^{k'_0+1}\bar{\gamma}$ quand $k' > 1$.

4. Supposons maintenant que $w_0 \neq v_1w$ et $\bar{w}_0 = v_1\bar{w} \neq v_3\bar{w}'$.

   En utilisant la même méthode, on obtient que X' génère par g_3 au moins $U'V'^{k'_0+2}V_3$, ce qui contredit $X'\bar{Z}'_0g_3h_4^{-1} = \alpha'\beta'^{k'_0+1}\bar{\gamma}$.

5. Supposons maintenant que $w_0 \neq v_1 w$ et $\bar{w}_0 \neq v_1 \bar{w}$.

   En utilisant l'image de $X\bar{Z}_{k-1}$, on obtient que X génère par g_3 au moins $UV^{k_0+2}V_1$, ce qui contredit $XZ_0 g_3 h_4^{-1} = \alpha\beta^{k_0+1}\gamma$.

Donc la transduction τ n'appartient pas à $H_p H_p^{-1} H_p H_p^{-1}$. Cependant les transductions τ et τ^{-1} sont déterministes et simples. Par conséquent, $H_p H_p^{-1} H_p H_p^{-1}$ est strictement inclus dans $\mathcal{D}_{*-1}$. □

2.5 Liens entre le cas injectif et le cas préfixe-suffixe

Il a été montré dans [21, 23] que les fonctions rationnelles peuvent être caractérisées comme une composition faisant intervenir aussi bien un morphisme injectif que des morphismes préfixes et suffixes : $M_r H_u H_i^{-1} H_a = M_r H_u H_p^{-1} H_s^{-1} H_a$. D'autre part, on sait aussi que $H_i^{-1} \neq H_p^{-1} H_s^{-1}$. Nous allons terminer ce chapitre par l'étude de la relation entre $(H_i + H_i^{-1})^*$ et $(H_p + H_p^{-1} + H_s + H_s^{-1})^*$. Notons $H_{p,s}$ l'ensemble $H_p + H_s$.

Proposition 2.5 *$(H_i + H_i^{-1})^*$ contient strictement $(H_{p,s} + H_{p,s}^{-1})^*$.*

Preuve Afin de prouver l'inclusion stricte, nous allons utiliser le morphisme h_i défini par $xh_i = a$, $yh_i = ab$, $zh_i = b^2a$ et $th_i = b^3ab^4$. Ce morphisme est construit sur un code qui n'est pas une composition de codes préfixes et de codes suffixes [4].

Supposons que h_i^{-1} appartienne à $(H_{p,s} + H_{p,s}^{-1})^*$. Puisque les morphismes préfixes et les morphismes suffixes sont des morphismes non effaçants, et puisque l'identité appartient à H_{ne} et $H_{p,s}^{-1}$, on a $(H_{p,s} + H_{p,s}^{-1})^* \subseteq (H_{ne} H_{p,s}^{-1})^*$.

Les morphismes non effaçants peuvent être exprimés comme compositions de morphismes uniformes, morphismes préfixes ou suffixes inverses, et morphismes strictement alphabétiques : $H_{ne} \subset H_u H_p^{-1} H_{sa}$ et $H_{ne} \subset H_{lu} H_s^{-1} H_{sa}$. De plus, on a certaines propriétés de commutation sur ces sortes de morphismes : $H_{sa} H_p^{-1} \subset H_p^{-1} H_{sa}$ et $H_{sa} H_s^{-1} \subset H_s^{-1} H_{sa}$ (cf. [23]).

En utilisant ces propriétés, on obtient

$$\begin{aligned} H_{ne} H_p^{-1} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne} &\subseteq H_u H_p^{-1} H_{sa} H_p^{-1} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne} \\ &\subseteq H_u H_p^{-1} H_{sa} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne} \\ &\subseteq H_u H_p^{-1} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne} \end{aligned}$$

$$H_{ne} H_s^{-1} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne} \subseteq H_{lu} H_s^{-1} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne}$$

Soit $D = \{a, ab, b^2, b^3ab^4\}$. Si h_i^{-1} appartient à $h_u h_p^{-1} (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne}$, le domaine de h_i^{-1} est D^* et doit être inclus dans le domaine de $h_u h_p^{-1}$. Puisque le mot a (resp. ab) appartient à D, le mot ah_u (resp. abh_u) appartient au domaine de h_p^{-1}. Par conséquent, le mot bh_u appartient aussi au domaine de h_p^{-1}. Le domaine de $h_u h_p^{-1}$ est $\{a, b\}^*$ et $h_u h_p^{-1}$ est équivalent à un morphisme non effaçant. Donc h_i^{-1} devrait appartenir à $H_{ne}(H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne} = (H_{ne} H_{p,s}^{-1})^{n\geq 0} H_{ne}$.

Si h_i^{-1} appartient à $h_{lu}h_s^{-1}(H_{ne}H_{p,s}^{-1})^{n\geq 0}H_{ne}$, le domaine de h_i^{-1} doit être inclus dans le domaine de $h_{lu}h_s^{-1}$. Puisque le mot a (resp. b^2a, b^3ab^4) appartient à D, le mot ah_{lu} (resp. b^2ah_{lu}, $b^3ab^4h_{lu}$) appartient au domaine de h_s^{-1}. Par conséquent, les mots b^2h_{lu}, $b^3ab^2h_{lu}$, b^3ah_{lu}, b^3h_{lu}, bh_{lu} appartiennent aussi au domaine de h_s^{-1}. Le domaine de $h_{lu}h_s^{-1}$ est $\{a,b\}^*$ et $h_{lu}h_s^{-1}$ est équivalent à un morphisme non effaçant. Donc h_i^{-1} devrait appartenir à $H_{ne}(H_{ne}H_{p,s}^{-1})^{n\geq 0}H_{ne} = (H_{ne}H_{p,s}^{-1})^{n\geq 0}H_{ne}$.

Par induction, le morphisme inverse h_i^{-1} devrait appartenir à H_{ne}, ce qui est impossible. □

Conclusion

Nous avons prouvé dans ce chapitre que la famille $(H_i + H_i^{-1})^*$ des compositions de morphismes injectifs et de morphismes injectifs inverses est égale à la famille $(H_iH_i^{-1})^2$ et contient strictement la famille $(H_i^{-1}H_i)^2$. Nous avons également prouvé que la famille $(H_p + H_p^{-1})^*$ des compositions de morphismes préfixes et de morphismes préfixes inverses n'est autre que la famille $\mathcal{D}_{*-1}$ des transductions simples et déterministes telles que les transductions inverses sont également simples et déterministes, et est égale à $(H_pH_p^{-1})^3$. Ayant montré que cette famille contient strictement $(H_pH_p^{-1})^2$ et $(H_p^{-1}H_p)^2$, il reste comme problèmes ouverts les questions de savoir quelles sont les positions de $(H_p + H_p^{-1})^5$ et $(H_p^{-1}H_p)^3$ dans cette hiérarchie.

Chapitre 3

Codes préfixe-suffixe composés

Ce chapitre est consacré aux codes préfixe-suffixe-composés, c'est-à-dire aux codes obtenus par composition de codes préfixes et suffixes. Sauf précision contraire, les codes que l'on considèrera seront supposés finis.

Cette étude a été motivée par l'existence d'une conjecture formulée dans [31] dans le cadre de l'étude d'un des principaux problèmes ouverts dans la théorie des codes, qui est de caractériser les codes qui sont inclus dans un code maximal fini, c'est-à-dire les codes ayant une complétion finie. Le plus petit exemple connu de code n'ayant pas de complétion finie, construit dans [30], est le code de quatre mots $\{b, ab, ba^2, a^5\}$. Il a été montré dans [31] que tout code préfixe-suffixe composé a une complétion finie, et que tout code de deux mots est préfixe-suffixe composé, et a donc une complétion finie. En relation avec la question de savoir si tout code de trois mots a une complétion finie ou non, les auteurs de [31] ont conjecturé que tout code de trois mots est préfixe-suffixe composé. Un contre-exemple à cette conjecture sera donné dans sixième section.

La première section est consacrée aux notations et aux définitions.

La deuxième section contient les outils de base qui serviront dans les sections suivantes.

Dans la troisième section, on trouve un algorithme permettant de décider si un code fini est préfixe-suffixe-composé ou non, et donnant explicitement une décomposition en codes préfixes et suffixes du code de départ, si celle-ci existe.

Dans la quatrième section, on prouve que l'algorithme présenté dans la troisième section donne la plus courte décomposition en codes préfixes et suffixes d'un code préfixe-suffixe-composé.

Dans la cinquième section, on établit que la plus courte décomposition en codes préfixes et suffixes d'un code préfixe-suffixe-composé de n mots ($n \geq 3$) contient au plus $2n-3$ codes, et que, pour tout $n \geq 3$, cette limite est atteinte, c'est-à-dire qu'il existe un code préfixe-suffixe-composé de n mots qui ne peut pas être exprimé comme la composition de moins de $2n-3$ codes préfixes et suffixes.

Enfin, la sixième section est consacrée à l'étude du cas particulier des codes de trois mots : on y donne un exemple de code de trois mots qui n'est pas préfixe-suffixe composé, réfutant ainsi la conjecture proposée par A. Restivo *et al.* dans [31]. On y propose également un code

que l'on peut conjecturer être un exemple de code de trois mots n'ayant pas de complétion finie.

3.1 Définitions et notations

On notera $\mathcal{P}$ (resp. $\mathcal{S}$) l'ensemble des codes préfixes (resp. suffixes) et $\mathcal{P}_n$ (resp. $\mathcal{S}_n$) l'ensemble des codes préfixes (resp. suffixes) de n mots.

Soit $\mathcal{F}$ et $\mathcal{G}$ deux familles de codes. La composition de $\mathcal{F}$ et $\mathcal{G}$ est $\mathcal{F} \circ \mathcal{G} = \{f \circ g, f \in \mathcal{F}, g \in \mathcal{G}, f$ et g composables $\}$. L'union de $\mathcal{F}$ et $\mathcal{G}$ est notée $\mathcal{F} + \mathcal{G}$. La famille $\mathcal{F}^n$ est l'ensemble des codes composés de n codes de $\mathcal{F}$. La famille $\mathcal{F}^*$ est $\bigcup_{n \geq 0} \mathcal{F}^n$.

Un code est *décomposable* s'il peut être exprimé comme une composition de deux codes non réduits à un alphabet. Un code est *préfixe-suffixe composé* (ou *p-s-composé*) s'il peut être exprimé comme la composition de codes préfixes et suffixes. Un code préfixe (resp. suffixe) est considéré être un code préfixe-suffixe composé. Un code qui n'est pas préfixe-suffixe composé est dit *préfixe-suffixe indécomposable* (ou *p-s-indécomposable*).

Soit C un code de A^*. Le plus petit sous-monoïde de A^* unitaire à droite (resp. à gauche) contenant C est noté $M_P(C)$ (resp. $M_S(C)$). La base de $M_P(C)$ (resp. $M_S(C)$) est appelée la *base préfixe* (resp. *base suffixe*) de C et est notée $B_P(C)$ (resp. $B_S(C)$).
Soit B la base préfixe (resp. suffixe) de C. Soit Z un alphabet tel que $Card(Z) = Card(B)$. Soit h un morphisme injectif tel que $h(Z) = B$. Le code $Q \subseteq Z^*$ tel que $C = Q \circ_h B$ est appelé le *quotient préfixe* (resp. *quotient suffixe*) de C et est noté $Q_P(C)$ (resp. $Q_S(C)$). Quand l'alphabet Z et le morphisme h sont fixés, le code Q est unique. Ces définitions donnent la relation de base suivante :

$$C = Q_P(C) \circ B_P(C) = Q_S(C) \circ B_S(C). \tag{3.1}$$

L'algorithme suivant, présenté dans [5], calcule $P^* = M_P(C)$.

Soit $(T_i)_{i \in N}$ la suite définie par :
$$\begin{cases} T_0 &= C^*, \\ T_{n+1} &= (T_n^{-1} T_n)^*. \end{cases}$$

On a $P^* = \bigcup_{n \geq 0} T_n$. Le langage $B_P(C) = P$ est facilement construit à partir de P^*.

Pour construire $Q_P(C)$ à partir de C et de $B_P(C)$, on doit d'abord :

- définir un alphabet Z tel que $Card(Z) = Card(B_P(C))$,
- définir un morphisme h tel que $h(Z) = B_P(C)$.

A partir de la définition de la composition, de l'égalité (3.1), et du fait que $B_P(C)$ est un code préfixe, on peut déduire que h est un morphisme préfixe, c'est-à-dire un morphisme injectif, et que $C = h(Q_P(C))$. Il reste juste à calculer $h^{-1}(C)$ pour obtenir $Q_P(C)$.

Exemple 3.1 *Soit* $E = \{a, aaba, abaaba\}$. *Les constructions présentées ci-dessus permettent de vérifier que :*

$$\begin{aligned} M_P(E) &= (a + ba)^*. \\ B_P(E) &= \{a, ba\}. \\ Q_P(E) &= \{x, xxy, xyxy\}. \\ M_S(E) &= (a + ab)^*. \\ B_S(E) &= \{a, ab\}. \\ Q_S(E) &= \{x, xyx, yxyx\}. \end{aligned}$$

Un code C est *fortement p-s-indécomposable* s'il est p-s-indécomposable et si $B_P(C) = B_S(C) = Alph(C)$. Notons que tout code p-s-indécomposable est la composition d'un code fortement p-s-indécomposable et d'un code (p-s-composé).

Exemple 3.2 *Le code* $C = \{b, ab, ba^2, a^5\}$ *est fortement p-s-indécomposable.*

On sait que ce code n'a pas de complétion finie (cf. [30]), et est donc p-s-indécomposable (cf. [31] ou proposition 3.4). De plus, on peut vérifier que la base préfixe et la base suffixe de C sont toutes deux égales à l'alphabet $\{a, b\}$.

Exemple 3.3 *Le code* $C = \{b, bab, bbaba, (ba)^5\}$ *est p-s-indécomposable mais n'est pas fortement p-s-indécomposable, puisque d'une part, la base préfixe de* C *est l'alphabet* $\{a, b\}$, *et d'autre part, la base et le quotient suffixe de* C *sont respectivement le code* $\{b, ba\}$ *et le code de l'exemple 3.2.*

Notons qu'un alphabet n'est pas fortement p-s-indécomposable puisqu'il est p-s-composé.

La dernière (resp. première) lettre d'un mot w de A^* est $Last(w) = A \cap (A^*)^{-1}w$ (resp. $First(w) = A \cap w(A^*)^{-1}$). Cette notion peut être étendue à un langage : l'ensemble des dernières (resp. premières) lettres d'un langage X de A^* est $Last(X) = \{Last(w), w \in X\} = A \cap (A^*)^{-1}X$ (resp. $First(X) = \{First(w), w \in X\} = A \cap X(A^*)^{-1}$).

3.2 Résultats de base

Dans cette section sont établis des résultats de base qui sont utiles pour l'étude de la décomposition d'un code p-s-composé en codes préfixes et suffixes.

Le premier d'entre eux est évident puisqu'il découle directement des définitions :

Fait 3.1 *La base préfixe (resp. suffixe) d'un code préfixe (resp. suffixe)* X *est le code* X *lui-même.*

Un résultat dual peut être déduit du fait précédent :

Fait 3.2 *Le quotient préfixe (resp. suffixe) d'un code préfixe (resp. suffixe) est un alphabet.*

Le lemme suivant est une égalité concernant le monoïde préfixe (resp. suffixe) de la composition de deux codes. Presque tous les résultats de ce chapitre sont basés sur ce lemme.

Lemme 3.1 *Soit X et Y deux codes composables. On a :*

$$M_P(X \circ Y) = M_P(B_P(X) \circ Y). \tag{3.2}$$
$$M_S(X \circ Y) = M_S(B_S(X) \circ Y). \tag{3.3}$$

Preuve Soit h le morphisme tel que $h(X) = X \circ Y$.
Soit $(T_i)_{i \in N}$ la suite définie par :

$$\begin{cases} T_0 = X^*, \\ T_{n+1} = (T_n^{-1} T_n)^*. \end{cases}$$

On peut montrer par récurrence que pour tout entier i, $h(T_i) \subseteq M_P(h(X))$.

- cas initial : $h(T_0) = h(X^*) = (h(X))^* \subseteq M_P(h(X))$.
- cas général : supposons le résultat vérifié pour $i = n$.

$$\begin{aligned} h(T_{n+1}) &= h((T_n^{-1} T_n)^*) \\ &= (h(T_n^{-1} T_n))^* \\ &\subseteq (h(T_n)^{-1} h(T_n))^* \\ &\subseteq (M_P(h(X))^{-1} M_P(h(X)))^* = (M_P(h(X)))^* \\ &= M_P(h(X)). \end{aligned}$$

Puisqu'il existe un entier i tel que $M_P(X) = T_i$, on a $h(M_P(X)) \subseteq M_P(h(X))$, et donc $M_P(h(M_P(X))) \subseteq M_P(h(X))$. L'inclusion inverse $M_P(h(X)) \subseteq M_P(h(M_P(X)))$ est évidente. On a donc $M_P(h(X)) = M_P(h(M_P(X)))$. Puisque

$$\begin{aligned} M_P(h(M_P(X))) &= M_P(h((B_P(X))^*)) \\ &= M_P((h(B_P(X)))^*) \\ &= M_P(h(B_P(X))), \end{aligned}$$

on obtient finalement $M_P(h(X)) = M_P(h(B_P(X)))$, c'est-à-dire, l'égalité (3.2). L'égalité (3.3) se prouve d'une manière symétrique. □

Des relations similaires peuvent être établies pour la base d'un monoïde préfixe ou suffixe. Voici des relations de base concernant la base préfixe ou suffixe et le quotient préfixe ou suffixe de la composition de deux codes :

Lemme 3.2 *Soit X et Y deux codes composables. On a :*

$$B_P(X \circ Y) = B_P(B_P(X) \circ Y) \tag{3.4}$$
$$= B_P(X \circ Q_P(Y)) \circ B_P(Y). \tag{3.5}$$
$$Q_P(X \circ Y) = Q_P(X \circ Q_P(Y)) \tag{3.6}$$
$$= Q_P(X) \circ Q_P(B_P(X) \circ Y). \tag{3.7}$$

et symétriquement

$$B_S(X \circ Y) = B_S(B_S(X) \circ Y) \tag{3.8}$$
$$= B_S(X \circ Q_S(Y)) \circ B_S(Y). \tag{3.9}$$
$$Q_S(X \circ Y) = Q_S(X \circ Q_S(Y)) \tag{3.10}$$
$$= Q_S(X) \circ Q_S(B_S(X) \circ Y). \tag{3.11}$$

Preuve L'égalité (3.4) est directement déduite de (3.2).

L'égalité (3.5) se prouve facilement :

$$\begin{aligned} B_P(X \circ Y) &= B_P(X \circ Q_P(Y) \circ B_P(Y)) \text{ (d'après (3.1))} \\ &= B_P(B_P(X \circ Q_P(Y)) \circ B_P(Y)) \text{ (d'après (3.4))} \\ &= B_P(X \circ Q_P(Y)) \circ B_P(Y) \text{ (d'après le fait 3.1)} \end{aligned}$$

L'égalité (3.6) se prouve de la manière suivante : Partant de (3.1), on obtient

$$X \circ Y = X \circ Q_P(Y) \circ B_P(Y),$$

et en appliquant (3.1) à $X \circ Y$ dans le membre gauche et à $X \circ Q_P(Y)$ dans le membre droit, on obtient

$$Q_P(X \circ Y) \circ B_P(X \circ Y) = Q_P(X \circ Q_P(Y)) \circ B_P(X \circ Q_P(Y)) \circ B_P(Y). \quad (3.12)$$

En appliquant (3.5) au membre droit de (3.12), on obtient

$$Q_P(X \circ Y) \circ B_P(X \circ Y) = Q_P(X \circ Q_P(Y)) \circ B_P(X \circ Y),$$

c'est-à-dire l'égalité (3.6), puisque $B_P(X \circ Y)$ est un code (préfixe).

Symétriquement, en partant de (3.1), on obtient

$$X \circ Y = Q_P(X) \circ B_P(X) \circ Y,$$

et en appliquant (3.1) à $X \circ Y$ dans le membre gauche et à $B_P(X) \circ Y$ dans le membre droit, on obtient

$$Q_P(X \circ Y) \circ B_P(X \circ Y) = Q_P(X) \circ Q_P(B_P(X) \circ Y) \circ B_P(B_P(X) \circ Y). \quad (3.13)$$

En appliquant (3.4) au membre droit de (3.13), on obtient

$$Q_P(X \circ Y) \circ B_P(X \circ Y) = Q_P(X) \circ Q_P(B_P(X) \circ Y) \circ B_P(X \circ Y),$$

c'est-à-dire l'égalité (3.7), puisque $B_P(X \circ Y)$ est un code (préfixe).
Les égalités (3.8) à (3.11) se prouvent de manière symétrique. □

Dans le cas particulier où Y est un code préfixe ou suffixe, le lemme précédent donne :

Lemme 3.3 *Soit X et Y deux codes composables. Si Y est un code préfixe, on a :*

$$B_P(X \circ Y) = B_P(X) \circ Y \quad (3.14)$$
$$Q_P(X \circ Y) = Q_P(X) \quad (3.15)$$

et symétriquement, si Y est un code suffixe, on a :

$$B_S(X \circ Y) = B_S(X) \circ Y \quad (3.16)$$
$$Q_S(X \circ Y) = Q_S(X) \quad (3.17)$$

Preuve L'égalité (3.14) découle de (3.4) et du fait 3.1 appliqué au code préfixe $B_P(X) \circ Y$. L'égalité (3.15) est facilement déduite de (3.14). Les égalités (3.16) et (3.17) se prouvent de manière symétrique. □

3.3 Décider si un code fini est préfixe-suffixe-composé ou non

Il existe un algorithme trivial pour décider si un code fini C est p-s-composé ou non : il suffit de calculer tous les codes p-s-composés ayant la taille de C et de vérifier si C est un des codes obtenus ou non. L'algorithme qui suit permet de décider plus directement si un code fini C est p-s-composé ou non. Les codes de la décomposition sont donnés de droite à gauche, c'est-à-dire, si, étant donné un code C, l'algorithme construit successivement $X_1, X_2, \ldots, X_n$, alors $C = X_n \circ X_{n-1} \circ \cdots \circ X_1$.
La décomposition d'un code préfixe-suffixe-indécomposable contient au moins un code préfixe-suffixe-indécomposable. Mais la composition d'un code et d'un code p-s-indécomposable peut être un code p-s-composé. L'exemple suivant montre plus particulièrement que même un code préfixe ou suffixe peut être le résultat de la composition d'un code préfixe ou suffixe et d'un code p-s-indécomposable . Le code p-s-indécomposable qui apparaît ici est le code de l'exemple 3.2.

Exemple 3.4 $\{x^2, y, z, t\} \circ \{b, ab, ba^2, a^5\} = \{b^2, ab, ba^2, a^5\}$.

Une condition suffisante pour un code d'être p-s-indécomposable est plus forte que la présence d'un code p-s-indécomposable dans une de ses décompositions. Le lemme suivant permet de trouver cette condition suffisante.

Lemme 3.4 *La composition d'un code p-s-indécomposable et d'un code est un code p-s-indécomposable .*

Preuve On sait que tout code p-s-indécomposable peut être écrit comme la composition d'un code fortement p-s-indécomposable et d'un code (p-s-composé). La preuve du lemme peut donc être restreinte au cas de la composition d'un code fortement p-s-indécomposable et d'un code. Nous supposons que le lemme est faux, et nous allons voir que ceci mène à une contradiction.
Soit (Z) la propriété suivante : un code p-s-composé vérifie (Z) si et seulement si il peut être écrit comme la composition d'un code fortement p-s-indécomposable et d'un code. Soit X le plus petit code p-s-composé vérifiant (Z), et donc réfutant le lemme. Soit I un code fortement p-s-indécomposable et C un code tel que $X = I \circ C$. Si le code X était préfixe (resp. suffixe) alors le code I serait également préfixe (resp. suffixe). Il existe donc un code préfixe ou suffixe $P \neq Alph(P)$ et un code p-s-composé $U \neq Alph(U)$ tels que $X = U \circ P$. Supposons que P soit préfixe (le cas "P est suffixe" est symétrique). D'après (3.4), on a $B_P(X) = B_P(I \circ C) = B_P(B_P(I) \circ C) = B_P(C)$. D'après (3.14), on a $B_P(X) = B_P(U) \circ P$, et donc $C = Q_P(C) \circ B_P(U) \circ P$. L'égalité $X = I \circ C$ peut se réécrire $U \circ P = I \circ Q_P(C) \circ B_P(U) \circ P$, et donc on a $U = I \circ Q_P(C) \circ B_P(U)$, puisque P est un code. Donc U est un code p-s-composé vérifiant (Z) et ayant une taille strictement inférieure à celle de X, ce qui contredit l'hypothèse de minimalité de la taille de X. □

L'algorithme consiste à décomposer un code C en $Q_1 \circ B_1$, où B_1 est la base préfixe ou suffixe de C, et à décomposer successivement chaque Q_i en $Q_{i+1} \circ B_{i+1}$ en utilisant la même procédure, jusqu'à ce que Q_i soit un alphabet ou un code fortement p-s-indécomposable. Soit

$B = B_i \circ \cdots \circ B_1$. Si Q_i est un alphabet, alors $C = B$ est un code p-s-composé. Si Q_i est fortement p-s-indécomposable, alors $C = Q_i \circ B$, et d'après le lemme 3.4, C est un code p-s-indécomposable.

Une décomposition dont le dernier code est un code préfixe (resp. suffixe) différent d'un alphabet est dite *ultimement préfixe* (resp. *ultimement suffixe*).

Si la base préfixe (resp. suffixe) d'un code C est un alphabet, alors le seul code préfixe (resp. suffixe) qui peut être le dernier d'une décomposition de C est un alphabet, ce qui signifie que C n'a pas de décomposition ultimement préfixe (resp. suffixe).

Des égalités (3.1) et (3.14), on peut déduire que $B_P(C) = B_P(Q_P(C)) \circ B_P(C)$, et donc que

$$B_P(Q_P(C)) \quad = \quad Alph(Q_P(C)). \tag{3.18}$$

et symétriquement

$$B_S(Q_S(C)) \quad = \quad Alph(Q_S(C)). \tag{3.19}$$

Ceci siginifie que la décomposition du quotient préfixe (resp. suffixe) de C ne peut pas être ultimement préfixe (resp. suffixe), et donc seule la décomposition ultimement suffixe (resp. préfixe) du quotient préfixe (resp. suffixe) de C a besoin d'être calculée. Les codes calculés par l'algorithme sont donc alternativement préfixes et suffixes, sauf éventuellement le dernier, qui peut être fortement p-s-indécomposable.

Voici l'algorithme en question :

```
Decompose(C)
    i ← 0
    Si B_P(C) = B_S(C) = Alph(C) alors
      Si C = Alph(C) alors
        X_1 ← C
        Ecrire "Le code est un alphabet, et est donc p-s-composé."
      Sinon
        X_1 ← C
        Ecrire "Le code est fortement p-s-indécomposable."
      FinSi
    Sinon
      Si B_P(C) = Alph(C) alors
        Ecrire "Le code n'a pas de décomposition ultimement préfixe."
      Sinon
        Ecrire "Décomposition ultimement préfixe : "
        Decompose_pref(C)
      FinSi
      Si B_S(C) = Alph(C) alors
        Ecrire "Le code n'a pas de décomposition ultimement suffixe."
      Sinon
        Ecrire "Décomposition ultimement suffixe : "
        Decompose_suff(C)
      FinSi
```

```
    FinSi
Fin_Decompose(C)

Decompose_pref(C)
    Si B_P(C) ≠ Alph(C) alors
      i ← i + 1; X_i ← B_P(C)
      Decompose_suff(Q_P(C))
    Sinon
      Si C = Alph(C) alors
        Ecrire "Le code est p-s-composé."
      Sinon
        i ← i + 1; X_i ← C
        Ecrire "Le code est p-s-indécomposable."
      FinSi
    FinSi
Fin_Decompose_pref(C)

Decompose_suff(C)
    Si B_S(C) ≠ Alph(C) alors
      i ← i + 1; X_i ← B_S(C)
      Decompose_pref(Q_S(C))
    Sinon
      Si C = Alph(C) alors
        Ecrire "Le code est p-s-composé."
      Sinon
        i ← i + 1; X_i ← C
        Ecrire "Le code est p-s-indécomposable."
      FinSi
    FinSi
Fin_Decompose_suff(C)
```

L'exemple suivant montre qu'un code p-s-composé peut avoir à la fois une décomposition ultimement préfixe et une décomposition ultimement suffixe :

Exemple 3.5 *Le code* $C = \{a, aaba, abaaba\}$ *a pour base préfixe le code* $\{a, ba\}$ *et pour base suffixe le code* $\{a, ab\}$. *L'application de l'algorithme conduit aux deux décompositions suivantes :*

$$C = \{x, xxy, xyxy\} \circ \{a, ba\}.$$
$$C = \{z, zt, tt\} \circ \{x, yx\} \circ \{a, ab\}.$$

3.4 La décomposition la plus courte en codes préfixes et suffixes d'un code préfixe-suffixe-composé

Dans cette section, nous prouvons que l'algorithme présenté dans la section 3.3 donne la décomposition la plus courte en codes préfixes et suffixes d'un code p-s-composé. Pour le prouver, nous avons besoin de ces résultats préliminaires :

Lemme 3.5 *Soit S un code suffixe et P un code préfixe. La décomposition ultimement préfixe du code $C = S \circ P$ est $C = Q_P(C) \circ B_P(C)$, ou $Q_P(C)$ est un code suffixe.*

Preuve D'après l'égalité $S = Q_P(S) \circ B_P(S)$, $Q_P(S)$ est un code suffixe. D'après (3.15), $Q_P(C) = Q_P(S \circ P) = Q_P(S)$, et donc $Q_P(C)$ est un code suffixe. □

Lemme 3.6 *Soit S un code suffixe et P un code préfixe. La décomposition ultimement suffixe du code $C = S \circ P$ est $C = Q_P(Q_S(C)) \circ B_P(Q_S(C)) \circ B_S(C)$, où $Q_P(Q_S(C))$ est un code suffixe, et $Q_S(C)$ est un code de $\mathcal{S} \circ \mathcal{P}$.*

Preuve De $C = Q_S(C) \circ B_S(C)$ et (3.7), on déduit

$$Q_P(C) = Q_P(Q_S(C)) \circ Q_P(B_P(Q_S(C)) \circ B_S(C)). \tag{3.20}$$

D'après le lemme 3.5, $Q_P(C)$ est un code suffixe, et donc, d'après l'égalité précédente, $Q_P(Q_S(C))$ est également un code suffixe. Puisque $Q_S(C) = Q_P(Q_S(C)) \circ B_P(Q_S(C))$, $Q_S(C)$ est un code de $\mathcal{S} \circ \mathcal{P}$. □

Le cas général peut maintenant être prouvé :

Lemme 3.7 *Pour tout entier $n \geq 1$, si C est un code de $(\mathcal{S} + \mathcal{P})^n \circ \mathcal{P}$, alors*

$$Q_P(C) \in (\mathcal{S} + \mathcal{P})^{n-1} \circ \mathcal{S}, \tag{3.21}$$
$$Q_S(C) \in (\mathcal{S} + \mathcal{P})^n \circ \mathcal{P}, \tag{3.22}$$

et symétriquement, si C est un code de $(\mathcal{S} + \mathcal{P})^n \circ \mathcal{S}$, alors

$$Q_S(C) \in (\mathcal{S} + \mathcal{P})^{n-1} \circ \mathcal{P}, \tag{3.23}$$
$$Q_P(C) \in (\mathcal{S} + \mathcal{P})^n \circ \mathcal{S}. \tag{3.24}$$

Preuve Les lemmes 3.5 et 3.6 permettent de prouver le résultat pour $n = 1$. Supposons que le résultat soit vérifié jusqu'à $n = i - 1$. Soit C un code de $(\mathcal{S} + \mathcal{P})^i \circ \mathcal{P}$. S'il existe un entier $j < i$ tel que C est un code de $(\mathcal{S} + \mathcal{P})^j \circ \mathcal{P}$, alors le résultat est vérifié par hypothèse de récurrence. On peut donc supposer que $C = X \circ P$, où X est un code de $(\mathcal{S} + \mathcal{P})^{i-1} \circ \mathcal{S}$ et P est un code préfixe. D'après (3.15), on a $Q_P(C) = Q_P(X)$. Par hypothèse de récurrence, $Q_P(X)$ est un code de $(\mathcal{S} + \mathcal{P})^{i-1} \circ \mathcal{S}$, et donc (3.21) est vérifiée pour C. D'après (3.11), on a $Q_S(C) = Q_S(X) \circ Q_S(B_S(X) \circ P)$. Par hypothèse de récurrence, $Q_S(X)$ est un code de $(\mathcal{S} + \mathcal{P})^{i-2} \circ \mathcal{P}$. D'après le lemme 3.6, $Q_S(B_S(X) \circ P)$ est un code de $\mathcal{S} \circ \mathcal{P}$. Finalement, $Q_S(C)$ est un code de $(\mathcal{S}+\mathcal{P})^{i-2} \circ \mathcal{P} \circ \mathcal{S} \circ \mathcal{P}$, d'où (3.22). Un raisonnement symétrique permet de prouver (3.23) et (3.24). □

Il n'est pas difficile de prouver, en utilisant le lemme 3.7, que si C est un code obtenu en composant n codes préfixes et suffixes dont le dernier est préfixe (resp. suffixe), alors la décomposition ultimement préfixe (resp. suffixe) de C contient au plus n codes préfixes et suffixes, ce qui signifie que l'algorithme de la section 3.3 donne la plus courte décomposition en codes préfixes et suffixes.

3.5 Codes de n mots

Nous allons maintenant étudier les relations existant entre le cardinal d'un code p-s-composé et le nombre de codes apparaissant dans sa (ses) décomposition(s) en codes préfixes et suffixes.

Rapellons d'abord un résultat prouvé dans [5] :
Soit C un code et B la base préfixe (resp. suffixe) de C. Tout mot de B est facteur droit (resp. gauche) d'au moins un mot de C, et donc :

$$Card(B) \leq Card(C). \tag{3.25}$$

En particulier :

Fait 3.3 *Soit C un code. Si $B_P(C) = Alph(C)$ alors $Last(C) = Alph(C)$. Symétriquement, si $B_S(C) = Alph(C)$ alors $First(C) = Alph(C)$.*

Si un code C contient autant de mots que sa base préfixe (resp. suffixe), alors la décomposition ultimement préfixe (resp. suffixe) de C contient au plus deux codes :

Lemme 3.8 *Soit C un code de n mots, B la base préfixe (resp. suffixe) de C et Q le quotient préfixe (resp suffixe) de C. Si $Card(B) = Card(C) = n$ alors Q est un code suffixe (resp. préfixe) de n mots, et donc C est un code de $\mathcal{S}_n \circ \mathcal{P}_n$ (resp. $\mathcal{P}_n \circ \mathcal{S}_n$).*

Preuve Soit C un code de n mots. Soit $B = B_P(C)$ et $Q = Q_P(C)$. On sait que $Card(Q) = Card(C) = n$ et $Card(Alph(Q)) = Card(B)$. D'après (3.18), on a $B_P(Q) = Alph(Q)$. D'après le fait 3.3, on a $Last(Q) = Alph(Q)$, et donc $Card(Last(Q)) = Card(B)$. Si $Card(B) = n$, alors $Card(Last(Q)) = Card(Q) = n$, et donc, Q est un code suffixe. Le résultat symétrique se prouve de la même manière. □

Une conséquence du lemme 3.8 est que la décomposition ultimement préfixe (resp. suffixe) d'un code de deux mots contient au plus deux codes :

Proposition 3.1 *La famille $\mathcal{C}_2$ des codes de deux mots est égale à $\mathcal{C}_2 = \mathcal{P}_2 \circ \mathcal{S}_2 = \mathcal{S}_2 \circ \mathcal{P}_2$.*

Preuve Soit C un code de deux mots et B la base préfixe (resp. suffixe) de C. D'après (3.25), B contient au plus deux mots. Si B ne contient qu'un mot α, alors C contient deux puissances de ce mot α, ce qui contredit l'hypothèse que C est un code. Le langage B contient donc exactement deux mots, et d'après le lemme 3.8, on conclut que C est un code $\mathcal{S}_2 \circ \mathcal{P}_2$ (resp. $\mathcal{P}_2 \circ \mathcal{S}_2$). □

Considérons l'algorithme présenté dans la section 3.3. Soit C le code initial à décomposer. Soit $X_1, \ldots, X_i$ les i codes obtenus aux i premières étapes de l'algorithme. Soit Y le code courant à décomposer après la $i$ème étape. On a $C = Y \circ X_i \cdots X_1$. Supposons que ni X_{i+1}, ni X_{i+2} ni X_{i+3} ne soit un alphabet. Alors $X_{i+1} = B_P(Y)$, $X_{i+2} = B_S(Q_P(Y))$ et $X_{i+3} = B_P(Q_S(Q_P(Y)))$. Le lemme 3.9 permet de montrer que pour tout entier i, $Card(X_{i+1}) \geq Card(X_i)$. Le lemme 3.10 montre qu'au plus deux codes consécutifs calculés par l'algorithme peuvent avoir le même cardinal, c'est-à-dire que pour tout entier i, si $Card(X_i) = Card(X_{i+1})$ alors $Card(X_{i+2}) > Card(X_{i+1})$.

Lemme 3.9 *Soit C un code. On a :*

$$Card(B_S(Q_P(C))) \geq Card(B_P(C)), \tag{3.26}$$
$$Card(B_P(Q_S(C))) \geq Card(B_S(C)). \tag{3.27}$$

Preuve Soit C un code, $X = Q_P(C)$ et $n = Card(B_P(C))$.
Par construction, $Card(Alph(X)) = n$. De $X \subseteq (B_S(X))^+$, on déduit que pour chaque lettre a de $Last(X)$, il existe au moins un mot w_a de $B_S(X)$ tel que $Last(w_a) = a$, et donc $Card(B_S(X)) \geq Card(Last(X))$. D'après (3.18), on a $B_P(X) = Alph(X)$. D'après le fait 3.3, on a $Last(X) = Alph(X)$, d'où $Card(B_S(X)) \geq Card(Alph(X))$, c'est-à-dire, $Card(B_S(X)) \geq n$. L'inégalité (3.27) se prouve de manière symétrique. □

Lemme 3.10 *Soit C un code p-s-composé ayant au moins trois codes dans sa décomposition en codes préfixes et suffixes. Si la base préfixe de C est un code de m mots, alors*

- $Card(B_S(Q_P(C))) \geq m$,
- *si* $Card(B_S(Q_P(C))) = m$ *alors* $Card(B_P(Q_S(Q_P(C)))) > m$.

Symétriquement, si la base suffixe de C est un code de m mots, alors

- $Card(B_P(Q_S(C))) \geq m$,
- *si* $Card(B_P(Q_S(C))) = m$ *alors* $Card(B_S(Q_P(Q_S(C)))) > m$.

Preuve Soit C un code p-s-composé de n mots ayant au moins trois codes dans sa décomposition en codes préfixes et suffixes. On suppose que $B_P(C) \neq Alph(C)$ et $Card(B_P(C)) = m$. Soit $X = Q_P(C)$, $S = B_S(X)$ et $Q = B_P(Q_S(X))$. Puisque C est composé d'au moins trois codes, ni X, ni S, ni Q ne peut être un alphabet. D'après le lemme 3.9, on a $Card(Q) \geq Card(S) \geq m$. D'après (3.4), $B_P(X) = B_P(B_P(Q_P(Q_S(X)) \circ Q \circ S)$. D'après (3.18), $B_P(Q_P(Q_S(X)) = Alph(Q_P(Q_S(X)))$. D'après (3.18), $B_P(X) = Alph(X)$. On obtient finalement $B_P(Q \circ S) = Alph(X)$. Par construction, $Card(Alph(X)) = m$, et donc $Card(B_P(Q \circ S)) = m$. Le cas $Card(Q) = m$ ne peut pas se produire. En effet, si on suppose que $Card(Q) = m$ alors $Card(Q \circ S) = m$ et d'après le lemme 3.8, $Q \circ S$ est un code suffixe, ce qui siginifie que soit $Q = Alph(Q)$ soit $S \neq B_S(X)$, ce qui est contraire à l'hypothèse. Le résultat symétrique se prouve de la même manière. □

La structure générale de la décomposition d'un code p-s-composé donnée par l'algorithme de la section 3.3 peut être écrite comme ceci :

Lemme 3.11 *Soit T_i la famille $((\mathcal{P}_i \circ \mathcal{S}_i) + (\mathcal{S}_i \circ \mathcal{P}_i))$, c'est-à-dire la famille des codes obtenus par la composition d'un code préfixe de i mots et d'un code suffixe de i mots, ou la composition d'un code suffixe de i mots et d'un code préfixe de i mots. Si C est un code p-s-composé de n mots tel que $Card(B_P(C)) = m < n$, alors C est dans la famille*

$$T_n \circ T_{i_1} \circ \cdots \circ T_{i_k} \circ (\mathcal{S}_m \circ \mathcal{P}_m),$$

où $n > i_1 > i_2 > \cdots > i_k > m$,
et symétriquement,

si C est un code p-s-composé de n mots tel que $Card(B_S(C)) = m < n$, alors C est dans la famille

$$T_n \circ T_{i_1} \circ \cdots \circ T_{i_k} \circ (\mathcal{P}_m \circ \mathcal{S}_m),$$

où $n > i_1 > i_2 > \cdots > i_k > m$.

Preuve Pour tout entier i, on a clairement

$$\mathcal{P}_i \subseteq \mathcal{S}_i \circ \mathcal{P}_i, \tag{3.28}$$

$$\mathcal{S}_i \subseteq \mathcal{P}_i \circ \mathcal{S}_i. \tag{3.29}$$

puisqu'un alphabet de i lettres est un code préfixe (resp suffixe) de i mots. La preuve peut être faite par récurrence : le cas $n = m$ est traité par le lemme 3.8. Le cas général est facilement traité en utilisant (3.28), (3.29), le lemme 3.10 et l'algorithme de la section 3.3. □

Proposition 3.2 *Soit C un code p-s-composé de n mots ($n \geq 3$). La plus courte décomposition de C en codes préfixes et suffixes contient au plus $2n-3$ codes. Cette limite est atteinte pour tout $n \geq 3$.*

Preuve Le lemme 3.11 montre qu'un code p-s-composé de n mots dont la base préfixe est un code m mots peut être décomposé en au plus $2(n-m+1)$ codes préfixes et suffixes. Le maximum de cette valeur est atteint pour $m = 2$ (le cas $m = 1$ ne peut pas se produire) et devrait être $2n-2$. Mais cette valeur peut être légèrement améliorée en utilisant la proposition 3.1. En effet, pour tout entier i, on a $\mathcal{P}_i \circ \mathcal{S}_2 \circ \mathcal{P}_2 = \mathcal{P}_i \circ \mathcal{P}_2 \circ \mathcal{S}_2 = \mathcal{P}_i \circ \mathcal{S}_2$ et symétriquement, $\mathcal{S}_i \circ \mathcal{P}_2 \circ \mathcal{S}_2 = \mathcal{S}_i \circ \mathcal{P}_2$. Finalement, on voit que $2n-3$ codes suffisent. Pour tout entier $n \geq 3$, il existe des codes qui atteignent cette limite, c'est-à-dire qui ne peuvent pas être décomposés en moins de $2n-3$ codes préfixes et suffixes.

Soit $A_{i,j}$ le code défini sur l'alphabet $\{a_1, \ldots, a_j\}$ par

$$A_{i,j} = \{a_1, \cdots, a_{j-1}, u_1, \cdots, u_{i-j}, a_1^{-1}u_{i-j+1}\}$$

où $u_1 = a_1 a_j a_1$ et $u_k = u_{k-1}a_1^{-1}u_{k-1}a_1$ pour $k > 1$.

Soit $X_{i,j}$ le code défini sur l'alphabet $\{x_1, \ldots, x_j\}$ par

$$X_{i,j} = \{x_1, \cdots, x_{j-1}, x_1x_j, u_1, \cdots, u_{i-j}, x_1^{-1}u_{i-j+1}\}$$

où $u_1 = x_1x_jx_jx_1$ et $u_k = u_{k-1}x_1^{-1}u_{k-1}x_1$ pour $k > 1$.

Un calcul facile nous mène à :

$$\begin{aligned} B_P(A_{i,j}) &= \{a_1, \ldots, a_{j-1}, a_ja_1,\} \\ B_S(A_{i,j}) &= \{a_1, \ldots, a_j,\} \\ B_S(X_{i,j}) &= \{x_1, \ldots, x_{j-1}, x_1x_j, x_jx_j,\} \\ B_P(X_{i,j}) &= \{x_1, \ldots, x_j.\} \end{aligned}$$

De plus, on a $A_{i,j} = X_{i-1,j} \circ B_P(A_{i,j})$ et $X_{i,j} = A_{i+1,j+1} \circ B_S(X_{i,j})$.

Puisque l'algorithme de la section 3.3 donne la plus courte décomposition en codes préfixes et suffixes, le code $A_{n,2}$ ne peut pas être décomposé en moins de $2n-3$ codes préfixes et suffixes. □

3.6 Codes de trois mots

Le probleme de décider si un code fini donné peut être plongé dans un code maximal fini est toujours ouvert. Le résultat suivant, établi dans [31], permet de donner une réponse partielle à ce problème :

Proposition 3.3 *Un code fini obtenu par composition de codes ayant une complétion finie a une complétion finie.*

Preuve Soit $X = X_1 \circ_{h_1} \cdots \circ_{h_{n-1}} X_n$ la composition de n codes ayant une complétion finie. Soit Y_i une complétion finie de X_i, pour $i \in \{1, \ldots, n\}$. Les Y_i ne peuvent pas être composés puisque la condition $Card(Alph(Y_i)) = Card(Y_{i+1})$ n'est pas vérifiée. Pour obtenir cette condition, il suffit de prendre $Z_n = Y_n$, et pour chaque $i < n$, de construire un code complet Z_i contenant Y_i et défini sur un alphabet A_i tel que $Card(A_i) = Card(Z_{i+1})$ (cela signifie que Z_{i+1} doit être construit avant Z_i). Pour chaque i de $\{1, \ldots, n-1\}$, on construit un morphisme h'_i tel que $h'_i(Alph(Z_i)) = Z_{i+1}$ et tel que toute lettre de $Alph(X_i)$ a même image par h'_i et h_i. Par construction, $Z_1 \circ_{h'_1} \cdots \circ_{h'_{n-1}} Z_n$ est une complétion finie de X.
Pour étendre un code complet C défini sur l'alphabet $\{a_1, \ldots, a_n\}$ à un code complet C' contenant C et défini sur l'alphabet $\{a_1, \ldots, a_n, \ldots, a_p\}$, on peut par exemple prendre $C' = s(C)$, où s est la substitution finie définie par $s(a_i) = a_i$, pour $i \in \{1, \ldots, n-1\}$ et $s(a_n) = a_n + a_{n+1} + \cdots + a_p$. Il n'est pas difficile de vérifier que si C est complet et est un code, alors $s(C)$ est complet et est un code également. □

Puisque tout code préfixe (resp. suffixe) a une complétion finie (cf. [31]), on obtient :

Proposition 3.4 *[31] Tout code fini préfixe-suffixe composé a une complétion finie.*

Le code de quatre mots $\{b, ab, ba^2, a^5\}$, construit dans [30], est le plus petit exemple connu de code n'ayant pas de complétion finie. La proposition 3.1 montre que tout code de deux mots est préfixe-suffixe composé (Une autre preuve est donnée dans [31]). Donc, d'après la proposition 3.3, tout code de deux mots a une complétion finie. La question reste ouverte pour les codes de trois mots. Les auteurs de [31] ont implicitement proposé la conjecture suivante :

Tout code de trois mots a une complétion finie. (C1)

En fait, ils ont explicitement conjecturé que

Tout code de trois mots est préfixe-suffixe composé. (C2)

D'après la proposition 3.4, la vérité de (C2) implique la vérité de (C1).

Pour trouver le plus petit contre-exemple à la conjecture (C2), il n'est pas utile de chercher des codes de trois mots p-s-indécomposables définis sur un alphabet contenant plus de 2 lettres. En effet, un code de trois mots p-s-indécomposable défini sur un alphabet de k lettres contient dans sa (ses) décomposition(s) donnée(s) par l'algorithme de la section 3.3 un code de trois mots p-s-indécomposable plus petit et défini sur un alphabet a deux lettres :

Proposition 3.5 *Tout code de trois mots p-s-indécomposable défini sur un alphabet à k lettres est la composition d'un code de trois mots p-s-indécomposable défini sur un alphabet à deux lettres et d'un code de deux mots (préfixe ou suffixe) défini sur un alphabet à k lettres.*

Preuve Soit C un code de trois mots p-s-indécomposable tel que $Card(Alph(C)) = k > 2$. Soit B la base préfixe (resp. suffixe) de C. Le code B contient au plus trois mots. Si B ne contient qu'un mot, alors C n'est pas un code, ce qui contredit l'hypothèse. Si B est un code de trois mots, alors, d'après le lemme 3.8, C est un code de $\mathcal{S}_3 \circ \mathcal{P}_3$ (resp. $\mathcal{P}_3 \circ \mathcal{S}_3$) et donc C n'est pas p-s-indécomposable, ce qui contredit l'hypothèse. Donc B est un code de deux mots défini sur un alphabet à k lettres. Soit Q le quotient préfixe (resp. suffixe) de C. Si Q était p-s-composé, alors C, égal à $Q \circ B$, le serait aussi. Donc Q est un code de trois mots p-s-indécomposable défini sur un alphabet à deux lettres. □

Le code suivant est le plus petit contre-exemple à la conjecture (C2).

Proposition 3.6 *Le code de trois mots $\{a, aba, baba^2b\}$ n'est pas préfixe-suffixe composé.*

Preuve Soit $C = \{a, aba, baba^2b\}$. Le code C n'est ni préfixe ni suffixe et on peut vérifier, en utilisant la méthode présentée dans la section 3.1, que la base préfixe et la base suffixe de C sont toutes deux égales à l'alphabet $\{a, b\}$. □

Le code de la précédente proposition est l'élément particulier $L_{0,\varepsilon}$ de la famille suivante de codes de trois mots qui ne sont pas préfixe-suffixe composés. En effet, on peut étendre la preuve précédente pour vérifier que pour tout entier n et tout mot u de $(a + ab)^*$, le langage

$$L_{n,u} = \{a, aba, (ba)^{n+2}uab\}$$

est un code dont la base préfixe et la base suffixe sont toutes deux égales à l'alphabet $\{a, b\}$.

La réfutation de la conjecture (C2) ne donne pas une réfutation de la conjecture (C1). En effet, un code inclus dans un code maximal p-s-composé est p-s-composé, mais la réciproque de la proposition 3.4 est fausse : il existe des codes maximaux (p-s-)indécomposables qui ne sont ni préfixes ni suffixes. Le premier exemple d'un tel code a été donné dans [9]. Une famille de tels codes a été construite dans [7]. Le plus petit exemple d'un tel code, donné dans [13], est

$$\{b, aba, ab^2, a^4, ba^2b, aba^3, ab^2a^2, ba^3b, ba^2ba^2, ba^3ba^2, ba^6\}.$$

Le code $L_{0,\varepsilon}$ a une complétion finie

$$M_{0,\varepsilon} = \{a, aba, b^2, b^3a, baba^3, baba^3ba, baba^2b, baba^2b^2a, babab^2, babab^3a, bababa ba\}.$$

Plus généralement, chaque $L_{n,u}$ a une complétion finie. Soit $M_{n,u}$ le code

$$(a + b^2 + (ba)^{n+2}(a^2 + ab + b^2)(a + b)^{|u|})(\varepsilon + ba + (ba)^2 + \cdots + (ba)^{n+1}) + (ba)^{2n+4}.$$

Il n'est pas difficile de vérifier que $M_{n,u}$ est un code maximal fini contenant $L_{n,u}$.

La plupart des codes de trois mots p-s-indécomposables que nous avons pu construire (pas seulement les $L_{n,u}$) ont une complétion finie. Mais il existe des codes de trois mots pour lesquels on ne sait pas encore si ils ont une complétion finie ou non. L'un d'entre eux est le code palindrome $\{a, aba, ba^2bababa^2b\}$. On peut conjecturer que :

Conjecture 3.1 *Le code de trois mots $\{a, aba, ba^2bababa^2b\}$ n'a pas de complétion finie.*

Chapitre 4

Codes maximaux

Ce chapitre a pour but l'étude de la maximalité de la base de l'intersection de deux monoïdes libres engendrés par des codes rationnels maximaux.

Cette question a été posée par Roman König, qui étudiait le problème général suivant :

Soit $\mathcal{C}$ l'ensemble de tous les codes définis sur des alphabets finis quelconques. Soit la relation d'ordre définie sur $\mathcal{C}$ par $X \leq Y \Longleftrightarrow X^* \subseteq Y^*$. Pour tout code X défini sur un alphabet fini A et tout code Y défini sur un alphabet fini B, on définit alors $X \vee Y$ comme la base du plus petit sous-monoïde libre de $(A \cup B)^*$ contenant $X \cup Y$ et $X \wedge Y$ comme la base de $X^* \cap Y^*$. L'ensemble $\mathcal{C}$ muni des opérations $\vee$ et $\wedge$ est un treillis. On peut se demander s'il en est de même pour l'ensemble des codes rationnels maximaux de $\mathcal{C}$. Il a été montré ([26]) que $X \vee Y$ n'est pas un code maximal en général, mais est maximal si X et Y sont définis sur le même alphabet. Il restait donc le problème de la maximalité de $X \wedge Y$, la base de $X^* \cap Y^*$.

La maximalité d'un code étant définie par rapport à un alphabet, il est important de préciser certains points concernant les alphabets des différents langages considérés dans le problème. Considérons deux codes rationnels maximaux X et Y et appelons A l'alphabet de X, B l'alphabet de Y, Z la base de $X^* \cap Y^*$ et C l'alphabet de Z.
Les alphabets C et $A \cap B$ peuvent coincider. Par exemple, si X et Y sont des codes finis maximaux, ils contiennent tous deux une puissance de chacune des lettres de leur alphabet respectif, et Z contient donc une puissance de chaque lettre de $A \cap B$, ce qui signifie que C contient chaque lettre de $A \cap B$. L'inclusion de C dans $A \cap B$ étant évidente, on obtient bien l'égalité entre C et $A \cap B$. Mais il existe des cas où C ne contient pas toutes les lettres de $A \cap B$, comme le montrent les exemples suivants :

Exemple 4.1

En prenant	$X = a^*b$	*sur* $A = \{a, b\}$
et	$Y = a$	*sur* $B = \{a\}$
on obtient	$X^* \cap Y^* = \{\varepsilon\}$, *et donc* $Z = \emptyset$,	
ce qui donne	$A \cap B = \{a\} \neq C = \emptyset$.	

Dans l'exemple ci-dessus, A et B sont des alphabets différents. Même en prenant deux codes X et Y définis sur le même alphabet, on peut obtenir un exemple du même type :

Exemple 4.2

En prenant	$X = a^*b$	*sur* $A = \{a, b\}$
et	$Y = b^*a$	*sur* $B = \{a, b\}$
on obtient	$X^* \cap Y^* = \{\varepsilon\}$, *et donc* $Z = \emptyset$,	
ce qui donne	$A \cap B = \{a, b\} \neq C = \emptyset$.	

Enfin, voici un exemple où l'alphabet de Z n'est pas vide :

Exemple 4.3

En prenant	$X = a + ba + bb$	*sur* $A = \{a, b\}$
et	$Y = (a + b)a^*b$	*sur* $B = \{a, b\}$
on obtient	$X^* \cap Y^* = (bb)^*$, *et donc* $Z = bb$,	
ce qui donne	$A \cap B = \{a, b\} \neq C = \{b\}$.	

Un code est maximal s'il est maximal sur son alphabet. Un code est maximal sur A s'il est maximal et si son alphabet est A. Ainsi, dans chacun des exemples 4.1 à 4.3, la base de $X^* \cap Y^*$ est un code maximal, mais n'est pas un code maximal sur $A \cap B$, puisque C et $A \cap B$ ne coincident pas.

On peut formuler de la manière suivante la conjecture que Roman König a proposée :

Conjecture 4.1 *Si X et Y sont deux codes rationnels maximaux, alors la base de $X^* \cap Y^*$ est un code rationnel maximal.*

On connait déjà le résultat suivant, qui indique que la conjecture est vraie si parmi les deux codes rationnels X et Y, définis sur le même alphabet, on trouve au moins un code bipréfixe :

Proposition 4.1 *([4, p. 259, ex. 6.4],[26]) Soit X un code coupant bipréfixe maximal sur A. Pour tout code coupant Y maximal sur A, la base de $X^* \cap Y^*$ est un code coupant maximal sur A.*

Nous montrons dans la deuxième section que la conjecture 4.1 est par contre fausse dans le cas général, en donnant quelques contre-exemples, et en présentant une méthode, basée sur les automates, permettant de construire facilement des couples (X, Y) de codes préfixes rationnels maximaux qui réfutent la conjecture.

Inversement, nous proposons dans la troisième section une méthode qui, pour tout code (préfixe) rationnel Z, permet de construire, de façon effective, deux codes (préfixes) rationnels maximaux X et Y tels que Z soit la base de $X^* \cap Y^*$.

Enfin, nous donnons dans la quatrième section une condition nécessaire et suffisante pour que la conjecture 4.1 soit vraie. Cette condition indique en particulier que pour que deux codes maximaux rationnels vérifient la conjecture, il suffit qu'au moins l'un des deux soit un code préfixe et qu'au moins l'un des deux soit un code suffixe.

4.1 Résultats précédents

Rappelons les résultats suivants, concernant les codes préfixes :

Proposition 4.2 *([4, p. 102, prop. 3.9]) Soit X un code préfixe coupant défini sur l'alphabet A. Les conditions suivantes sont équivalentes :*

(i) *X est préfixe maximal.*
(ii) *l'automate minimal de X^* est complet.*
(iii) *tous les états de l'automate minimal de X^* sont récurrents.*
(iv) *l'état initial de l'automate minimal de X^* est récurrent.*

Il est clair que l'intersection de deux monoïdes unitaires à droite est un monoïde unitaire à droite. On déduit alors de la proposition 0.1 que :

Proposition 4.3 *Si X et Y sont deux codes préfixes, alors la base de $X^* \cap Y^*$ est un code préfixe.*

Proposition 4.4 *([4, p. 95, prop. 2.2]) Soit P un sous-ensemble de A^*. Les conditions suivantes sont équivalentes :*

(i) *P est un sous-monoïde unitaire à droite.*
(ii) *l'automate minimal de P a pour unique état final l'état initial.*

Proposition 4.5 *([4, p. 101, th. 3.7]) Soit X un code coupant de A^+. Les conditions suivantes sont équivalentes :*

(i) *X est préfixe et maximal sur A.*
(ii) *X est complet à droite dans A^*.*

Proposition 4.6 *([4, p. 264, prop. 2.1]) Soit X un code préfixe coupant maximal sur A et $\mathcal{A} = \{Q, A, \delta, \{q_0\}, \{q_0\}\}$ un automate déterministe émondé reconnaissant X^*. Les conditions suivantes sont équivalentes :*

(i) *X est un code bipréfixe maximal sur A.*
(ii) *Pour tout mot w de A^*, $\delta(Q, w)$ contient q_0.*
(iii) *Pour tout mot w de A^*, si $\delta(q, w) = \delta(q_0, w)$ alors $q = q_0$.*

4.2 Construction de contre-exemples

Dans de nombreux problèmes concernant les codes, il est plus facile de commencer l'étude en se restreignant au cas particulier des codes préfixes. On peut ainsi utiliser avec profit les propriétés particulières, rappelées dans la section 4.1, que possèdent les automates représentant de tels codes. Ce n'est donc pas un hasard si, dans le premier contre-exemple à la conjecture 4.1 qui a été trouvé, les deux codes rationnels maximaux X et Y sont deux codes préfixes. Ce contre-exemple est :

Exemple 4.4 *Soit les deux codes maximaux*

$$X = a^2 + ab + ba^*b \text{ et } Y = ba + b^2 + ab^*a.$$

La base de $X^* \cap Y^*$ est le code non maximal

$$a^2 + b^2.$$

Dans l'exemple ci-dessus, X et Y sont des codes infinis. Même dans le cas où X et Y sont deux codes préfixes maximaux finis, la conjecture 4.1 est encore fausse, comme le montre l'exemple suivant :

Exemple 4.5 *Soit les deux codes maximaux finis*

$$X = (a+b)(a+ba+b^2) \text{ et } Y = (a+ba+b^2)(a+b).$$

La base de $X^* \cap Y^*$ est le code non maximal

$$b^2a + b^3 + a(b^2a + b^3)^*a.$$

Il est possible, à partir de tout code rationnel préfixe non bipréfixe X maximal sur A, de construire un code rationnel préfixe non bipréfixe Y maximal sur A, tel que Z, la base de $X^* \cap Y^*$, ne soit pas un code maximal sur A. L'alphabet de Z n'étant pas nécessairement égal à A, il est malgré tout possible que Z soit maximal. Cette construction est décrite dans l'énoncé du lemme 4.1. La proposition 4.6 permet de justifier l'existence de l'état q_i et du mot u utilisés dans le lemme 4.1.

Lemme 4.1 *Soit X un code rationnel préfixe, non bipréfixe et maximal sur A, et $\mathcal{A}_X = \{Q, A, \delta, \{q_0\}, \{q_0\}\}$ l'automate déterministe minimal reconnaissant X^*. Soit q_i un état de $Q \setminus \{q_0\}$ et u un mot de A^* tels que $\delta(q_0, u) = \delta(q_i, u)$. Soit Y^* le langage reconnu par l'automate $\mathcal{A}_Y = \{Q, A, \delta, \{q_i\}, \{q_i\}\}$. Soit Z la base de $X^* \cap Y^*$. Si l'alphabet de Z est A, alors Z est un code préfixe non maximal.*

Preuve Les automates reconnaissant X^* et Y^* ne différant que par leur état initial et leur état final, on peut raisonner sur leur fonction de transition δ qui leur est commune. Ainsi, $X^* \cap Y^*$ est l'ensemble des mots w de A^* tels que $\delta(q_0, w) = q_0$ et $\delta(q_i, w) = q_i$. De $\delta(q_0, u) = \delta(q_i, u)$, on déduit que pour tout mot v de A^*, on a $\delta(q_0, uv) = \delta(q_i, uv)$. Puisque $q_i \neq q_0$, le mot uv ne fait partie de $X^* \cap Y^*$ pour aucun mot v de A^*, et donc, Z, la base de $X^* \cap Y^*$, est un code qui n'est pas complet à droite. Or, d'après la proposition 4.3, Z est un code préfixe. Puisque Z est un code rationnel, donc coupant, d'après la proposition 4.5, si l'alphabet de Z est A, alors Z n'est pas un code maximal. □

Il existe des cas assez généraux où l'on peut affirmer que si X et Y sont deux codes maximaux sur A, alors l'alphabet de (la base de) $X^* \cap Y^*$ est encore A. Il suffit par exemple que X et Y contiennent tous deux une puissance de chacune des lettres de A, ce qui est le cas lorsque X et Y sont finis. La construction du lemme 4.1 permet donc de généraliser l'exemple 4.5 de la manière suivante :

Proposition 4.7 *Soit P_1 et P_2 deux codes préfixes finis maximaux sur A tels que $(P_1P_2)^* \cap (P_2P_1)^*P_2 \neq \emptyset$. Soit $X = P_1P_2$ et $Y = P_2P_1$. Les langages X et Y sont deux codes préfixes finis maximaux sur A, et la base de $X^* \cap Y^*$ est un code préfixe de A^* non maximal.*

Preuve Le produit de deux codes préfixes est non ambigu. Si P_1 et P_2 sont deux codes préfixes finis maximaux sur A, alors les langages $X = P_1P_2$ et $Y = P_2P_1$ sont également deux codes préfixes finis maximaux sur A. Soit $\mathcal{A}_X = \{Q, A, \delta, \{q_0\}, \{q_0\}\}$ l'automate déterministe minimal reconnaissant X^*. Le produit de P_1 par P_2 étant non ambigu, l'ensemble des états atteints à partir de q_0 en lisant un mot de P_1 est un singleton $\{q_i\}$, où $q_i \neq q_0$. L'ensemble des états atteints à partir de q_i en lisant un mot de P_2 est évidemment le singleton $\{q_0\}$. On voit clairement sur la figure 4.1 que le langage Y^* est reconnu par l'automate $\mathcal{A}_Y = \{Q, A, \delta, \{q_i\}, \{q_i\}\}$. L'ensemble des mots atteignant l'état q_0 à partir de q_0 (resp. q_i) est le langage $(P_1P_2)^*$ (resp. $(P_2P_1)^*P_2$). Si $(P_1P_2)^* \cap (P_2P_1)^*P_2 \neq \emptyset$, alors il existe un mot u de A^* tel que $\delta(q_0, u) = \delta(q_i, u) = q_0$. L'alphabet de la base de $X^* \cap Y^*$ est bien A puisqu'une puissance de chacune des lettres de A apparait dans $X^* \cap Y^*$. Il ne reste donc qu'à appliquer le lemme 4.1 pour obtenir le résultat. □

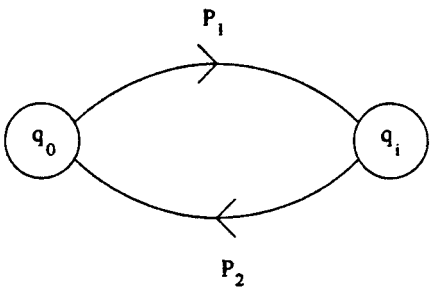


Figure 4.1 : Automate minimal de X^* et Y^*.

On peut toutefois remarquer que, si on se donne un code maximal préfixe fini X de la forme P_1P_2, et tel que la base de $X^* \cap (P_2P_1)^*$ est un code maximal, cela ne signifie pas forcément que la base de $X^* \cap Y^*$ sera maximale pour tout code maximal préfixe fini Y. Par exemple, pour

$$X = aA^5 + baA^4 + bbA^2, \text{ où } A = \{a, b\},$$

on peut prendre

$$P_1 = aA^4 + baA^3 + bbA \text{ et } P_2 = A.$$

On peut vérifier par le calcul que la base de $X^* \cap (P_2P_1)^*$ est un code maximal. Mais X se décompose également en $X = Q_1Q_2$ où

$$Q_1 = aA^3 + baA^2 + bb \text{ et } Q_2 = A^2.$$

Dans ce cas, la proposition 4.7 indique que la base de $X^* \cap (Q_2Q_1)^*$ n'est pas un code maximal.

La proposition suivante montre le rôle particulier joué par les codes bipréfixes maximaux vis-à-vis de la conjecture 4.1 :

Proposition 4.8 *Soit X un code rationnel maximal sur A. Il y a équivalence entre*

(i) La base de $X^* \cap Y^*$ est un code maximal sur A pour tout code rationnel Y maximal sur A.

(ii) X est un code bipréfixe.

Preuve Grâce à la proposition 4.1, on sait que (ii) implique (i). Pour montrer la réciproque, considérons un code X rationnel maximal sur A non bipréfixe et montrons qu'on peut toujours construire un code Y maximal sur A tel que la base de $X^* \cap Y^*$ n'est pas un code maximal sur A. On peut supposer X non suffixe (le cas où X est non préfixe est symétrique). Puisque X est un code coupant qui n'est pas un code suffixe maximal, d'après la proposition 4.5, il n'est pas complet à gauche. Il existe donc un mot u qui n'est facteur droit d'aucun mot de X^*. Comme tout code de la forme $A^*S \setminus A^*SA^+$, où S est un sous-ensemble non vide de A^+, le code $Y = A^*u \setminus A^*uA^+$ est un code préfixe maximal (cf. [4, p. 108, prop. 5.1]). Tout mot de Y^* se terminant par u, on a $X^* \cap Y^* = \{\varepsilon\}$, dont la base n'est pas un code maximal sur A. □

La proposition 4.8 n'est plus valable si on remplace *maximal sur A* par *maximal.* Par exemple, le code $X = a + ba + b^2$ est un code préfixe maximal vérifiant la condition $A^*a \subseteq X^*$. Le code X est non bipréfixe, et pourtant, pour tout code maximal Y, la base de $X^* \cap Y^*$ est un code maximal, comme le montre la proposition suivante :

Proposition 4.9 *Soit X un code rationnel maximal sur $A = \{a, b\}$. Si $A^*a \subseteq X^*$, alors pour tout code rationnel Y maximal, la base de $X^* \cap Y^*$ est un code maximal.*

Preuve Soit X un code rationnel maximal sur $A = \{a, b\}$ tel que $A^*a \subseteq X^*$. Comme tous les codes synchronisants (cf. [4, p. 115]), X est un code préfixe maximal. Soit Y un code rationnel maximal sur B. Soit Z la base de $X^* \cap Y^*$ et C l'alphabet de Z. L'alphabet C, nécessairement inclus dans $A \cap B$, contient au plus deux lettres. S'il contient zéro ou une lettre, alors Z est un code maximal. On suppose donc maintenant que C contient deux lettres, c'est-à-dire $C = \{a, b\}$. Le monoïde $X^* \cap Y^*$ contient donc un mot de la forme vaw, où v et w sont deux mots de A^*. Le monoïde X^* étant unitaire à droite et contenant les deux mots va et vaw, contient donc le mot w. Tout mot de A^* facteur d'un mot u de Y^* est également facteur du mot $uvaw$ de $X^* \cap Y^*$. Si Y est un code rationnel complet, Z est également un code rationnel complet et est donc maximal. □

4.3 Réciproque de la conjecture

La conjecture 4.1 étant fausse, on peut se demander quels sont les codes qui peuvent être obtenus comme base de l'intersection de deux monoïdes libres engendrés par des codes rationnels maximaux. La réponse, surprenante, est : tous les codes rationnels peuvent être obtenus. On peut prouver cela en montrant que, pour tout code rationnel Z, on peut construire, de manière effective, deux codes rationnels maximaux X et Y tels que Z soit la base de $X^* \cap Y^*$.

Proposition 4.10 *Soit Z un code rationnel non vide de A^*. Pour tout alphabet B contenant A, on peut construire, de manière effective, deux codes rationnels X et Y maximaux sur B tels que $Z^* = X^* \cap Y^*$.*

Preuve Soit Z un code rationnel non vide de A^* et B un alphabet contenant A (Le code Z est supposé non vide car la proposition est fausse dans le cas ou Z est le code vide et B est un alphabet d'une lettre). Si Z est un code maximal sur B, on obtient le résultat en prenant

$X = Y = Z$. On suppose maintenant que Z n'est pas maximal sur B (mais peut l'être sur A). Soit D un code rationnel maximal sur B et contenant Z. Le code D peut être obtenu de manière effective comme résultat d'une complétion du code $Z + (B \setminus A)$ obtenue grâce à la construction de Ehrenfeucht et Rozenberg ([18], [4, p. 62, prop. 5.2]). On peut toujours trouver dans D deux mots distincts u et v qui ne sont pas dans Z. Soit

$$X = Z + u + (D \setminus Z \setminus u)(D \setminus u)^* u$$

et

$$Y = Z + v + (D \setminus Z \setminus v)(D \setminus v)^* v.$$

On peut, grâce à la proposition 4.13, voir la maximalité des codes X et Y comme une conséquence de la maximalité des codes A_1 et A_2 de l'exemple 4.6. Tous les mots de X et Y sont des mots de D^*. Tout mot de $X^* \cap Y^*$ est donc également un mot de D^*, qui admet une décomposition unique à la fois sur X, Y et D. Soit w un mot de $X^* \cap Y^*$ se décomposant sur D en $w = d_1 \cdots d_n$. Le mot w ne peut se décomposer de manière unique à la fois sur X et sur Y que si tous les d_i sont dans Z. En effet, supposons que certains d_i ne soient pas dans Z et considérons j l'indice tel que d_j n'est pas dans Z et pour tout indice i supérieur à j, d_i est dans Z. On a alors $d_j = u = v$, ce qui contredit l'hypothèse que u et v sont distincts. Tout mot de $X^* \cap Y^*$ est donc dans Z^*. Réciproquement, il est clair que tout mot de Z^* est dans $X^* \cap Y^*$. On obtient donc finalement $Z^* = X^* \cap Y^*$. □

On obtient un résultat similaire à celui de la proposition précédente si on se restreint à la famille des codes préfixes rationnels :

Proposition 4.11 *Soit P un code préfixe rationnel non vide de A^*. Pour tout alphabet B contenant A, on peut construire, de manière effective, deux codes préfixes rationnels X et Y maximaux sur B tels que $P^* = X^* \cap Y^*$.*

Preuve Il suffit d'apporter quelques modifications à la preuve de la proposition 4.10. En effet, bien que le résultat de la construction de Ehrenfeucht et Rozenberg soit un code qui n'est jamais préfixe, il n'est pas difficile, pour un code préfixe rationnel P donné, de construire un code préfixe rationnel maximal D contenant P et au moins deux mots distincts u et v qui ne sont pas dans P. On voit clairement que si D est un code préfixe rationnel, alors

$$X = P + u + (D \setminus P \setminus u)(D \setminus u)^* u$$

et

$$Y = P + v + (D \setminus P \setminus v)(D \setminus v)^* v$$

sont deux codes préfixes rationnels maximaux tels que $P^* = X^* \cap Y^*$. □

La proposition 4.11 peut également être prouvée grâce à une construction basée sur des automates :

Soit P un code préfixe rationnel de A^* et B un alphabet contenant A.
Soit $\mathcal{A} = \{Q, B, \delta, \{I\}, \{I\}\}$ l'automate déterministe minimal reconnaissant P^* comme partie de B^*.
Soit $\mathcal{A}_1 = \{Q_1, B, \delta_1, \{I_1\}, \{I_1\}\}$ et $\mathcal{A}_2 = \{Q_2, B, \delta_2, \{I_2\}, \{I_2\}\}$ deux copies de $\mathcal{A}$, c'est-à-dire deux automates obtenus à partir de $\mathcal{A}$ par simple renommage des états.

Soit S_1 (resp. S_2) l'état puits de $\mathcal{A}_1$ (resp. $\mathcal{A}_2$).
Nous allons construire deux automates $\mathcal{A}_X$ et $\mathcal{A}_Y$ possédant les mêmes états et les mêmes transitions, et ne différant que par leur état initial et terminal. Leur ensemble d'états commun $Q_{X,Y}$ est obtenu en réunissant les états de $\mathcal{A}_1$ et de $\mathcal{A}_2$ et en fusionnant leur état puits respectif en un seul état $S_{X,Y}$. On a donc $Q_{X,Y} = (Q_1 \setminus \{S_1\}) \cup (Q_2 \setminus \{S_2\}) \cup S_{X,Y}$. Leur ensemble de transitions commun $\delta_{X,Y}$ s'obtient en deux étapes : on réunit d'abord les transitions de $\mathcal{A}_1$ et de $\mathcal{A}_2$, dans lesquelles les états S_1 et S_2 ont été renommés en $S_{X,Y}$. Ceci achève l'opération de fusion des deux états puits. On modifie ensuite les transitions partant du nouvel état $S_{X,Y}$ de manière à ce qu'au moins une d'entre elles arrive dans un état de $Q_1 \setminus \{S_1\}$ (resp. $Q_2 \setminus \{S_2\}$). Par construction, le graphe composé des états de $Q_{X,Y}$ et des transitions de $\delta_{X,Y}$ est fortement connexe. Les automates $\mathcal{A}_X = \{Q_{X,Y}, B, \delta_{X,Y}, \{I_1\}, \{I_1\}\}$ et $\mathcal{A}_Y = \{Q_{X,Y}, B, \delta_{X,Y}, \{I_2\}, \{I_2\}\}$ reconnaissent donc deux langages X^* et Y^*, où X et Y sont des codes préfixes maximaux. On peut voir que le produit cartésien $\mathcal{A}_{X,Y}$ des deux automates $\mathcal{A}_X$ et $\mathcal{A}_Y$ est un automate équivalent à $\mathcal{A}$. En effet, par construction, toutes les transitions de l'automate produit non concernées par l'état $S_{X,Y}$ sont de la forme $((q_1, q_2), a, (r_1, r_2))$, où (q, a, r) est une transition de $\mathcal{A}$ non concernée par l'état puits S. Aux transitions arrivant dans l'état puits S dans $\mathcal{A}$, correspondent les transitions arrivant dans l'état $(S_{X,Y}, S_{X,Y})$ dans l'automate produit. Le seul état terminal de $\mathcal{A}_{X,Y}$ est l'état initial (I_1, I_2). Tous les états atteints à partir de $(S_{X,Y}, S_{X,Y})$ sont de la forme (q, q), où q est un état de $Q_{X,Y}$, et sont donc non terminaux. Ils sont donc tous équivalents à l'état puits S de $\mathcal{A}$. Finalement, l'automate $\mathcal{A}_{X,Y}$ est équivalent à $\mathcal{A}$, c'est-à-dire que l'intersection de X^* et Y^* n'est autre que le langage P^*.

La figure 4.2 représente l'automate $\mathcal{A}_{X,Y}$ construit à partir du langage $P = a^2 + b^2$. On peut remarquer que les langages X et Y tels que $X^* \cap Y^* = P^*$ que l'on obtient alors sont différents de ceux de l'exemple 4.4.

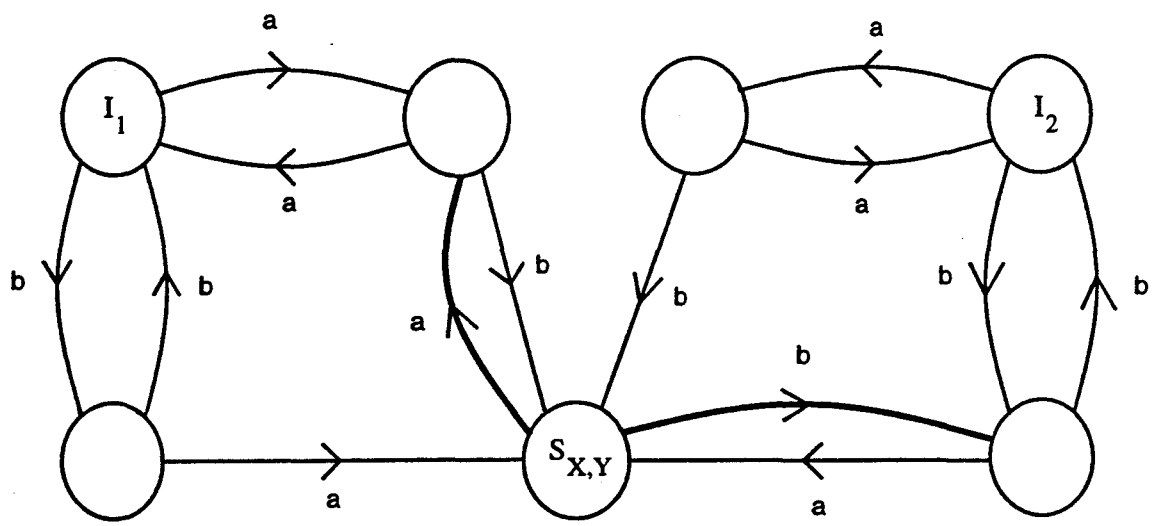


Figure 4.2 : Automate $\mathcal{A}_{X,Y}$ pour $P = a^2 + b^2$.

En vue de montrer que la construction proposée dans la preuve de la proposition 4.10 n'est pas la seule possible, la proposition suivante nous sera utile :

Proposition 4.12 *Soit l'alphabet $A = \{a_1, \ldots, a_n\}$. Soit $\{M_1, \ldots, M_n\}$ une partition en n codes rationnels d'un code rationnel M maximal sur l'alphabet B. Soit s la substitution rationnelle définie par $s(a_i) = M_i$, pour $i \in \{1, \ldots, n\}$. Soit X et Y deux codes rationnels maximaux sur A, et Z la base de $X^* \cap Y^*$. Les langages $s(X)$ et $s(Y)$ sont deux codes rationnels maximaux sur B et $s(Z)$ est la base de $(s(X))^* \cap (s(Y))^*$.*

Preuve Soit P^* le monoïde de langages de M^* engendré par $P = \{M_1, \ldots, M_n\}$. Chaque mot de M appartenant à un seul langage de P, et chaque mot de M^* se factorisant de manière unique sur M, il existe donc pour chaque mot w de M^*, un seul langage de P^* contenant w. Cela signifie que deux éléments distincts de P^* définissent deux langages de M^* disjoints. Le monoïde P^* est donc libre, et on peut voir la substitution s comme un morphisme injectif de A^* dans P^*, qui renomme chaque lettre de A en une lettre de P. Il est alors évident que si Z est la base de $X^* \cap Y^*$, alors $s(Z)$ est la base de $(s(X))^* \cap (s(Y))^*$. Montrons maintenant que si X est un code complet sur A, alors $s(X)$ est un code complet sur B. Le code M étant complet sur B, tout mot de B^* est facteur d'un mot de M^*. Tout mot de M^* appartient à un langage de P^*. Si X est un code complet sur A, alors $s(X)$ est un code complet sur P, et tout langage de P^* est facteur d'un langage de $(s(X))^*$, et donc, tout mot d'un langage de P^* est facteur d'un mot d'un langage de $(s(X))^*$. Finalement, tout mot de B^* est facteur d'un mot (d'un langage) de $(s(X))^*$, et donc $s(X)$ est un code complet sur B. □

La proposition suivante est un corollaire de la proposition précédente.

Proposition 4.13 *Soit Z un code rationnel de A^*, B un alphabet contenant A, et D un code rationnel maximal sur B et contenant Z. Soit $\{Z, D_1, \ldots, D_n\}$ une partition de D en codes rationnels. Soit A_1 et A_2 deux codes maximaux sur $C = \{a_0, a_1, \ldots, a_n\}$, tels que $A_1^* \cap A_2^* = a_0^*$. Soit s la substitution définie sur C par $s(a_0) = Z$ et $s(a_i) = D_i$, pour $i \in \{1, \ldots, n\}$. Les codes $X = s(A_1)$ et $Y = s(A_2)$ sont deux codes rationnels maximaux sur B et Z est la base de $X^* \cap Y^*$.*

La proposition 4.13 permet de trouver plusieurs constructions prouvant la proposition 4.10. Soit Z un code rationnel de A^*, B un alphabet contenant A, et D un code rationnel maximal sur B et contenant Z. Ces données sont communes aux trois exemples qui suivent. Chacun des exemples contient comme données un alphabet C, deux codes A_1 et A_2, et une substitution s et comme résultats deux codes X et Y. Les données de chaque exemple sont telles que l'on puisse leur appliquer la proposition 4.13 : on peut vérifier que A_1 et A_2 sont deux codes maximaux sur C, que la base de $A_1^* \cap A_2^*$ est une lettre de C dont l'image par s est Z, et que les images par s des lettres de C forment une partition de D. Les codes X et Y obtenus en résultat sont les codes $X = s(A_1)$ et $Y = s(A_2)$, qui, d'après la proposition 4.13, sont maximaux sur B et tels que Z est la base de $X^* \cap Y^*$.

Exemple 4.6 *Soit $C = \{a, b, c, d\}$. Les codes maximaux sur C*

$$A_1 = a + b + (c + d)(a + c + d)^* b$$

et

$$A_2 = a + c + (b + d)(a + b + d)^* c$$

et la substitution s définie sur C par

$$\begin{aligned} s(a) &= Z, \\ s(b) &= \{u\}, \\ s(c) &= \{v\}, \\ s(d) &= D \setminus Z \setminus \{u\} \setminus \{v\}, \end{aligned}$$

permettent de construire, grâce à la proposition 4.13, les codes maximaux sur B

$$X = Z + u + (D \setminus Z \setminus u)(D \setminus u)^* u$$

et

$$Y = Z + v + (D \setminus Z \setminus v)(D \setminus v)^* v.$$

qui sont tels que Z est la base de $X^* \cap Y^*$.
On retrouve ici les codes X et Y utilisés dans la preuve de la proposition 4.10.

Exemple 4.7 *Soit $C = \{a, b, c\}$. Les codes maximaux sur C*

$$A_1 = a + b + c(a + c)^* b$$

et

$$A_2 = a + c + b(a + b)^* c$$

et la substitution s définie sur C par

$$\begin{aligned} s(a) &= Z, \\ s(b) &= \{u\}, \\ s(c) &= D \setminus Z \setminus \{u\}, \end{aligned}$$

permettent de construire, grâce à la proposition 4.13, les codes maximaux sur B

$$X = Z + u + (D \setminus Z \setminus u)(D \setminus u)^* u$$

et

$$Y = (D \setminus u) + u(Z + u)^* (D \setminus Z \setminus u),$$

qui sont tels que Z est la base de $X^* \cap Y^*$.

Exemple 4.8 *Soit $C = \{a, b\}$. Les codes maximaux sur C*

$$A_1 = a + b(ba^* b)^* a$$

et

$$A_2 = a + b(aa^* b)^* b$$

et la substitution s définie sur C par

$$\begin{aligned} s(a) &= Z, \\ s(b) &= D \setminus Z, \end{aligned}$$

permettent de construire, grâce à la proposition 4.13, les codes maximaux sur B

$$X = Z + (D \setminus Z)((D \setminus Z)Z^*(D \setminus Z))^*Z$$

et

$$Y = Z + (D \setminus Z)(ZZ^*(D \setminus Z))^*(D \setminus Z),$$

qui sont tels que Z est la base de $X^* \cap Y^*$.

Les propositions 4.10 et 4.11 ne sont plus valides si on remplace *rationnel* par *fini*. Il existe en effet des codes (préfixes) finis qui ne peuvent être obtenus comme base de l'intersection de deux monoïdes libres engendrés par des codes (préfixes) finis maximaux.

Exemple 4.9 *Le code (préfixe) fini* $P = a + ba$*, comme tout code fini ne contenant pas une puissance de chaque lettre de son alphabet, ne peut être la base de l'intersection de deux monoïdes libres engendrés par des codes finis maximaux, qui contiennent nécessairement une puissance de chaque lettre de leur alphabet.*

Dans toutes les méthodes présentées ci-dessus qui permettent, à partir d'un code rationnel Z, de construire deux codes maximaux X et Y tels que Z soit la base de $X^* \cap Y^*$, les codes X et Y obtenus sont infinis. De même, il semble que tous les codes non maximaux obtenus, grâce à la proposition 4.7, comme base de l'intersection de deux monoïdes libres finiment engendrés, soient eux aussi infinis. On peut donc se demander s'il existe des codes (préfixes) finis non maximaux qui peuvent être obtenus comme base de l'intersection de deux monoïdes libres engendrés par des codes (préfixes) finis maximaux.

Proposition 4.14 *Il existe des codes préfixes finis X et Y maximaux sur A tels que la base de* $X^* \cap Y^*$ *est un code préfixe fini non maximal sur A.*

Preuve Soit $P_1 = (a+b)(a+ba+bb)$ et $P_2 = (a+ba+bb)(a+b)$. La base de $P_1^* \cap P_2^*$ est le langage $Z = a^2 + b^2a + b^3 + abb(ab^2+b^3)^*(aa+ba)$. Appelons partie finie de Z le langage $F = P_1 \cap P_2 = a^2+b^2a+b^3$, et partie infinie de Z le langage $I = Z \setminus F = abb(ab^2+b^3)^*(aa+ba)$. Si X est un code inclus dans P_1^* et Y un code inclus dans P_2^*, alors $X^* \cap Y^*$ est un monoïde inclus dans Z^*. On peut construire X et Y de telle manière que tout mot de Z^* contenant dans sa décomposition sur Z un facteur appartenant à la partie infinie, ne puisse appartenir à la fois à X^* et à Y^*. Pour cela, on place d'abord tous les mots de F^2 dans X et Y, puis on s'arrange, si cela est possible, pour que la parité du nombre de mots de F apparaissant après la dernière occurence d'un mot de I dans la décomposition sur Z d'un mot de Z^* soit différente dans X^* et dans Y^*. On obtient alors $X^* \cap Y^* = (F^2)^*$. Cette synchronisation est possible dans le cas qui nous intéresse : soit $W = ab^2(P_1 - a^2 + a^2P_1 - b^3 + b^3(P_1 - ba + baP_1))$, $X = aba + ba + W + F(F + aba + baP_1 + W)$ et $Y = ab + bab + ba^2 + F(F + ab + bab + ba^2P_2)$. Les langages X et Y sont deux codes préfixes finis maximaux et la base de $X^* \cap Y^*$ est le code préfixe fini non maximal $(a^2 + b^2a + b^3)^2$. □

4.4 Condition nécessaire et suffisante de maximalité

Une autre question intéressante vis-à-vis de la conjecture 4.1 est de savoir quelles conditions doivent vérifier X et Y pour qu'elle soit vraie. On sait déjà que le fait que X ou Y soit bipréfixe est une condition suffisante. Nous donnons ici une condition nécessaire et suffisante pour qu'elle soit vérifiée.

Nous avons besoin pour cela des définitions et des résultats préliminaires suivants :

Définition 4.1 *Soit X un code de A^*. Un mot p de A^* est* prolongeable *à droite (resp. à gauche) dans X^* si pour tout mot w de A^*, il existe un mot v de A^* tel que pwv (resp. vwp) est dans X^*.*

Définition 4.2 *Un mot s de A^* est* simplifiant *à droite (resp. à gauche) pour X^* si pour tout mot x de X^* et tout mot v de A^*, l'appartenance de xsv (resp. vsx) à X^* entraine celle de sv (resp. vs) à X^*.*

Proposition 4.15 *([4, p. 319, ex. 3.1]) Soit X un code coupant complet. Un mot est prolongeable à droite (resp. à gauche) dans X^* si et seulement si il est simplifiant à droite (resp. à gauche) pour X^*.*

Proposition 4.16 *Soit X un code coupant. Si z est un mot prolongeable à droite et à gauche dans X^*, alors il existe une puissance de z dans X^*.*

Preuve Soit X un code coupant de A^* et z un mot prolongeable à droite et à gauche dans X^*. Soit u un mot de A^* qui n'est facteur d'aucun mot de X. Puisque z est prolongeable à droite dans X^*, il existe pour tout entier i non nul un mot v_i de A^* tel que w^iuv_i est dans X^*. Puisque u n'est facteur d'aucun mot de X, il existe un facteur gauche u_i de u tel que w^iu_i est dans X^*. Le nombre de facteurs gauches de u étant fini, il existe deux entiers non nuls n et p tels que $n > p$ et $u_n = u_p$. Soit $m = n - p$ et $x = w^pu_p$. Les mots x et z^mx sont tous deux dans X^*. D'après la proposition 4.15, le mot z étant prolongeable à gauche dans X^*, est simplifiant à gauche pour X^*. Puisque X^* contient x et z^mx, il contient donc également z^m. □

Les faits suivants découlent directement des définitions.

Fait 4.1 *Soit X un code de A^*. Si un mot p de A^* est prolongeable à droite (resp. à gauche) dans X^*, alors pour tout mot w de A^*, le mot pw (resp. wp) est prolongeable à droite (resp. à gauche) dans X^*.*

Fait 4.2 *Soit X un code de A^*. Si un mot p de A^* est prolongeable à droite (resp. à gauche) dans X^*, alors pour tout mot x de X^*, le mot xp (resp. px) est prolongeable à droite (resp. à gauche) dans X^*.*

On peut maintenant prouver la proposition suivante :

Proposition 4.17 *Soit X et Y deux codes rationnels de A^*. La base de $X^* \cap Y^*$ est un code maximal sur A si et seulement si il existe un mot x_0 de X^* prolongeable à gauche dans Y^* et un mot y_0 de Y^* prolongeable à droite dans X^*.*

Preuve Soit X et Y deux codes rationnels de A^* et Z la base de $X^* \cap Y^*$. Si Z est un code maximal sur A, alors il existe un mot x de Z^* prolongeable à gauche dans Z^* et un mot y de Z^* prolongeable à droite dans Z^*. Puisque Z^* est inclus dans X^* et Y^*, il suffit de prendre $x_0 = x$ et $y_0 = y$.
Réciproquement, supposons l'existence de deux mots x_0 et y_0 vérifiant les conditions de la proposition. Pour montrer que Z est un code maximal sur A, nous allons montrer qu'il est complet dans A^*. Le mot y_0 de Y^* étant prolongeable à droite dans X^*, tout mot w de A^* est facteur d'un mot y_0wv de X^*, et donc le code X est complet. Il existe donc un mot x_1 de A^* prolongeable à gauche dans X^*. De même, le mot x_0 de X^* étant prolongeable à gauche dans Y^*, le code Y est complet et il existe un mot y_1 de A^* prolongeable à droite dans Y^*. Posons $x = x_1x_0$ et $y = y_0y_1$. D'après les faits 4.1 et 4.2, x est prolongeable à gauche dans X^* et Y^*, et y est prolongeable à droite dans X^* et Y^*. Soit w un mot quelconque de A^*. D'après le fait 4.1, le mot ywx est prolongeable à la fois à droite et à gauche à la fois dans X^* et Y^*. D'après la proposition 4.16, X^* et Y^* contiennent donc tous deux une puissance de ywx. Le monoïde $Z^* = X^* \cap Y^*$ contient donc également une puissance de ywx. Tout mot w de A^* étant facteur d'un mot de Z^*, Z est complet et donc maximal. □

Si X est un code préfixe (resp. suffixe) maximal sur A, alors tout mot de A^* est prolongeable à droite (resp. à gauche) dans X^*. La proposition 4.17 a donc pour corollaire immédiat la proposition suivante :

Proposition 4.18 *Si X est un code préfixe maximal sur A et Y est un code suffixe maximal sur A, alors la base de $X^* \cap Y^*$ est un code maximal sur A.*

La proposition 4.1 est un autre corollaire de la proposition 4.17. En effet, si X est un code rationnel bipréfixe maximal sur A, alors pour tout code rationnel Y maximal sur A, tout mot de Y^* est prolongeable à droite dans X^*, et Y étant complet, il contient un mot y prolongeable à gauche dans Y^*. Il existe alors un mot wy de X^* qui est également prolongeable à gauche dans Y^*.

Les propositions 4.1 et 4.18 ne sont plus vraies si on remplace *maximal sur A* par *maximal*, comme le montre l'exemple suivant :

Exemple 4.10 *Soit le code préfixe maximal sur $A = \{a, b, c\}$*

$$X = a + c + bc + b^2 + ba(a + b)^*c$$

et le code bipréfixe maximal sur $B = \{a, b\}$

$$Y = a + b$$

La base de $X^* \cap Y^*$ est le code non maximal

$$a + b^2.$$

Nous dirons qu'un code X maximal sur A conserve sa maximalité sur ses sous-alphabets, si pour tout alphabet B inclus dans A, $X \cap B^*$ est un code maximal sur B. La proposition [4, p. 67, prop. 5.9] nous indique que tout code fini conserve sa maximalité sur ses sous-alphabets. Il n'est pas difficile de voir, grâce à la proposition [4, p. 146, prop. 2.2], que tout code bipréfixe coupant possède la même propriété. On peut remplacer *maximal sur A* par *maximal* dans les propositions 4.1 et 4.18 si les codes X et Y considérés conservent leur maximalité sur leurs sous-alphabets. En effet, considérons deux tels codes X maximal sur A et Y maximal sur B, et appelons C l'alphabet de $X^* \cap Y^*$. Soit X_C la base de $X^* \cap C^*$ et Y_C la base de $Y^* \cap C^*$. Les codes X_C et Y_C sont maximaux sur C et $X^* \cap Y^* = X_C^* \cap Y_C^*$. On peut alors appliquer les propositions 4.1 et 4.18 pour conclure à la maximalité de la base de $X^* \cap Y^*$. On a donc en particulier les deux corollaires suivants :

Proposition 4.19 *Si X est un code préfixe fini maximal et Y est un code suffixe fini maximal alors la base de $X^* \cap Y^*$ est un code maximal.*

Proposition 4.20 *Si X est un code bipréfixe coupant maximal et Y est un code fini maximal alors la base de $X^* \cap Y^*$ est un code maximal.*

Conclusion

Suite aux résultats de ce chapitre, les principales questions qui se posent concernent surtout les codes finis. Par exemple, on peut se demander si la réciproque de la proposition 4.7 est vraie, c'est-à-dire si deux codes préfixes finis P_1 et P_2 maximaux sur A sont tels que $(P_1P_2)^* \cap (P_2P_1)^*P_2 = \emptyset$, alors la base de $(P_1P_2)^* \cap (P_2P_1)^*$ est-elle un code maximal ? Elle l'est dans tous les exemples rencontrés jusqu'à maintenant.
De même, lorsque la base de $(P_1P_2)^* \cap (P_2P_1)^*$ n'est pas un code maximal, il semble qu'elle soit alors toujours infinie. Une preuve ou un contre-exemple reste à trouver.
Enfin, est-il possible de trouver d'autres exemples plus simples prouvant la proposition 4.14 ? Ici "plus simple" ne signifie pas forcément "de taille inférieure". En effet, pour savoir s'il existe ou non un exemple plus petit, il reste un nombre fini de cas à tester. Il serait en fait intéressant d'avoir un exemple ayant une structure plus simple (plus symétrique par exemple), ou obtenu à partir d'une méthode plus facile à appréhender.

Références

[1] M.F. Barnsley, *Fractals Everywhere*, Academic Press (1988).

[2] J. Berstel, *Transductions and context-free languages.* B.G. Teubner, Stuttgart (1979).

[3] J. Berstel and M. Morcrette, *Compact representations of patterns by finite automata*, Proc. Pixim '89, Hermes, Paris, 387-402 (1989).

[4] J. Berstel and D. Perrin, *Theory of Codes*, Academic Press (1985).

[5] J. Berstel, D.Perrin, J.F. Perrot and A. Restivo, *Sur le théorème du défaut*, Journal of Algebra, Vol 60, 1979, pp. 169-180.

[6] J. Berstel and Ch. Reutenauer, *Rational Series and Their Languages*, Springer-Verlag, Berlin (1988).

[7] J.M. Boë, *Une famille remarquable de codes indécomposables*, Lecture Notes in Computer Science, 1978, pp. 105-112.

[8] V. Bruyère, L. Wang and L. Zhang, *On completion of codes with finite deciphering delay*, European Journal of Combinatorics 11, 1990, pp. 513-521

[9] Y. Césari, *Sur l'application du théorème de Suschkevitch à l'étude des codes rationnels complets*, Lecture Notes in Computer Science, 1974, pp. 342-350.

[10] K. Culik II and S. Dube, *Rational and Affine Expressions for Image Description*, Discrete Applied Mathematics 41, 1993, pp. 85-120.

[11] K. Culik II and J. Karhumäki, *Finite Automata Computing Real Functions*, SIAM J. Comp., to appear.

[12] K. Culik II and J. Kari, *Image Compression Using Weighted Finite Automata*, Computer & Graphics 17, 1993, pp. 305-313.

[13] C. De Felice, *Construction et complétion de codes finis*, Thèse de 3ème cycle, Rapport LITP 85-3, 1985.

[14] D. Derencourt, A three-word code which is not prefix-suffix composed, TCS, à paraître.

[15] D. Derencourt, J. Karhumäki, M. Latteux and A. Terlutte, *On Continuous Fonctions Computed By Finite Automata*, Theoretical Informatics and Applications 28, 1994, pp. 387-403.

[16] D. Derencourt, J. Karhumäki, M. Latteux and A. Terlutte, *On Computational Power of Weighted Finite Automata*, Proceedings of MFCS'92, LNCS 629, 1992, pp. 236-245.

[17] D. Derencourt and A. Terlutte, *Composition of codings*, Developments in language theory, World Scientific publ., 1993.

[18] A. Ehrenfeucht and G. Rozenberg, *Each regular code is included in a regular maximal code*, RAIRO, Theor. inform. appl. 17, 1983, pp. 89-96.

[19] S. Eilenberg, *Automata, Languages and Machines.* Vol A, Academic Press, New York (1974).

[20] C.C. Elgot and J.E. Mezei, *On relations defined by generalized finite automata.* IBM J. Res. Develop. 9, 1965, pp. 47-68.

[21] T. Harju, H.C.M. Kleijn and M. Latteux, *Compositional representation of rational functions.* RAIRO Theor. Inf. and Applications. 26, 1992, pp. 243-255.

[22] T. Harju, H.C.M. Kleijn and M. Latteux, *Deterministic rational functions.* Acta Informatica 29, 1992, pp. 545-554.

[23] T. Harju, H.C.M. Kleijn, M. Latteux and A. Terlutte, *Representation of rational functions with prefix and suffix codings.* Theoretical Computer Science 134, 1994, pp. 403-413.

[24] J. Karhumäki and M. Linna, *A note on morphic characterization of languages.* Discrete Appl. Math. 5, 1983, pp. 243-246.

[25] S.C Kleene, *Representation of events in nerve nets and finite automata* Automata Studies (C. Shannon and J. McCarthy eds, Princeton univ., Press Princeton, NJ, 1956), pp. 3-41.

[26] R. König, exposé Journées Montoises 1994.

[27] M. Latteux and J. Leguy, *On the composition of morphisms and inverse morphisms.* Lecture Notes in Comput. Sci. 154, 1983, pp. 420-432.

[28] M. Latteux and P. Turakainen, *A new normal form for the composition of morphisms and inverse morphisms.* Math. Syst. Theory 20, 1987, pp. 261-271.

[29] M. Nivat, *Transductions des langages de Chomsky.* Annales de l'Institut Fourier 18, 1968, pp. 339-456.

[30] A. Restivo, *On codes Having no Finite Completion*, Discrete Mathematics, Vol 17, 1977, pp. 309-316.

[31] A. Restivo, S. Salemi and T. Sportelli, *Completing codes*, RAIRO Theoretical Informatics and applications, Vol. 23, 1989, pp. 135-147.

[32] C.E. Shannon, *A mathematical Theory of Communication*, Bell. System Tech. J.27, 1948, pp. 379-656.

[33] M.P. Schützenberger, *une Théorie algébrique du codage*, Séminaire Dubreil-Pisot, 15, 1955-56, Institut Henri Poincaré, Paris.

[34] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Springer-Verlag, Berlin (1978).

[35] Z. Shen and L. Zhang, *Completion of recognizable bifix codes*, 1994.

[36] P. Turakainen, *A homomorphic characterization of principal semi-AFLs without using intersection with regular sets.* Inform. Sci. 27, 1982, pp. 141-149.

[37] P. Turakainen, *A machine-oriented approach to composition of morphisms and inverse morphisms.* EATCS Bull. 20, 1983, pp. 162-166.

Index