Autour des automates : génération aléatoire et contribution à quelques extensions

Vincent Carnino

Thèse

pour l'obtention du

doctorat de l'université Paris-Est

(spécialité informatique)

au titre de l'École doctorale de Mathématiques et des Sciences et Techniques de l'Information et de la Communication

soutenue le 5 décembre 2014

par Vincent CARNINO

Autour des automates : génération aléatoire et contribution à quelques extensions

---

Composition du jury :

Olivier CARTON,
Flavio D'ALESSANDRO,
Paul GASTIN (président du jury),
Jacques SAKAROVITCH,
Cyril NICAUD,
Sylvain LOMBARDY,

A mes oncles, Nino *et* Sylvio,
et à mes grand-mères, Solange *et* Mauricette.

Remerciements

Je tiens à remercier très sincèrement Sylvain Lombardy, pour qui j'ai le plus grand respect, qui m'a pris sous son aile avant même le début de ma thèse et qui m'a permis d'arriver beaucoup plus loin que je ne l'aurais espéré. Il est le maitre d'oeuvre de tout mon travail de recherche et sans lui je ne suis pas sûr que j'aurais ne serait-ce que trouvé un titre à ce manuscrit... Je me rappelerai toujours de ses innombrables exemples qui valent bien mieux que de longues explications.

Je tiens à remercier Cyril Nicaud pour avoir pris le relais en temps que directeur de thèse au moment où Sylvain est parti à Bordeaux et aussi pour avoir été le chef d'orchestre de mon premier article en collaboration avec Sven DeFelice. Merci également à ce dernier qui a été mon camarade de bureau pendant ces trois années de recherche et avec lequel j'ai échangé sur des sujets qui n'étaient pas toujours en rapport avec les automates.

Je remercie également Olivier Carton et Flavio D'Alessandro d'avoir accepté d'être rapporteurs pour mon manuscrit de thèse ainsi que Paul Gastin et Jacques Sakarovitch qui ont également accepté de participer au jury de ma soutenance.

Je remercie les professeurs et les chercheurs du laboratoire de l'institut Gaspard Monge qui, après m'avoir prodigué leur enseignement m'ont également appris à le transmettre, et qui ont su rester fidèles à eux-mêmes.

Je tiens à remercier du fond du coeur mes parents et ma soeur qui ont su me pousser toujours plus loin dans les études sans pour autant me forcer la main ("les études avant les filles" n'est ce pas maman !) ainsi que ma famille qui a toujours été derrière moi et qui était présente dans les moments les plus difficiles. Je regrette que certains ne soient plus là pour partager le résultat de ces années d'efforts...

Je remercie, d'abord et avant tout, ma schtroumpfette qui partage désormais ma vie ainsi que nos petits monstres qui sont toujours là pour nous distraire quand les heures de travail s'accumulent de trop. Elle a su me communiquer son énergie et sa force de caractère depuis le début et m'a toujours poussé vers l'avant, sans elle je n'aurai jamais été aussi loin.

Table des matières

Chapitre I

Introduction

Après l'arrivée des ordinateurs dans la première partie du vingtième siècle, on a vu l'émergence d'une nouvelle branche des mathématiques discrètes et de la logique centrée sur la résolution de problèmes dans le traitement de l'information par la machine : l'informatique théorique. Ces termes désignent une façon d'aborder l'informatique sous un angle plus formel et moins empirique, en faisant le plus souvent abstraction des aspects pratiques. Plusieurs domaines sont regroupés sous cette dénomination comme par exemple l'algorithmique, la théorie de la complexité, la calculabilité ou encore la théorie de l'information. Cette étude va avoir pour sujet central un autre de ces domaines : la théorie des automates.

Cette théorie, destinée à modéliser des mécanismes mathématiques formalisant des méthodes de calcul, se base sur le modèle de référence d'Alan Turing pour simuler le comportement d'un ordinateur. Ce modèle, appelé *machines de Turing*, est un modèle mathématique qui permet de rendre compte des capacités d'un ordinateur d'une façon plus simple et intuitive. Il reste cependant assez lourd et complexe, raison pour laquelle on a vu l'émergence d'outils plus spécifiques dont le plus connu reste l'automate qui est le sujet central de cette thèse. D'un point de vue très générique, l'objectif est de convertir un "problème" en un langage formel, chaque instance du problème étant représentée par un mot, et d'analyser ce langage grâce à un automate pour résoudre le "problème". Cela sous-entend qu'une grande partie du vocabulaire de la théorie des automates est emprunté à la théorie des langages formels mais, comme on le verra, également à la théorie des graphes.

Le sujet de cette étude est centré sur le thème des automates mais n'est pas axé sur un problème particulier. Nous allons aborder plusieurs extensions de ce concept comme par exemple les automates sur mots infinis ou encore les automates bidirectionnels. Nous traiterons également du sujet, plus concret, de la génération aléatoire d'automates déterministes acycliques qui sera utile

pour l'analyse d'éventuels algorithmes traitant de ce type d'automate.

$\sim$

Dans le premier chapitre, sans grande surprise, nous faisons des rappels sur les fondamentaux nécessaires à la lecture de cette thèse ainsi que sur les notations employées. Ainsi ce chapitre n'a pas été écrit pour être lu mais plutôt pour que le lecteur si reporte en cas de besoin. Il s'agit de définitions et de résultats classiques déjà énoncés un très grand nombre de fois. Cela dit en observant le même élément mais de différents points de vue, il est possible d'obtenir des résultats légèrement différents et surtout d'avoir une vue d'ensemble plus claire. C'est pourquoi nous formulons nos rappels dans l'optique des résultats que nous souhaitons prouver. Tant que cela n'implique pas des écarts dans notre sujet, nous essayons de donner les définitions les plus générales possibles.

Dans une première sous-section, nous commençons donc par présenter les éléments fondamentaux et les notations concernant les ensembles et les relations que nous manipulerons tout au long de ce mémoire sous différentes formes.

Nous rappelons ensuite toutes les notions mathématiques concernant les structures algébriques qui sont indispensables pour la suite. Nous nous intéressons plus particulièrement aux monoïdes qui sont à mettre en parallèle avec les automates ainsi qu'aux semi-anneaux qui sont utilisés dans le cas où l'on souhaite "compter" dans un automate. Nous rappelons également la notion de morphisme de monoïde qui nous amène au concept primordial de reconnaissabilité dans un monoïde que nous abordons d'abord d'un point de vue général puis dans un cadre plus restreint en traitant des monoïdes libres et des langages. Pour clôturer cette sous-partie, nous rappelons également les concepts propres aux séries formelles et plus particulièrement aux séries rationnelles qui sont à mettre en parallèle avec les automates à multiplicité que nous manipulons dans la suite.

Dans la dernière sous-section, nous abordons le vocabulaire de la théorie des graphes dont nous avons besoin pour traiter le sujet central : les automates. Nous présentons le concept d'automate en commençant par la version la plus générique (sur une structure de semi-anneau) et en avançant au fur et à mesure vers les automates "classiques" qui sont les plus largement utilisés. Nous concluons ces rappels en abordant le cas des automates à multiplicité où l'on ne fait plus que reconnaitre des mots d'un langage mais où l'on "compte" le nombre de fois où un mot est reconnu.

Le second chapitre est consacré à l'étude de ce que l'on appelle l'*automate universel* d'un langage mais dans le cas particulier des langages de mots infinis. Au même titre que l'automate minimal d'un langage, il s'agit d'un automate canoniquement associé au langage qu'il reconnait. Il a été introduit sous la forme d'une *matrice des facteurs* par J.H. Conway [20] et peut naturellement être présenté comme un automate. Les caractéristiques spécifiques de cet objet sont qu'il s'agit du plus petit automate qui reconnait le langage et dans lequel tout automate équivalent s'envoie par morphisme. La construction et les propriétés de cet automate reposent sur le concept de *factorisations* d'un langage (ou, de façon plus générale, d'un sous-ensemble d'un monoïde) qui consiste à "découper" un langage en sous-ensembles maximaux.

Cet objet a été défini, au départ, sur les langages de mots finis et nous l'avons donc étendu aux langages de mots infinis, également appelés ω-langages. Nous commençons donc ce chapitre avec un certain nombre de rappels sur les mots infinis ainsi que sur les automates qui les reconnaissent ; il en existe de plusieurs sortes comme les automates de Muller ou de Rabin, mais nous utilisons, pour notre part, les automates de Büchi qui sont plus intuitifs bien que possédant certains désavantages. Leur mode de reconnaissance très "simple", nous a ainsi poussés à introduire une forme normale dont nous donnons la définition après quoi nous prouvons que pour tout automate de Büchi, il en existe un équivalent et en forme normale. Nous faisons également un récapitulatif sur la reconnaissance par morphisme mais pour les ω-semigroupes qui correspondent à la façon correcte d'étendre la structure de monoïde pour les mots infinis.

Dans la seconde partie, nous présentons les résultats déjà connus sur l'automate universel après avoir rappelé les définitions des factorisations d'un langage qui forment le socle de la définition de cet automate. Nous rappelons également certaines propriétés intéressantes de cet objet notamment les liens étroits entre le label de ses états et leurs ensembles "passé/futur".

La dernière partie est dévolue à la présentation des nouveaux résultats : après avoir étendu le concept de factorisation pour les ω-langages, nous donnons la définition de l'automate universel d'un langage ω-rationnel et nous en prouvons les différentes propriétés : il s'agit du plus petit automate de Büchi en forme normale dans lequel tout automate de Büchi équivalent et en forme normale s'envoie par morphisme. Pour cela, nous prouvons, au préalable, quelques propriétés sur le passé et le futur de ses états qui sont assez similaires à ce que l'on a avec l'automate universel sur mots finis.

$\sim$

Le chapitre suivant a pour sujet les automates bidirectionnels à multiplicité. Deux extensions des automates "classiques" sont regroupés sous cette dénomination et notre objectif est d'étudier leur amalgame vis-à-vis du pouvoir d'expressivité.

La première, les automates bidirectionnels, est une généralisation des automates unidirectionnels (les automates "classiques"). Dans ces derniers le sens de lecture est figé et un mot en entrée est forcément lu de gauche à droite alors que dans la généralisation bidirectionnelle, on peut lire aussi bien de la gauche vers la droite que de la droite vers la gauche et ce à n'importe quel endroit dans le mot d'entrée, ce qui implique qu'une même lettre d'un mot peut être lue plus d'une fois.

La seconde extension est celle des automates à multiplicité (ou pondérés) dans lesquels on ne se contente pas de savoir si un mot est accepté ou pas mais de "compter" le nombre de fois où il est accepté. Pour cela, on utilisera une structure de semi-anneau qui permettra de sommer et de multiplier les poids des transitions rencontrées pour pouvoir attribuer une valeur à chaque calcul sur un mot donné (produit des transitions) et une valeur à chaque mot (la somme des calculs étiquetés par le mot).

Ainsi dans les deux premières parties, nous présentons d'abord les automates à multiplicité dans un semi-anneau quelconque ainsi que certains résultats importants les concernant, comme par exemple, l'extension du célèbre théorème de Kleene, appelé ici théorème Kleene-Schutzenberger, pour ensuite présenter l'extension qui nous intéresse, à savoir les automates bidirectionnels à multiplicité. Nous introduisons un certain nombre de définitions qui nous seront nécessaires dans la suite comme les $\mathbb{K}$-revêtements appliqués aux automates bidirectionnels, la δ-localité qui concerne la direction des transitions d'un automate bidirectionnel, le caractère réduit d'un calcul dans un automate bidirectionnel ou encore les "slices" d'un automate bidirectionnel qui vont servir de passerelle avec les automates unidirectionnels par le biais de l'automate des slices qui va être largement utilisé dans les preuves.

La troisième partie entre dans le vif du sujet en s'appuyant sur la construction de l'automate des slices pour affirmer que tout $\mathbb{K}$-automate bidirectionnel non ambigu est équivalent à un $\mathbb{K}$-automate unidirectionnel non ambigu. On poursuit par un résultat plus précis, en prouvant que tout $\mathbb{K}$-automate unidirectionnel non ambigu est équivalent à un $\mathbb{K}$-automate bidirectionnel déterministe.

La dernière partie se concentre sur une classe d'automates bidirectionnels à multiplicité que l'on appelle les automates tropicaux bidirectionnels. Il s'agit d'automates bidirectionnels pondérés dans les semi-anneaux tropicaux min-plus. On se concentre tout d'abord sur une version "positive" des automates tropicaux que l'on appelle automates de distance et on prouve

que tout automate de distance bidirectionnel est équivalent à un automate de distance unidirectionnel. Finalement, nous traitons le cas des automates min-plus bidirectionnels avec des poids éventuellement négatifs ce qui, couplé à la notion de circuit immobile, amène le problème de la validité d'un calcul, c'est à dire de savoir si la somme potentiellement infinie des poids des transitions de ce calcul est définie. Nous montrons donc qu'il existe des automates min-plus bidirectionnels tels que le langage de mots acceptés avec un poids défini n'est pas rationnel et nous prouvons également qu'il est décidable si un automate min-plus bidirectionnel est valide ou pas.

Dans le dernier chapitre de cette thèse on s'intéresse à une partie plus concrète de la théorie des automates : la génération aléatoire. L'objectif est de produire une certaine classe d'objets de façon optimale et en s'assurant que chaque objet de la classe possède la même probabilité d'être généré. Au final, on veut obtenir un générateur aléatoire qui va nous permettre de générer les objets de la classe en grand nombre, rapidement et de façon uniforme ce qui est très utile, par exemple, pour tester l'efficacité d'un algorithme qui prend ce type d'objet en entrée ou encore pour leur trouver des propriétés communes. Dans notre cas, nous avons mis au point un générateur aléatoire d'automates déterministes, accessibles et acycliques ainsi qu'une extension qui permet de générer des automates acycliques minimaux.

Dans la première partie, nous commençons par rappeler quelques définitions spécifiques dont nous auront besoin comme par exemple celle des automates en hamac ainsi qu'une définition alternative de la minimalité d'automate ou encore la définition d'isomorphisme, une classe particulière de morphisme qui va particulièrement nous servir pour assurer l'uniformité de nos algorithmes.

La seconde partie est dédiée aux rappels des fondamentaux pour tout ce qui concerne les chaînes de Markov et leur place dans le domaine de la génération aléatoire. Nous y abordons les différentes propriétés que l'on attend d'une chaîne de Markov et comment ces propriétés nous assurent que le générateur qui en résulte se comporte "correctement".

Les parties suivantes contiennent les résultats inédits que nous avons formulés : nous fournissons un algorithme de génération aléatoire pour les automates déterministes, accessibles et acycliques et un autre pour la classe plus restreinte des automates acycliques minimaux. Nous prouvons que ces deux algorithmes reposent sur des chaînes de Markov ergodiques (*i.e.* irréductibles et apériodiques) et symétriques, ce qui nous assure une génération quasi uniforme ainsi qu'une complexité en temps de l'ordre de $O(nT)$ et une complexité en espace de l'ordre de $O(n)$, où n est le nombre d'états de l'automate et où T est le nombre de déplacements dans la chaîne de Markov.

Chapitre II

Notions Fondamentales

1 Notations

Cette (courte) section est dédiée aux différentes notations qui seront utilisées au cours de ce mémoire. Je vais essayer de respecter un certain nombre de règles typographiques qui permettront au lecteur de se repérer plus facilement parmi les différentes structures utilisées.

1.1 Ensembles

Tout d'abord on utilisera des lettres majuscules classiques pour décrire les différents ensembles d'objets rencontrés ainsi que les monoïdes, avec le cas spécial de l'alphabet fini qui sera utilisé tout au long de cette étude et qui sera noté A. Si E est un ensemble, on notera $|E|$ le cardinal de cet ensemble. D'autre part, des majuscules ajourées (une fonte qui repose sur des doubles barres) seront utilisées pour noter les différents semi-anneaux rencontrés comme par exemple celui des entiers naturels $\mathbb{N}$, celui des réels $\mathbb{R}$ ou encore celui des entiers relatifs $\mathbb{Z}$. L'addition d'un semi-anneau sera notée $\oplus$ et sa multiplication $\otimes$.

On notera à l'aide de lettres minuscules classiques les éléments d'un ensemble ou d'un semi-anneau.

Dans le cas des mots finis, des lettres majuscules calligraphiées seront utilisées pour nommer les éventuels automates unidirectionnels et bidirectionnels ($\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$) avec le cas particulier de $\mathcal{L}$ qui sera utilisé pour les langages. Dans le cas des mots infinis, on utilisera des lettres de type gothiques ($\mathfrak{B}$) pour désigner les automates de Büchi avec le cas particulier de $\mathfrak{L}$ pour les langages de mots infinis.

Le *produit cartésien* de deux ensembles E et F (ou tout simplement produit) correspond à l'ensemble :

$$E \times F = \{(e, f) \mid e \in E, f \in F\}$$

On notera $\mathfrak{P}(E)$ l'ensemble des parties de l'ensemble E et on dira qu'un sous-ensemble de $\mathfrak{P}(E)$ est une *partition* de E si tous les éléments sont disjoints deux à deux et si leur union est égale à E.

1.2 Relations et fonctions

On emploiera des lettres minuscules grecques pour évoquer les éventuelles relations, fonctions et autres morphismes qui seront rencontrés.

Une fonction φ de E dans F, qu'on notera $\varphi\colon E \to F$, est un procédé qui associe à un élément e de E une valeur dans F qu'on appellera image de e (par φ) et que l'on notera $\varphi(e)$. L'image d'un ou de plusieurs éléments de E peut ne pas être définie, on introduit dans ce cas le concept de support de φ, noté $Supp(\varphi)$, qui sera constitué de l'ensemble des éléments de E qui ont une image par φ. De façon symétrique, on parlera de l'image de φ pour désigner l'ensemble des éléments de F qui sont l'image d'au moins un élément de E. On dira que φ est une application, lorsque son support est intrinsèquement égal à E tout entier de par sa définition (c'est le cas par exemple des morphismes qui seront présentés).

On dira qu'une fonction est injective si deux éléments distincts de E ont des images distinctes et surjective si tout élément de F est l'image d'au moins un élément de E. Si une fonction est à la fois injective et surjective, elle est dite bijective.

L'image inverse d'un sous-ensemble P de F, noté $\varphi^{-1}(P)$, est l'ensemble des éléments de E qui ont pour image un élément de P, c'est à dire

$$\varphi^{-1}(P) = \{e \in E \mid \varphi(e) \in P\}$$

Un relation φ de E dans F est un ensemble de couples (e, f) avec e dans E et f dans F, où l'on dira que e et f sont mis en relation par φ. Ce sous-ensemble de $E \times F$ peut également être vu comme une fonction $\varphi\colon E \to F$ de E dans F et de ce fait, on peut employer le même vocabulaire. Avec une telle définition, il est clair qu'une fonction peut être vue comme une relation et inversement (plus précisément, une relation peut être vue comme une application).

Une relation $\varphi\colon E \to E$ de E dans lui-même est dite :

- réflexive si, $\forall e \in E$, $e \in \varphi(e)$;
- transitive si, $\forall e, f, g \in E$, $(f \in \varphi(e)$ et $g \in \varphi(f)) \Longrightarrow g \in \varphi(e)$;
- symétrique si, $\forall e, f \in E$, $f \in \varphi(e) \Longrightarrow e \in \varphi(f)$, *i.e.* si $\varphi = \varphi^{-1}$;
- antisymétrique si, $\forall e, f \in E$, $(f \in \varphi(e)$ et $e \neq f) \Longrightarrow e \notin \varphi(f)$.

Une relation qui est à la fois réflexive, symétrique et transitive est appelée relation d'équivalence. Si $\varphi\colon E \to E$ est une relation d'équivalence sur E et si $f \in \varphi(e)$, on note $e \equiv f[\varphi]$ et on appelle classe (d'équivalence) de e par φ le sous-ensemble $\varphi(e)$. On peut ensuite construire de nouveaux ensembles en "assimilant" les éléments d'une même classe d'équivalence à un seul et même élément. L'ensemble des classes est appelé ensemble quotient de E par φ et est noté E/φ. Ainsi, une application φ de E dans F définie naturellement une équivalence sur E :

$$\forall x, y \in E, \qquad x \sim y \Longleftrightarrow \varphi(x) = \varphi(y)$$

Une relation qui est à la fois réflexive, antisymétrique et transitive est appelée relation d'ordre. Si $\varphi\colon E \to E$ est une relation d'ordre sur E, on note $e\varphi f$ pour $f \in \varphi(e)$ (on préfèrera utiliser des symboles comme $\leq, \geq, \preceq, \succeq, \ldots$ plutôt que des lettres grecques pour désigner de telles relations. De cette façon, avec une relation d'ordre (qu'on appelle aussi simplement un ordre) $\leq$, avec $e \leq f$ on dira que e est inférieur à f pour l'ordre $\leq$ et, si pour tous e et f dans E, on a $e \leq f$ ou $f \leq e$ alors cet ordre est dit total.

2 Structures algébriques et langages

2.1 Monoïdes et semi-anneaux

Monoïdes

Définition 2.1 Un *monoïde* est un ensemble M muni d'une loi binaire associative, que l'on appellera généralement multiplication, et d'un (unique) *élément neutre* que l'on notera 1_M. Un monoïde qui ne dispose pas d'un élément neutre est appelé un *semigroupe*.

La multiplication est notée avec un point : $p \cdot q$ ou simplement pq s'il n'y a pas de risque d'ambiguïté. Lorsque l'on voudra spécifier une certaine opération de multiplication $\otimes$ pour un monoïde M, on désignera le monoïde par $(M, \otimes)$. Si cette dernière est commutative, on dit que le monoïde est commutatif et on appellera en général cette opération l'addition. S'il existe

un élément z tel que pour tout élément m de M on a $zm = mz = z$, on dit que z est absorbant, on l'appellera le *zéro* du monoïde et on le notera 0_M.

REMARQUE 2.1 Il est toujours possible de munir un semigroupe d'un élément neutre et donc d'en faire un monoïde. De ce fait, s'il s'avère que la structure de base est un semigroupe, on la transformera implicitement en monoïde pour ne manipuler que ce type de structure.

EXEMPLE 2.1 Étant donné que les axiomes de monoïde sont relativement faibles, tous les groupes sont des monoïdes de même que la plupart des ensembles classiques munis d'une loi qui vérifie ces axiomes, en voici quelques exemples :

i) $(\mathbb{N}, +)$, l'ensemble des entiers naturels muni de l'addition avec 0 pour élément neutre.

ii) $(\mathbb{N}, max)$, l'ensemble des entiers naturels muni de la loi Max (qui à deux entiers associe le plus grand des deux) avec 0 pour élément neutre.

iii) $(\mathbb{Z}, *)$, l'ensemble des entiers relatifs muni de la multiplication avec 1 pour élément neutre (et 0 comme élément absorbant).

iv) $(\mathfrak{P}(E), \cup)$, l'ensemble des partie d'un ensemble E muni de l'union ensembliste avec l'ensemble vide pour élément neutre.

v) $(A^*, .)$, l'ensemble des mots sur l'alphabet A avec l'opération de concaténation et avec le mot vide pour élément neutre.

Définition 2.2 Soit $(M, \cdot)$ un monoïde et soient x et y deux éléments de M. Le *quotient à gauche* de y par x, noté $x^{-1}y$ correspond à l'ensemble des éléments de M par lesquels on peut multiplier x pour obtenir y et est égal à $\{z \in M \mid x \cdot z = y\}$. Le quotient à gauche d'un sous-ensemble N de M par y correspond à l'union des quotients des éléments de N par y.

Définition 2.3 Un *sous-monoïde* S d'un monoïde M est un sous-ensemble de M qui contient l'élément neutre et qui est stable pour la multiplication.

EXEMPLE 2.2 L'ensemble $\{0, \ldots, 10\}$ muni de la loi $min(x + y, 10)$ est un monoïde d'élément neutre 0. Avec cette même loi mais en restreignant l'ensemble à $\{0, 1, 3, 6, 8, 10\}$, on obtient un sous-monoïde du monoïde de départ.

Définition 2.4 Soit $(M, \cdot)$ un monoïde. Pour tout x dans M, pour tout n dans $\mathbb{N}$, on définit la *puissance* n-ième de x, notée x^n par $x^0 = 1_M$ et $x^n = x^{n-1} \cdot x$. Un élément e de M est un *idempotent* si $e \cdot e = e$. Si tous les éléments de M sont idempotents, alors M est un monoïde idempotent.

La définition de puissance d'un élément peut être naturellement étendue aux ensembles. Soit X et Y, deux sous-ensembles de M, on définit la multiplication de X et Y par :

$$X \cdot Y = \{x \cdot y \mid x \in X \text{ et } y \in Y\}$$

ce qui nous amène à :

$$X^0 = 1_M \quad \text{et} \quad \forall n \in \mathbb{N}, \ \ X^n = X^{n-1} \cdot X$$

EXEMPLE 2.3 Pour les entiers naturels $\mathbb{N}$ munis de la multiplication classique, 1 est toujours idempotent et s'il existe un zéro alors ce dernier est également un idempotent. L'ensemble $\{0, 1\}$ muni de la loi binaire "OU" est un monoïde idempotent de même que le monoïde $(\mathfrak{P}(E), \cup)$.

Définition 2.5 Si P est une partie quelconque de M, on appelle *sous-monoïde engendré* par P, et on note $\langle P \rangle$, le plus petit sous-monoïde de M qui contient P. On dira qu'une partie P de M est un *ensemble générateur* de M si M est le plus petit ensemble qui contient P et qui est stable pour la multiplication, ou autrement dit, si le plus petit sous-monoïde de M qui contient P est M lui-même (si $M = \langle P \rangle$). S'il existe un tel ensemble P fini, on dira que M est *finiment engendré*.

EXEMPLE 2.4 Si l'on prend l'ensemble des entiers naturels $\mathbb{N}$ muni de l'addition $+$ alors le monoïde $(2\mathbb{N}, +)$ des entiers pairs est un sous-monoïde de $(\mathbb{N}, +)$ (l'élément neutre, 0, est le même que $(\mathbb{N}, +)$). Ces deux monoïdes sont finiment engendrés (par $\{1\}$ pour $(\mathbb{N}, +)$ et par $\{2\}$ pour $(2\mathbb{N}, +)$).
Le monoïde $(\mathbb{N}, max)$, quant à lui, n'est pas finiment engendré.

Définition 2.6 Un *morphisme (de monoïdes)* est une application $\varphi \colon M \to N$ d'un monoïde $(M, *)$ dans un monoïde $(N, \star)$ qui envoie l'élément neutre de M sur celui de N, *i.e.* $\varphi(1_M) = 1_N$, et qui respecte la multiplication, *i.e.*

$$\forall p, q \in M, \ \varphi(p * q) = \varphi(p) \star \varphi(q)$$

Lorsqu'un morphisme est bijectif, on parle d'isomorphisme.

EXEMPLE 2.5 L'application φ de $(\mathbb{N}, +)$ dans $(2\mathbb{N}, +)$ définie par :

$$\begin{array}{ccc} \varphi & (\mathbb{N}, +) & \to (2\mathbb{N}, +) \\ & n & \to \quad 4n \end{array}$$

qui associe à un entier son quadruple est un morphisme de monoïde. En effet, quels que soient m et n dans $\mathbb{N}$, $\varphi(m) + \varphi(n) = 4m + 4n = 4(m + n) = \varphi(m + n)$.

Définition 2.7 Soit $\sim$ une *relation d'équivalence* sur un monoïde $(M, \cdot)$. Cette équivalence est une *congruence* droite si elle est régulière à droite pour $\cdot$, c'est-à-dire si le produit à droite de deux éléments équivalents par n'importe lequel des éléments de M est lui aussi équivalent :

$$\forall x, y \in M \quad x \sim y \implies \forall z \in M,\ x \cdot z \sim y \cdot z$$

De même, c'est une congruence gauche si elle est régulière à gauche :

$$\forall x, y \in M \quad x \sim y \implies \forall z \in M,\ z \cdot x \sim z \cdot y$$

Si une relation d'équivalence est régulière à droite et à gauche, c'est une congruence. L'ensemble des classes d'équivalence de M modulo cette équivalence, appelé également *ensemble quotient*, forme alors un monoïde.

EXEMPLE 2.6 La relation d'équivalence $\sim_2$ sur $(\mathbb{N}, \times)$ qui associe deux éléments s'il ont même parité est une congruence. En effet, pour tout m et n dans $\mathbb{N}$, si $m \sim_2 n$ alors quel que soit k dans $\mathbb{N}$, $m \times k \sim_2 n \times k$ (c'est-à-dire que $\sim_2$ est une congruence droite) car si k est impair, les produits $m \times k$ et $n \times k$ sont impairs lorsque m et n le sont également sinon les produits sont pairs. Étant donné que $(\mathbb{N}, \times)$ est commutatif, $\sim_2$ est une congruence (gauche et droite).

Il existe également un certain nombre d'équivalences bien connues qui permettent de décrire les éléments d'un monoïde (ou d'un semigroupe). Il s'agit des relations de Green qui sont des relations d'équivalence se basant sur le concept d'idéal.

Définition 2.8 Un sous-ensemble I d'un monoïde M est un *idéal à gauche* (*resp.* à droite) si $MI \subseteq I$ (*resp.* si $IM \subseteq I$). Si un sous-ensemble est à la fois un idéal gauche et un idéal droit, on dit que c'est un *idéal bilatéral*. Un idéal est dit *principal* s'il est engendré par un unique élément de M.

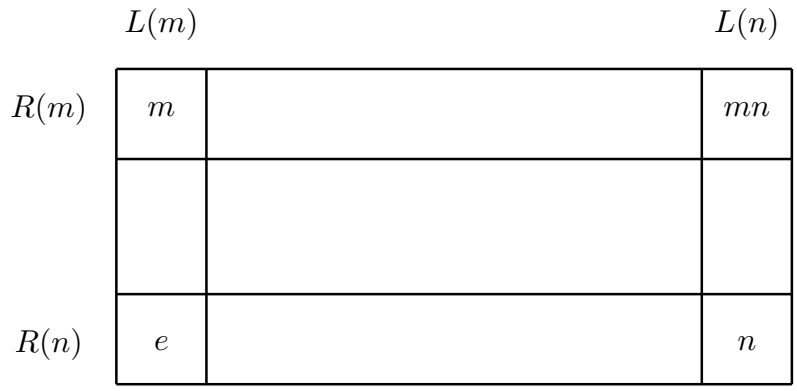


FIGURE 2.1 – Représentation d'une $\mathcal{D}$-classe.

Soit n un élément de M, les idéaux (principaux) engendrés par n sont :
- l'idéal à gauche engendré par n : $Mn = \{mn \mid m \in M\}$
- l'idéal à droite engendré par n : $nM = \{nm \mid m \in M\}$
- l'idéal bilatéral engendré par n : $MnM = \{mnm' \mid m, m' \in M\}$

Ces idéaux permettent de définir des classes d'équivalence sur les éléments d'un monoïde de sorte que deux éléments sont équivalents s'ils engendrent le même idéal (gauche, droit ou bilatéral). Les relations qui en découlent sont appelées relations de Green d'un monoïde.

Définition 2.9 Soit M un monoïde et soient m et n deux éléments de M :
- $m\mathcal{L}n \iff \exists x, y,\ m = x \cdot n$ et $n = y \cdot m$.
- $m\mathcal{R}n \iff \exists x, y,\ m = n \cdot x$ et $n = m \cdot y$.
- $m\mathcal{J}n \iff \exists x, y, x', y'\ m = xn \cdot y$ et $n = x'm \cdot y'$.
- $m\mathcal{H}n \iff m\mathcal{L}n$ et $m\mathcal{R}n$.
- $m\mathcal{D}n \iff \exists m' \in M,\ m\mathcal{L}m'$ et $m'\mathcal{R}n$.

On notera X_m, la classe d'équivalence d'un élément m pour la relation $\mathcal{X}$.

On peut remarquer que H_m correspond à l'intersection entre L_m et R_m. De plus, la relation $\mathcal{D}$ est incluse dans $\mathcal{J}$ (c'est-à-dire que $m\mathcal{D}n \Longrightarrow m\mathcal{J}n$) et si M est un monoïde fini, ces deux relations sont les mêmes. On peut voir la relation $\mathcal{D}$ comme une union de $\mathcal{R}$-classes, ou de $\mathcal{L}$-classes ou encore de $\mathcal{H}$-classes, ce qui a amené à une représentation de ces classes d'équivalence sous forme de "boites à œufs" (cf. [19]). Chaque $\mathcal{R}$-classe correspond à une ligne de la "boite à œufs", chaque $\mathcal{L}$-classe correspond à une colonne, tandis que les $\mathcal{H}$-classes sont les "œufs" eux-mêmes (voir Figure 2.1).

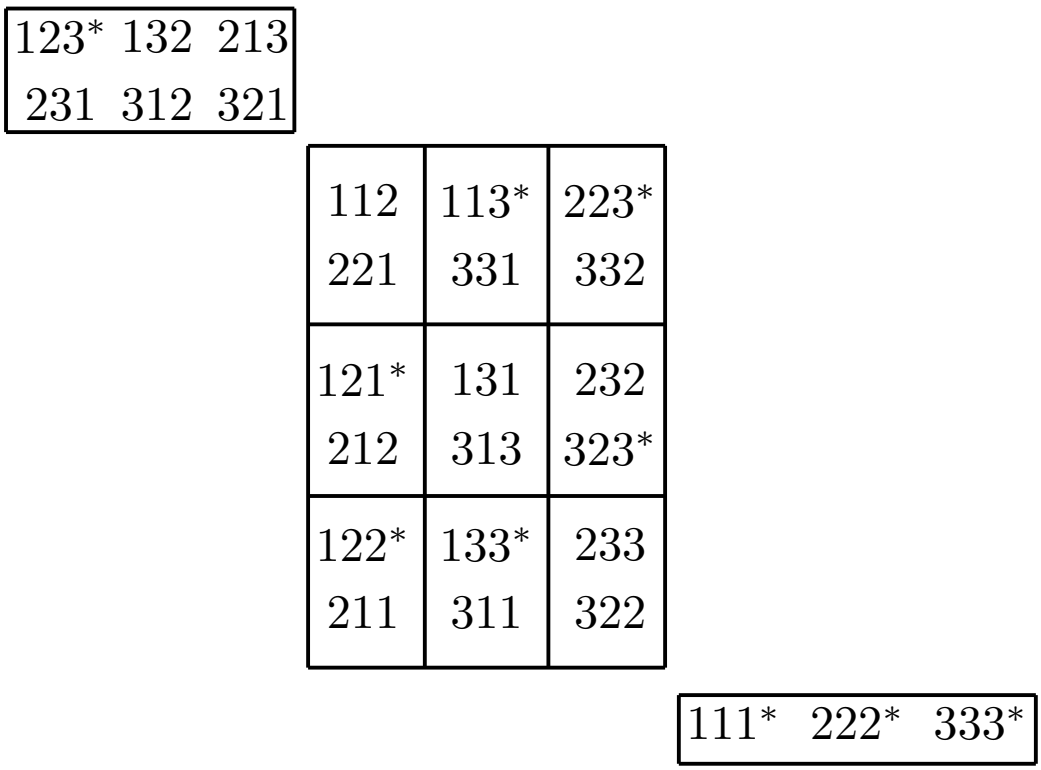


FIGURE 2.2 – Le monoïde des transformations à 3 éléments.

EXEMPLE 2.7 L'exemple le plus répandu est celui du monoïde des applications de $\{1, 2, 3\}$ dans lui-même, appelé aussi monoïde des transformations à 3 éléments (les éléments annotés avec un astérisque sont les idempotents du monoïde), qui est représenté sur la figure 2.2.

On construit en général ces "boites à œufs" de sorte que le produit de deux éléments d'une même $\mathcal{D}$-classe ne puisse se situer que dans une $\mathcal{D}$-classe se trouvant plus "bas" dans le diagramme. Ces relations de Green sont en général un bon moyen de comprendre et d'observer la structure interne d'un monoïde en décrivant le comportement de ses éléments. C'est pourquoi elles ont été utilisées pour repérer certaines caractéristiques de monoïde comme par exemple le fait d'être un groupe (les 5 relations coïncident et l'on a une seule et même $\mathcal{H}$-classe) ou encore d'être apériodique (les $\mathcal{H}$-classes ne possèdent qu'un seul élément).

Semi-anneaux

Définition 2.10 Un *semi-anneau* $\mathbb{K}$ est un ensemble muni de deux lois, une addition et une multiplication qui respectent les propriétés suivantes :

i) $\mathbb{K}$ est un monoïde pour la multiplication et son élément neutre $1_{\mathbb{K}}$ est appelé unité de $\mathbb{K}$.

ii) $\mathbb{K}$ est un monoïde commutatif pour l'addition et son élément neutre $0_{\mathbb{K}}$ est appelé le zéro de $\mathbb{K}$.

iii) la multiplication est distributive par rapport à l'addition (à gauche et à droite).

iv) $0_{\mathbb{K}}$ est absorbant pour la multiplication (c'est un zéro).

Un semi-anneau $(\mathbb{K}, \oplus, \otimes)$ est dit idempotent si $(\mathbb{K}, \oplus)$ est un monoïde idempotent.

EXEMPLE 2.8

i) Les ensembles $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ et $\mathbb{R}$ munis des lois d'addition et de multiplication classiques sont des semi-anneaux. On parlera respectivement du semi-anneau des entiers naturels, de celui des entiers relatifs, de celui des rationnels et de celui des réels.

ii) Le semi-anneau de Boole est l'ensemble $\{0, 1\}$ muni des deux lois (commutatives) :

$\oplus$	0	1
0	0	1
1	1	1

$\oplus$	0	1
0	0	0
1	0	1

Il s'agit, en fait, de l'ensemble $\{TRUE, FALSE\}$ muni des lois binaires OR et AND, respectivement pour l'addition et la multiplication. Ce semi-anneau est idempotent car, comme nous l'avons vu plus haut, $\{0, 1\}$ muni de l'addition est un monoïde idempotent.

iii) Une des familles de semi-anneaux très utilisée est celle des semi-anneaux $\mathbb{Z}/n\mathbb{Z}$, où n est un entier. Par exemple, le semi-anneau $\mathbb{Z}/2\mathbb{Z}$ possède les mêmes lois que le semi-anneau de Boole exceptée l'addition où $1 \oplus 1 = 0$.

iv) Le semi-anneau $\mathcal{M} = (\mathbb{N} \cup \{+\infty\}, min, +)$, où $+\infty$ est plus grand que n'importe lequel des éléments de $\mathbb{N}$ et aussi l'élément neutre pour min et où 0 est l'élément neutre pour $+$.

v) L'un des semi-anneaux les plus utilisés est le semi-anneau des parties d'un ensemble E muni de l'union ensembliste ainsi que de l'intersection, $(\mathfrak{P}(M), \cup, \cap)$. Il s'agit en fait d'une algèbre de Boole.

Définition 2.11 Soit $(\mathbb{K}, \oplus, \otimes)$ un semi-anneau. Pour tout n dans $\mathbb{N}$, on pose :

$$X_n = \bigoplus_{k=0}^{n} x^k$$

La *limite* de X_n lorsque n tend vers l'infini (si cette limite existe), est appelée *étoile* de x et est notée x^* :

$$x^* = \bigoplus_{k=0}^{\infty} x^k \ , \quad x^+ = \bigoplus_{k=1}^{\infty} x^k$$

REMARQUE 2.2 Cette notation s'étend naturellement aux ensembles. Un semi-anneau muni de cet opérateur unaire (l'étoile de Kleene) est appelé semi-anneau itératif ou étoilé.
A noté également que si l'on part d'un monoïde M et que l'on se place dans le semi-anneau $(\mathfrak{P}(M), \cdot, \cup)$, l'étoile d'une partie P revient à prendre le sous-monoïde engendré par P, c'est-à-dire que $P^* = \langle P \rangle$.

Définition 2.12 Un *morphisme de semi-anneau* $\varphi \colon \mathbb{K} \to \mathbb{L}$ est une application de $\mathbb{K}$ dans $\mathbb{L}$ telle que :

$$\forall k, l \in \mathbb{K} \qquad \begin{cases} \varphi(k+l) = \varphi(k) + \varphi(l) & \text{et} \quad \varphi(0_{\mathbb{K}}) = 0_{\mathbb{L}} \\ \varphi(k \cdot l) = \varphi(k) \cdot \varphi(l) & \text{et} \quad \varphi(1_{\mathbb{K}}) = 1_{\mathbb{L}} \end{cases}$$

A noter que le $+$ et le $\cdot$ des membres gauches des égalités correspondent à l'addition et à la multiplication de $\mathbb{K}$ alors que dans les membres droits, il s'agit des opérations binaires de $\mathbb{L}$. En d'autres termes, φ est un morphisme à la fois pour l'addition et pour la multiplication.

2.2 Ensembles reconnaissables et ensembles rationnels

Dans la suite, nous ferons coïncider deux types d'ensembles, les parties reconnaissables et les parties rationnelles. Il s'agit d'un théorème extrêmement important, le théorème de Kleene dont on rappellera les détails dans la suite. Pour le moment, nous allons nous intéresser aux concepts généraux de reconnaissabilité et de rationalité.

Définition 2.13 Un sous-ensemble L d'un monoïde M est *reconnu* par un monoïde N s'il existe un morphisme (surjectif) φ de M dans N et un sous-ensemble P de N tels que $L = \varphi^{-1}(P)$ (ou, autrement dit, si L est l'image

inverse par φ d'une partie de N). Si N est un monoïde fini, alors L est un sous-ensemble *reconnaissable* de M. Une autre façon de voir les choses est de dire que L est une partie reconnaissable de M si elle est saturée par une congruence d'index fini.

On note Rec M l'ensemble des parties reconnaissables de M.

EXEMPLE 2.9 Soit S le semigroupe défini sur $\{0,1,2\}$ par $x+y = min(x+y, 2)$ et soit φ un morphisme (surjectif) de $(\mathbb{N},+)$ dans S défini par $\varphi(0)=0$, $\varphi(1)=1$ et $\varphi(n)=2$ quel que soit $n>1$.

Alors les ensembles reconnaissables de $\mathbb{N}$ par φ sont $\varphi^{-1}(\emptyset)=\emptyset$, $\varphi^{-1}(0)=\{0\}$, $\varphi^{-1}(1)=\{1\}$, $\varphi^{-1}(2)=\{2,3,\ldots\}$, $\varphi^{-1}(\{0,1\})=\{0,1\}$, $\varphi^{-1}(\{0,2\})=\{0,2,3,\ldots\}$, $\varphi^{-1}(\{1,2\})=\{1,2,3,\ldots\}$ et $\varphi^{-1}(\{1,2,3\})=\mathbb{N}$.

Définition 2.14 On appelle *opérations rationnelles* sur un monoïde M (en réalité, il s'agit d'un abus de langage car ces opérations sont en fait définies sur $\mathfrak{P}(M)$) les trois opérations suivantes :

i) l'union $X \cup Y$

ii) le produit $X \cdot Y$

iii) l'étoile X^*

L'ensemble Rat M des ensembles rationnels de M est la plus petite famille, au sens de l'inclusion, de sous-ensembles de M qui contient l'ensemble vide $\emptyset$, les singletons, et qui est stable pour les opérations rationnelles. Il s'agit de la clôture des parties finies de M pour ces opérations.

L'ensemble Rat M forme un sous-semi-anneau de $\mathfrak{P}(M)$.

REMARQUE 2.3 Dans le cas particulier d'un monoïde libre ou d'un monoïde fini, ces deux notions vont coïncider, cela dit, dans le cas général d'un monoïde quelconque, il y a une différence entre un ensemble rationnel et un ensemble reconnaissable.

2.3 Monoïde libre et langages

Précédemment, nous avons rappelé des concepts de base des monoïdes. Il y a une famille de monoïdes qui occupe une place particulière, il s'agit des monoïdes libres.

Définition 2.15 Un sous-ensemble N d'un monoïde M est une base de M si N est un *ensemble générateur* de M dans lequel tout élément a une unique décomposition. Un monoïde qui admet une base (unique) est appelé *monoïde libre*.

Définition 2.16 On appelle *alphabet* un ensemble fini[1] A non vide de symboles appelés lettres. Un *mot* sur A est le résultat du produit (la concaténation) d'éléments de A. Le monoïde libre engendré par A, que l'on note A^*, correspond à l'ensemble des mots sur l'alphabet A. Étant donné l'unicité du produit (ici la concaténation), il est logique de parler de la i-ième lettre d'un mot w que l'on note w_i.

L'élément neutre de ce monoïde est le mot vide que l'on note 1_{A^*}. Si l'élément neutre est absent, on parle du semigroupe libre engendré par A que l'on note A^+.

REMARQUE 2.4 Dans notre cas, les notions de mots et de langages ont des sens totalement différents de ceux qu'on leur donne habituellement. Ici, un mot n'est qu'une suite de lettres quelconques comme par exemple le mot *abbabbabab* sur l'alphabet $\{a, b\}$ qui n'a de sens dans aucune langue. On peut envisager toutes sortes d'alphabets, comme par exemple l'ensemble $\{0, \ldots, 9, A, B, C, D, E, F\}$ qui permet d'écrire des nombres sous forme hexadécimale, ou encore l'alphabet binaire $\{0, 1\}$, voire même des alphabets potentiellement infinis (ce dernier cas ne sera pas traité ici car nous ne nous intéressons qu'aux monoïdes libres finiment engendrés).

Définition 2.17 La *longueur* d'un mot correspond à la taille de la suite de lettres qui constituent le mot (chaque lettre à donc une taille de 1). De cette façon, la longueur d'un mot $u = u_1 \ldots u_n$ dans A^*, que l'on note $|u|$, est égale à n et on a pour tout mot v de A^* :

$$|uv| = |u| + |v|$$

Le mot vide 1_{A^*} est le seul mot de A^* de taille nulle. Cette notion de taille est également utilisée de façon plus subtile en se concentrant sur une lettre précise. De cette façon, on note $|w|_a$ le nombre d'occurrences de la lettre a présentes dans le mot w. Ainsi, pour tout mot u et v de A^*, on a :

$$|u| = \sum_{a \in A} |u|_a \quad \text{et} \quad |uv|_a = |u|_a + |v|_a$$

1. En réalité, un alphabet peut être infini mais nous traitons pas de ce cas ici.

EXEMPLE 2.10 Soit $A = \{a, b\}$ et $u = abbab$ un mot de A^*. On a $|u| = 5$, $|u|_a = 2$ et $|u|_b$.

Dans ce qui suit, la loi multiplicative étant clairement la concaténation, nous notons simplement uv le produit de deux éléments u et v. Grâce à l'unicité de la décomposition dans ce monoïde libre, on peut définir certaines notions qui n'auraient pas de sens autrement.

Définition 2.18 Soient u et v deux mots sur A. On dit que :
- u est un *préfixe* (ou facteur gauche) de v s'il existe w dans A^* tel que $v = uw$.
- u est un *suffixe* (ou facteur droit) de v s'il existe w dans A^* tel que $v = wu$.
- u est un *facteur* de v s'il existe w et w' dans A^* tels que $v = wuw'$.
- la suite de mots $(x_1, x_2, \ldots, x_n)$ est une *factorisation* de u si $u = x_1x_2 \ldots x_n$.
- u est un *sous-mot* de v s'il existe $u_1, \ldots, u_n, v_1, \ldots, v_n$ dans A^* tels que $u = u_1 \ldots u_n$ et $v = v_0u_1v_1u_2 \ldots x_nv_n$. Autrement dit, u s'obtient à partir de v en supprimant des lettres dans v.
- $u = u_1 \ldots u_n$ est l'*image miroir* de v si $v = u_n \ldots u_1$ où $u_1, \ldots, u_n$ sont dans A.

EXEMPLE 2.11 Pour le mot $u = abbab$ sur A^*, ab est à la fois un suffixe et un préfixe de u, bba est un facteur de u, bbb est un sous-mot de u, $babba$ est l'image miroir de u et (ab, ba, b) est une factorisation de u.

REMARQUE 2.5 Le mot vide 1_{A^*} est préfixe, suffixe et facteur de tout mot de A^*. On dit que u est un préfixe (*resp.* suffixe) propre de v, si $u = vw$ (*resp.* $u = wv$) avec w différent du mot vide. De même, on dit que $u = wvw'$ est un facteur propre de v si w et w' ne sont pas simultanément égaux au mot vide. De cette manière, un mot $u = u_1 \ldots u_n$ possède $n + 1$ préfixes (n préfixes propres) et $n + 1$ suffixes (n suffixes propres). Il est clair que le nombre de facteurs d'un mot ne dépend pas uniquement de sa longueur.

Il est possible de définir des relations d'ordre (partielles) sur les mots de A^* à l'aide de certaines de ces notions. Par exemple, l'ordre suffixe consiste à dire qu'un mot est plus petit qu'un autre (ou inférieur à un autre) s'il est suffixe de ce mot. Il existe également des relations d'ordre totales sur A^* qui sont construites à partir d'un ordre total sur l'alphabet A.

Définition 2.19 Soit $A = \{a_1, a_2, \ldots, a_n\}$ un alphabet totalement ordonné : $a_1 < a_2 < \cdots < a_n$ avec 1_{A^*} plus petit que n'importe laquelle des lettres de A. Pour tout $u = au'$ et $v = bu'$, on définit l'*ordre lexicographique* sur A^* que l'on note $\preceq$ de la façon suivante :

$$u \preceq v \iff \begin{cases} a < b & \text{ou} \\ a = b \ \text{ et } \ u' \preceq v' \end{cases}$$

Dans certains cas, il peut être utile de munir un monoïde d'une distance qui permet de mesurer l'écart entre deux éléments. Par exemple, dans le monoïde $(\mathbb{N}, +)$ il est naturel de choisir comme distance entre deux éléments la différence des deux en valeur absolue. Dans notre cas, la notion de distance se base sur l'idée de préfixe.

Définition 2.20 Soient u et v deux mots de A^*. On note $u \wedge v$ le plus grand préfixe commun entre u et v. On définit la *distance préfixe* de u et v par

$$d_p(u, v) = |u| + |v| - 2\,|u \wedge v|$$

Définition 2.21 On appelle *langage* sur A tout ensemble de mot écrits sur l'alphabet A ou, autrement dit, tout sous-ensemble de A^*. Le *langage complémentaire* d'un langage L, noté $\overline{L}$, est l'ensemble des mots de A^* qui n'appartiennent pas à L.

De la même façon que pour les mots, il n'y a pas de notion de sémantique dans le concept de langage tel que nous l'avons décrit plus haut. Ici un langage est une partie de A^* c'est-à-dire un élément de $\mathfrak{P}(A^*)$, il est donc naturel d'utiliser toutes les opérations ensemblistes "classiques" comme l'union, l'intersection, la complémentation, ...

EXEMPLE 2.12 Soit $A = \{a, b\}$ un alphabet :

- l'ensemble $\mathcal{L}_1 = \{u \in A^* \mid |u|_a \mod 2 = 0\}$ est le langage des mots contenant un nombre pair de a.
- l'ensemble $\mathcal{L}_2 = \{u \in A^* \mid |u|_b \mod 2 = 1\}$ est le langage des mots contenant un nombre impair de b.
- l'ensemble $\mathcal{L}_1 \cap \mathcal{L}_2$ (*resp.* $\mathcal{L}_1 \cup \mathcal{L}_2$) qui est l'intersection (*resp.* l'union) des langages $\mathcal{L}_1$ et $\mathcal{L}_2$ est le langage des mots qui contiennent un nombre pair de a et (*resp.* ou) un nombre impair de b.
- l'ensemble $\overline{\mathcal{L}_1}$ qui est le complémentaire du langage $\mathcal{L}_1$ est le langage des mots contenant un nombre impair de a.

En utilisant les opérations rationnelles que l'on a évoquées plus tôt, nous obtenons une définition très importante, celle des langages rationnels. Ils s'agit d'une classe de langages qui est énormément utilisée que ce soit en informatique, en linguistique ou en traitement de texte et nous verrons par la suite ses différentes propriétés.

Définition 2.22 L'ensemble des *langages rationnels* de A^* est le plus petit ensemble stable pour les opérations rationnelles et qui contient l'ensemble vide. Autrement dit, il s'agit de la clôture rationnelle des parties finies de A^* que l'on note Rat A^*.

L'ensemble des langages rationnels possède un certain nombre de propriétés intéressantes, dont en voici une partie :

- ils sont stables par morphisme, autrement dit l'image par morphisme d'un langage rationnel est également un langage rationnel.
- ils sont fermés par quotient gauche, autrement dit si L est un langage rationnel et u un mot, alors le quotient gauche de X par u est un langage rationnel. Cette propriété sera très importante dans la suite...
- ils sont stables par passage à l'image miroir, autrement dit si un langage L est rationnel alors le langage formé par les images miroir des mots de X est également rationnel.
- Si L est un langage rationnel, alors l'ensemble des préfixes, l'ensemble des suffixes ainsi que l'ensemble des facteurs des mots de L sont également des langages rationnels.

EXEMPLE 2.13 L'ensemble $\{a^n b^n \mid n \in \mathbb{N}\}$ est un langage qui n'est pas rationnel (il est algébrique). En effet, il n'est pas possible de l'exprimer à l'aide d'union, de produit ou d'étoile de langages rationnels.

Cette définition nous donne également un moyen plus concis et plus clair de décrire un langage rationnel à l'aide d'une formule. Pour cela, on utilise ce que l'on appelle des expressions rationnelles qui sont obtenues à partir des lettres de l'alphabet A, des constantes 0 et 1 et des opérations "rationnelles" $+$ pour l'union, . pour le produit (que l'on omet en général) et $*$ pour l'étoile. De cette façon, une expression rationnel e dénote un langage rationnel $L(e)$ défini récursivement comme suit :

- $L(0) = \emptyset$, $L(1) = \{1^{A^*}\}$ et $L(a) = \{a\}$ pour $a \in A$.
- $L(e + f) = L(e) \cup L(f)$
- $L(e.f) = L(e).L(f)$
- $L(e^*) = L(e)^*$

EXEMPLE 2.14 Les langages de l'exemple 12 s'expriment très simplement par des expressions rationnelles :

- $\mathcal{L}_1 = (b^*ab^*ab^*)^* = (b + ab^*a)^*$
- $\mathcal{L}_2 = (a^*ba^*ba^*ba^*)^* = (a + ba^*ba^*b)^*$
- $\overline{\mathcal{L}_1} = (b^*ab^*ab^*ab^*)^* = (b + ab^*ab^*a)^*$

Comme nous l'avons évoqué plus tôt dans le cas des monoïdes généraux, il n'y a pas équivalence entre les ensembles reconnaissables et les ensembles rationnels. Cela dit dans certains cas particuliers, cette propriété est vérifiée. C'est le cas dans les monoïdes libres finiment engendrés. Cette propriété sera étudiée plus en détails dans la suite de ces rappels.

2.4 Séries formelles et séries rationnelles

Dans cette partie, on introduit une généralisation des notions de langage et d'ensemble rationnel en associant des coefficients aux éléments d'un ensemble.

Définition 2.23 On appelle *série formelle*[2] sur un monoïde M à coefficients (ou à multiplicité) dans un semi-anneau $\mathbb{K}$, toute application de M dans $\mathbb{K}$. Pour toute série s, on note $< s, m >$ l'image d'un élément m de M par la série s (au lieu de $s(m)$) que l'on appelle également le *coefficient* de m dans s. L'ensemble des séries sur M à coefficients dans $\mathbb{K}$ est noté $\mathbb{K}\langle\langle M\rangle\rangle$.

Avec une telle définition, on voit aisément que les langages que nous avons évoqués précédemment sont en fait des séries de $\mathbb{B}\langle\langle A^*\rangle\rangle$. En général, on note $\mathbb{K}^M$ l'ensemble des applications de M dans $\mathbb{K}$. Cela dit, malgré la structure de semi-anneau qui découle naturellement de l'addition et de la multiplication de $\mathbb{K}$, nous allons utiliser une structure qui dérive de la structure de monoïde de M. Ceci justifie la différence de notation entre les séries et les applications. On note formellement une série s sur un monoïde M :

$$s = \sum_{m \in M} < s, m > m$$

EXEMPLE 2.15 Soit $\mathbb{K} = \mathbb{N}$ et $M = A^*$. Les séries s_a et s_b définies par :

$$\forall u \in A^*, \quad < s_a, u >= |u|_a \iff s_a = \sum_{u \in A^*} |u|_a \, u$$
$$\forall u \in A^*, \quad < s_b, u >= |u|_b \iff s_b = \sum_{u \in A^*} |u|_b \, u$$

2. par opposition aux séries analytiques où l'objectif est de les évaluer en un point ou d'étudier leur domaine de convergence.

associent, à chaque mot u de A^*, respectivement le nombre de a et le nombre de b de u.

Définition 2.24 Le *support d'une série* s de $\mathbb{K}\langle\langle M\rangle\rangle$, noté $Supp(s)$, est l'ensemble des éléments de M dont le coefficient est non nul :

$$Supp(s) = \{m \in M \mid < s, m > \neq 0_{\mathbb{K}}\}$$

Pour tout sous-ensemble N de M, on appelle *série caractéristique* de N, notée $\underline{N}$, la série dont le support est N et dont les coefficients non nuls valent $1_{\mathbb{K}}$:

$$\forall m \in m < \underline{N}, m >= \begin{cases} 1_{\mathbb{K}} & \text{si } m \in N \\ 0_{\mathbb{K}} & \text{sinon} \end{cases}$$

EXEMPLE 2.16 Le support de la série s_a (*resp.* s_b) de l'exemple 15 est le langage des mots qui contiennent au moins un a (*resp.* un b).

Une série dont le support est un ensemble fini est appelée un polynôme. L'ensemble des polynômes, noté $\mathbb{K}\langle M\rangle$, forme un sous-semi-anneau de $\mathbb{K}\langle\langle M\rangle\rangle$. Ces polynômes vont être à la base de la définition des séries rationnelles. Pour cela, il faut redéfinir les différentes opérations rationnelles.

Pour notre part, nous allons nous restreindre aux séries sur A^* pour plus de simplicité. Pour de plus amples détails sur les séries formelles sur un monoïde quelconque, se référer à [45].

Définition 2.25 Soit $(\mathbb{K}, \oplus, \otimes)$ un semi-anneau. On appelle *semi-anneau des séries* sur A^* à multiplicité dans $\mathbb{K}$, noté $\mathbb{K}\langle\langle A^*\rangle\rangle$, l'ensemble des applications de A^* dans $\mathbb{K}$ muni des opérations suivantes :

i) la multiplication à gauche et à droite par un scalaire k d'une série s que l'on note respectivement ks et sk :

$$\forall m \in M, < ks, m >= k < s, m > \text{ et } \forall m \in M, < sk, m >=< s, m > k$$

ii) l'addition de deux séries s et t que l'on note $s + t$:

$$\forall m \in M, < s + t, m >=< s, m > \oplus < t, m >$$

iii) le produit de deux séries s et t que l'on note st (appelé produit de Cauchy) :

$$\forall m \in M, < st, m >= \bigoplus_{xy=m} < s, x > \otimes < t, y >$$

REMARQUE 2.6 Le produit de deux séries fait intervenir une somme qui peut être potentiellement infinie si l'on se trouve dans un monoïde quelconque car il est possible qu'un élément possède un nombre infini de factorisations (ou décompositions). Ce n'est pas le cas ici car nous ne manipulerons que des séries sur le monoïde libre dans lequel chaque mot possède un nombre fini de factorisations qui ne dépend uniquement que de la taille du mot.

REMARQUE 2.7 Le produit qui découle "naturellement" de la multiplication (c'est-à-dire de la multiplication terme à terme) est le produit d'Hadamard qui n'est pas une opération rationnelle.

EXEMPLE 2.17 La somme des séries s_a et s_b de l'exemple 15, notée $s_a + s_b$ et définie par :

$$\forall u \in A^*, \quad < s_a + s_b, u >= |u|_a + |u|_b = |u|$$

est la série qui associe à chaque mot de A^* sa taille. Le support de cette série est A^+.

Définition 2.26 On appelle *terme constant* d'un série s de $\mathbb{K}\langle\langle A^*\rangle\rangle$ le coefficient du mot vide dans s : $c(s) =< s, 1_{A^*} >$. On dira qu'une série est propre si son terme constant est nul. De même, on appelle *partie propre* d'une série s la série s_p définit par :

$$< s_p, 1_{A^*} >= 0_{\mathbb{K}} \text{ et } \forall u \in A^+, < s_p, u >=< s, u >$$

L'objectif étant de redéfinir toutes les opérations rationnelles, on cherche évidemment à redéfinir l'opération étoile $(*)$ pour les séries. Il s'avère que le fait que l'étoile d'une série soit définie ou non dépend en grande partie de la définition de l'étoile dans le semi-anneau $\mathbb{K}$ et plus particulièrement de l'étoile de son terme constant.

Proposition 2.1 *L'étoile d'une série s de $\mathbb{K}\langle\langle A^*\rangle\rangle$ est définie si et seulement si l'étoile de son terme constant est définie (dans $\mathbb{K}$). Auquel cas, on a :*

$$s^* = (c(s)^* s_p)^* c(s)^*$$

Nous allons, pour notre part, manipuler uniquement des séries propres et il s'avère que l'étoile d'une série propre de $\mathbb{K}\langle\langle A^*\rangle\rangle$ est toujours définie.

En effet, pour une série s de $\mathbb{K}\langle\langle A^*\rangle\rangle$, étant donné qu'un mot u sur A^* n'a qu'un nombre fini de factorisations propres qui est limité par sa longueur, il ne peut être égal à un produit dont le nombre de facteurs dépasse $|u|$. Ce qui implique que dans n'importe qu'elle série s^n, puissance n-ième de s, si on a $n > |u|$ alors son coefficient est égal à $0_{\mathbb{K}}$:

$$\forall u \in A^*, \forall n > |u|, < s^n, u >= 0_{\mathbb{K}}$$

Cela implique que l'étoile du coefficient de n'importe lequel des éléments de A^* est une somme finie et donc est définie. De ce fait, l'étoile est définie sur toute série propre de $\mathbb{K}\langle\langle A^*\rangle\rangle$.

Maintenant que nous disposons de toutes ces opérations rationnelles, qui sont plus précisément des opérations $\mathbb{K}$-rationnelles, nous pouvons aisément définir la notion de série ($\mathbb{K}$-)rationnelles.

Définition 2.27 Un sous-ensemble de $\mathbb{K}\langle\langle A^*\rangle\rangle$ est rationnellement clos s'il est stable pour les opérations rationnelles. Une série de $\mathbb{K}\langle\langle A^*\rangle\rangle$ est rationnelle si elle appartient à la clôture rationnelle de l'ensemble des polynômes, $K\langle A^*\rangle$.

De la même façon que l'on note Rat A^* l'ensemble des langages rationnels, on note $\mathbb{K}$Rat A^* l'ensemble des séries rationnelles à coefficients dans $\mathbb{K}$ (que l'on appelle également séries $\mathbb{K}$-rationnelles) qui forme un sous-semi-anneau des séries formelles.

REMARQUE 2.8 De la même façon que pour les langages rationnels pour lesquels on dispose des expressions rationnelles pour les exprimer de façon concise, on dispose d'expressions $\mathbb{K}$-rationnelles pour exprimer les langages $\mathbb{K}$-rationnels.

De plus, et pas des moindres, de façon analogue au théorème de Kleene sur les langages rationnels et reconnaissables, on a une extension sur les séries sur le monoïde libre.

Théorème 2.2 *[Kleene-Schützenberger] Une série de $\mathbb{K}\langle\langle A^*\rangle\rangle$ est $\mathbb{K}$-rationnelle si et seulement si elle est $\mathbb{K}$-reconnaissable :*

$$\mathbb{K}Rat\ A^* = \mathbb{K}Rec\ A^*$$

Nous allons voir dans le chapitre sur les automates bidirectionnel à multiplicité le sens que nous donnons pour une série d'être $\mathbb{K}$-reconnaissable à l'aide d'un outil qui sera au centre de cette étude et que nous présentons dans la section suivante, les automates...

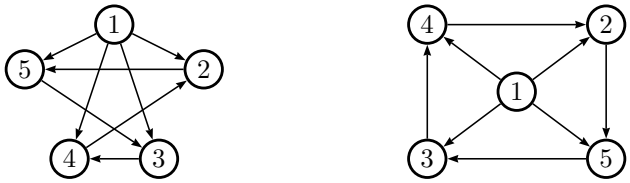


FIGURE 3.1 – Des représentations sagittales différentes pour une même relation.

3 Graphes et automates

3.1 Graphes orientés

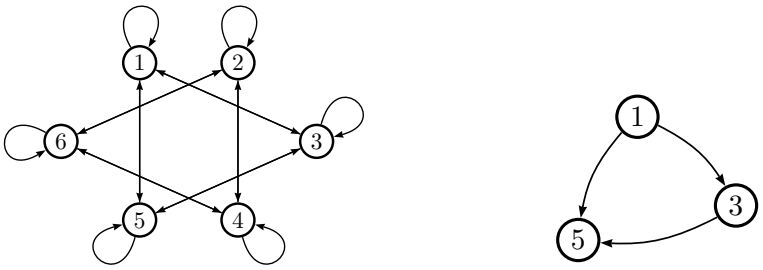
Les automates que nous allons manipuler reposent sur le concept de graphe orienté. Nous allons donc rapidement aborder le sujet car beaucoup de vocabulaire de la théorie des graphes est ré-utilisé pour les automates.

Définition 3.1 Un *graphe orienté* $\mathcal{G}$ est une paire $\langle Q, E\rangle$, où Q est l'ensemble des sommets et E l'ensemble des arcs. Concrètement, les arcs se composent d'un sommet d'origine et d'un sommet d'arrivée (arcs orientés) que l'on note sous forme de paire. Formellement, l'ensemble des arcs est en fait un sous-ensemble de $Q \times Q$ ce qui fait de $\mathcal{G}$ la représentation d'une relation de Q dans lui-même.

Au niveau de la représentation[3], on utilise des cercles pour représenter les sommets d'un graphe (dans lesquels on peut indiquer l'éventuel nom du sommet) et des flèches pour représenter les arcs (la pointe de la flèche arrivant logiquement sur le sommet d'arrivée de l'arc). Comme on peut le voir sur la figure 3.1 un même graphe peut avoir des représentations sagittales très différentes (en apparence). On utilisera deux fonctions σ et τ qui associe à chaque arc (p, q) respectivement son sommet d'origine p et son sommet d'arrivée q.

On dira qu'un graphe est fini si l'ensemble de ses sommets est fini. Dans ce cas, il est également naturel de représenter un graphe $\mathcal{G} = \langle Q, E\rangle$, avec $|Q| = n$, par une matrice (booléenne) de dimensions $n \times n$ que l'on appelle matrice d'adjacence[4]. Pour cela, on suppose que les sommets de $\mathcal{G}$ sont numérotés de façon arbitraire $q_1, \ldots, q_n$, et dans ce cas, le coefficient (p_i, p_j) de la matrice d'adjacence vaut 1 s'il existe un arc d'origine p_i et d'arrivée p_j

3. le terme exact est représentation sagittale
4. Il existe également une représentation dite par dictionnaire que nous n'aborderons pas ici



(a) (b)

FIGURE 3.2 – Le graphe $\mathcal{G}$ et un de ses sous-graphes $\mathcal{G}'$.

dans le graphe.

Définition 3.2 Soit $\mathcal{G} = \langle Q, E \rangle$ un graphe. Un graphe $\mathcal{G}' = \langle Q', E' \rangle$ est un *sous-graphe* de $\mathcal{G}$ si $E' \subseteq E$ et $Q' \subseteq Q$.

EXEMPLE 3.1 Le graphe $\mathcal{G} = \langle Q, E \rangle$ de la figure 3.2a avec $Q = \{1, 2, 3, 4, 5, 6\}$ dont la matrice d'adjacence est

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

représente la relation sur Q telle que :

$$\forall p, q \in Q, \quad (p, q) \in E \iff p + q \text{ est pair}$$

Le graphe $\mathcal{G}' = \langle Q', E' \rangle$ de la figure 3.2b avec $Q' = \{1, 3, 5\}$ est un sous-graphe de $\mathcal{G}$ et il représente la relation sur Q' telle que :

$$\forall p, q \in Q', \quad (p, q) \in E' \iff p < q \text{ ET } p + q \text{ est pair}$$

Définition 3.3 Un *chemin* dans un graphe $\mathcal{G} = \langle Q, E \rangle$ est une suite d'arcs $(e_1, \ldots, e_n)$ tels que $\tau(e_i) = \sigma(e_{i+1})$ pour tout $i < n$. La longueur d'un chemin est égale au nombre d'arcs qui le forment. Pour tout sommet p de $\mathcal{G}$, il existe un chemin de longueur 0 dont le sommet d'origine et celui d'arrivée est p.

REMARQUE 3.1 On considère la matrice d'adjacence M d'un graphe $\mathcal{G} = \langle Q, E \rangle$ tel que ses sommets sont numérotés de 1 à n. Il existe un chemin de longueur exactement k qui a pour origine le sommet i et pour arrivée le sommet j si et seulement si le coefficient (i, j) de la matrice M^k est non nul.

Définition 3.4 S'il existe un chemin d'un sommet p à un sommet q, on dit que p est un *ancêtre* de q et que q est un *successeur* de p. Si ce chemin est de longueur 1, alors p est un *ancêtre direct* de q et q est un *successeur direct* de p. Un chemin $(e_1, \ldots, e_n)$ dans $\mathcal{G}$ est un *circuit* (on parle également de *cycle*) si $n > 0$ et $\sigma(e_1) = \tau(e_n)$. Un circuit est dit *élémentaire* s'il ne contient pas de cycle de taille inférieure. Un graphe qui ne contient aucun cycle est dit *acyclique*.

EXEMPLE 3.2 Soit $\mathcal{G} = \langle Q, E \rangle$ le graphe de la figure 3.2a. La suite d'arcs $((2, 4), (4, 6), (6, 6), (6, 4))$ est un chemin dans $\mathcal{G}$ du sommet 2 au sommet 6 qui contient un circuit élémentaire (l'arc $(6, 6)$) et un circuit (non-élémentaire) composé des 4 derniers arcs.
Le graphe $\mathcal{G}'$ de la figure 3.2b ne contient, lui, aucun cycle et est donc acyclique.

Cette notion de chemin permet de fragmenter l'ensemble des sommets d'un graphe $\mathcal{G}$ en fonction de leur accessibilité ce qui se traduit par une relation (d'équivalence) qui associe deux sommets p et q si et seulement s'il existe un chemin de p à q et un chemin de q à p dans $\mathcal{G}$ (autrement dit, si p et q appartiennent à un même cycle). Les classes de Q modulo cette équivalence forment ce que l'on nomme des composantes fortement connexes.

Définition 3.5 Soit $\mathcal{G} = \langle Q, E \rangle$ un graphe. Une *composante fortement connexe* de $\mathcal{G}$ est un sous-graphe maximal de $\mathcal{G}$ tel que pour toute paire de sommets, il existe un cycle qui les contient tous les deux. Un graphe est dit *fortement connexe* s'il est composé d'une unique composante fortement connexe.

Autrement dit, une composante fortement connexe consiste en un ensemble de sommets tel qu'il existe un chemin de n'importe lequel de ces sommets vers n'importe quel autre sommet de l'ensemble. Avec une telle définition on comprend également que pour chaque sommet du graphe, il existe une composante fortement connexe triviale composée du sommet lui-même (sans arc). Ainsi on appellera pelote, une composante fortement connexe non

triviale, c'est à dire qui contient au moins un arc.

EXEMPLE 3.3 Le graphe de la figure 3.2a est fortement connexe. Ce n'est pas le cas du graphe de la figure 3.2b qui, lui, ne possède même pas une seule composante fortement connexe non triviale.

Définition 3.6 Soit $\mathcal{G} = \langle Q, E \rangle$ et $\mathcal{G}' = \langle Q', E \rangle$ deux graphes. Une application φ de Q dans Q' est un *morphisme de graphes* si l'image de tout arc dans $\mathcal{G}$ est un arc dans $\mathcal{G}'$. Formellement, φ est un morphisme (de graphes) si il s'agit d'une application de Q dans Q' telle que :

$$\forall (p, q) \in E, \quad (\varphi(p), \varphi(q)) \in E'$$

3.2 Automates génériques

Il est possible d'étudier les langages sous une autre forme qu'une structure algébrique. Nous allons donner les définitions de base de ce que l'on appelle un automate ce qui va nous permettre de représenter "graphiquement" un langage et de l'étudier. Un automate se présente sous la forme d'un graphe orienté et étiqueté dans lequel les sommets peuvent posséder deux nouvelles propriétés, un caractère initial et/ou un caractère final, ce qui va nous amener à redéfinir le concept de langage "reconnaissable". On ne parlera plus d'arc mais de transition de même que l'on parlera d'état et non plus de sommet.

Les automates tels que nous allons les manipuler (dit "classiques") sont en quelque sorte "simplifiés" par rapport au concept général et il ne s'agit que d'un cas particulier des automates "génériques". Nous allons d'abord présenter le cas des automates sur un semi-anneau ainsi que sur un monoïde pour ensuite se pencher sur le cas particulier des automates sur le monoïde libre.

Dans tout ce qui suit, M est un monoïde dont l'opération de multiplication est notée $\cdot$ et $\mathbb{K}$ désigne un semi-anneau avec $\oplus$ comme addition et $\otimes$ comme produit.

Automates sur un semi-anneau

Définition 3.7 Un automate $\mathcal{A}$ sur un semi anneau $\mathbb{K}$ est un tuple $\langle Q, \mathbb{K}, E, I, T\rangle$ tel que :
- Q est un ensemble non vide d'éléments, appelé ensemble des états.
- E est une matrice de $\mathbb{K}^{Q\times Q}$ appelée matrice de transition.
- I est un vecteur (ligne) de $\mathbb{K}^{Q}$ appelé vecteur initial.
- T est un vecteur (colonne) de $\mathbb{K}^{Q}$ appelé vecteur final.

On dit que l'on a une transition t de p à q étiquetée par k, que l'on notera $t = p \xrightarrow{k} q$, dans $\mathcal{A}$ si et seulement si $E(p, q) = k$ avec $k \neq 0_{\mathbb{K}}$. Graphiquement, cette transition est représentée comme un arc partant de l'état p et arrivant sur l'état q et au-dessus duquel on indique l'étiquette de t.
Un état initial p de Q est un état tel que $I(p) \neq 0_{\mathbb{K}}$. Ils sont toujours notés avec une flèche entrante (pointant vers l'état).
Un état final f de Q est un état tel que $T(f) \neq 0_{\mathbb{K}}$. Pour les états finals[5], il y a deux notations possibles et chacune d'elles sera utilisée dans ce mémoire : la première consiste à utiliser un double cercle pour représenter l'état final et sera utilisée dans la partie concernant les automates sur les mots infinis dans laquelle le rôle des états initiaux et finals n'est pas symétrique. La seconde notation exprime clairement la dualité avec les états initiaux en utilisant une flèche sortante (partant de l'état final et pointant "dans le vide") et sera utilisée dans tous les autres cas (elle permet notamment de spécifier plus facilement une étiquette pour un état final).

Définition 3.8 Soit $\mathcal{A} = \langle Q, \mathbb{K}, E, I, T\rangle$ un automate sur $\mathbb{K}$. Le *graphe sous-jacent* de $\mathcal{A}$ est le graphe $\mathcal{G} = \langle Q, E\rangle'$ tel que $E' = \{(p, q) \mid E(p, q) \neq 0_{\mathbb{K}}\}$

Dans un automate, les opérations dans le semi-anneau des étiquettes nous amène à parler de calcul plutôt que de chemin comme c'est le cas dans les graphes.

Définition 3.9 Soit $\mathcal{A} = \langle Q, \mathbb{K}, E, I, T\rangle$ un automate sur $\mathbb{K}$. Un *calcul* dans $\mathcal{A}$ correspond à un chemin dans le graphe sous-jacent, c'est à dire à une suite d'états $(p_0, \ldots, p_n)$ tels qu'il y a une transition entre p_i et p_{i+1} pour tout i dans $[0; n-1]$. La longueur d'un tel calcul est égale à la longueur du chemin sous-jacent. Un calcul est *réussi* si p_0 est un état initial et si p_n est un état final. L'*étiquette* d'un calcul c, notée $|c|$, correspond au produit des étiquettes de ses transitions, c'est à dire qu'on a $|c| = E_{p_0,p_1} \otimes E_{p_1,p_2} \otimes \cdots \otimes E_{p_{n-1},p_n}$.

5. ou finaux mais cette version est moins utilisée

La valeur d'un tel calcul est égale à $I_{p_0} \otimes |c| \otimes T_{p_n}$. L'élément de $\mathbb{K}$ reconnu par l'automate $\mathcal{A}$, que l'on note $|\mathcal{A}|$, correspond à la somme (dans $\mathbb{K}$[6]) des calculs réussis dans $\mathcal{A}$.

On étend la notation $p \xrightarrow{k} q$ aux calculs dans un automate, autrement dit, si $c = (p_0, \ldots, p_n)$ est un calcul dans un automate $\mathcal{A}$ tel que $E(p_i, p_{i+1}) = k_i$ pour tout i dans $[1; n-1]$, on note $c = p_0 \xrightarrow{k_0} p_1 \xrightarrow{k_1} p_2 \ldots p_{n-1} \xrightarrow{k_{n_1}} p_n$. Par soucis de concision, on notera également $c = p_0 \xrightarrow{k} p_n$ avec $k = k_0 \otimes k_1 \otimes \cdots \otimes k_{n-1}$.

Définition 3.10 Un calcul dans un automate est un *circuit* si le chemin correspondant dans le graphe sous-jacent est un circuit. Un automate est dit *acyclique* si son graphe sous-jacent est acyclique.

Avec la représentation matricielle pour les transitions, on peut aisément faire le lien avec les étiquettes des calculs dans l'automate. En effet si un état p est à la fois initial et final alors on a un calcul de longueur nulle et de valeur $I(p) \otimes T(p)$. De cette façon, on peut exprimer la somme de tous les calculs de longueur nulle par

$$I \otimes T = \bigoplus_{p \in Q} I(p) \otimes T(p)$$

De la même manière, en utilisant en plus le produit matriciel (classique), on obtient la somme des calculs de longueur 1 par

$$I \otimes E \otimes T = \bigoplus_{p,q \in Q} I(p) \otimes E_{p,q} \otimes T(q)$$

Cette forme de calcul se généralise pour toutes les longueurs de calcul et on obtient la somme des calculs de longueur k par $I \otimes E^k \otimes T$.

Cette dernière constatation mène logiquement à l'élément reconnu par un automate $\mathcal{A}$ et qui correspond à la somme des calculs de longueur quelconque :

$$\bigoplus_{n \in \mathbb{N}} I \otimes E^n \otimes T = I \left(\bigoplus_{n \in \mathbb{N}} E^n \right) \otimes T = I \otimes E^* \otimes T$$

L'élément reconnu par un automate est donc défini si et seulement si l'étoile de E est définie.

REMARQUE 3.2 Il est tout à fait possible de "supprimer" les étiquettes des transitions initiales et finales d'un automate $\mathcal{A}$ tout en conservant le comportement de l'automate (*i.e.* de sorte que l'automate reconnaisse le même

6. Cette somme est potentiellement infinie et donc peut ne pas être définie dans $\mathbb{K}$

élément). Pour cela, on ajoute un état i_0 tel que $I(i_0) = 1_{\mathbb{K}}$ et ensuite, pour tout état $i \neq i_0$ de $\mathcal{A}$ tel que $I(i) \neq 0_{\mathbb{K}}$, on ajoute une transition $i_0 \xrightarrow{I(i)} i$. Pour les transitions finales, le procédé est symétrique.
De cette manière, I et T ne sont plus dans $\mathbb{K}^Q$ mais dans $\{0_{\mathbb{K}}, 1_{\mathbb{K}}\}^Q$ (des vecteurs "booléens") ce qui va nous permettre de les traiter comme des ensembles d'états.
Désormais nous utiliserons uniquement des automates dont les transitions initiales et finales sont "sans étiquette" (autrement dit étiquettées par $1_{\mathbb{K}}$).

Il est souvent très pratique et plus commode de considérer les transitions, y compris celles initiales et finales, non plus comme des matrices ou des vecteurs, mais comme des ensembles d'éléments. Ainsi la matrice des transitions peut être vue comme un sous-ensemble E de $Q \times M \times Q$ tel que

$$E = \{(p, k, q) \mid p, q \in Q, k \neq 0_{\mathbb{K}}\}$$

Ainsi, E est muni canoniquement de trois applications σ,τ et λ telles que pour toute transition $t = (p, k, q)$, $\sigma(t) = p$, $\lambda(t) = k$ et $\tau(t) = q$. De la même manière, grâce à la remarque 2 qui permet de "supprimer" les étiquettes des transitions initiales et finales, le vecteur initial et le vecteur final peuvent être vue comme des sous-ensembles I et T de Q tels que

$$I = \{p \in Q \mid I(p) = 1_{\mathbb{K}}\} \quad \text{et} \quad T = \{p \in Q \mid T(p) = 1_{\mathbb{K}}\}$$

L'ensemble des transitions de $\mathcal{A}$ correspond donc au support de la matrice de transition E. On peut également noter qu'entre deux états, il ne peut y avoir, au plus, qu'une seule transition. En effet, si l'on disposait de deux transitions (p, k, q) et (p, k', q), il est aisé de voir que l'on pourrait les "fusionner" en une unique transition dont l'étiquette serait $k \oplus k'$.

Définition 3.11 Deux automates sur un semi-anneau $\mathbb{K}$ sont *équivalents* s'ils reconnaissent le même élément de $\mathbb{K}$.

Avec ces définitions, on comprend bien l'importance des états initiaux et finals qui sont nécessaires pour exprimer le concept de calcul réussi. On peut, dans cette optique, exprimer le fait qu'un état soit atteignable (ou non) à partir d'un état initial ou qu'il puisse atteindre (ou non) un état final.

Définition 3.12 Soit $\mathcal{A}$ un automate sur $\mathbb{K}$. Un état p de $\mathcal{A}$ est dit *accessible* s'il existe un calcul d'un état initial de $\mathcal{A}$ vers p. De même, un état est dit *co-accessible* s'il existe un calcul de p vers un état final de $\mathcal{A}$. Un automate est dit *émondé* si tous ses états sont à la fois accessibles et co-accessibles.

Lorsque ces deux notions sont couplées, elles permettent de caractériser les états "utiles" d'un automate, autrement dit les états de l'automate qui

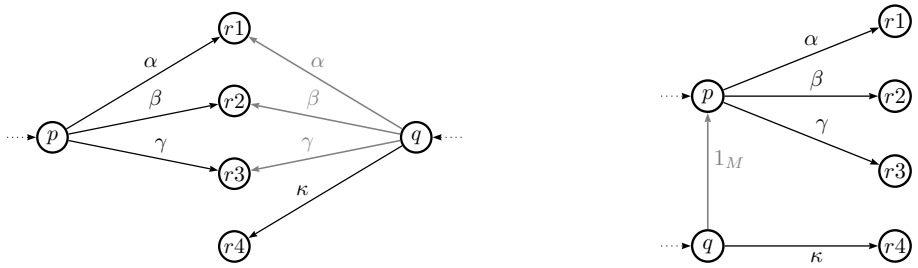


FIGURE 3.3 – Réduire le nombre de transitions.

appartiennent à au moins un calcul réussi. Il est clairement possible de ne conserver que les états utiles d'un automate (c'est à dire la partie émondée de l'automate) et donc de supprimer tout état qui ne soit pas à la fois accessible et co-accessible sans pour autant changer l'ensemble reconnu.

Proposition 3.1 *Pour tout automate sur $\mathbb{K}$, il existe un automate émondé équivalent.*

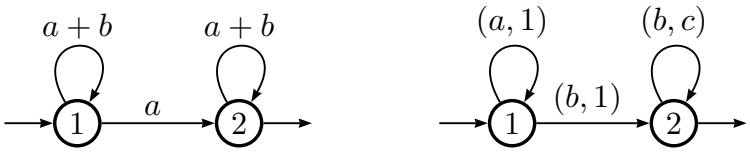
Automates sur un monoïde

En se restreignant au semi-anneau $\mathfrak{P}(M)$ des parties d'un monoïde M, il est possible de définir des concepts et des propriétés sur les automates qui n'auraient que peu de sens et encore moins d'utilité sur un semi-anneau quelconque.

Définition 3.13 Un automate $\mathcal{A}$ sur un monoïde M est un tuple $\langle Q, M, E, I, T\rangle$, où Q est l'ensemble fini des états, où E est une matrice de Rat $M^{Q\times Q}$ et où I et T sont des vecteurs, respectivement initial et final, dans Rat M^Q.

REMARQUE 3.3 Si l'étiquette d'une transition d'un état p à un état q est une union d'éléments de M, il est possible de "fractionner" cette transition en différentes transitions, toujours de p à q, ayant chacune pour étiquette un des éléments de l'union.

REMARQUE 3.4 Il est parfois utile de recourir à des transitions étiquetées par 1_M (l'élément neutre pour le produit) pour des raisons de facilité d'écriture ou pour condenser un automate en évitant un trop plein de transitions qui ne seront pas indiquées mais qui pourront être déduites. Ces transitions, dites spontanées, correspondent au passage d'un état à un autre sans modification



(a) L'automate $\mathcal{A}_1$ (b) L'automate $\mathcal{A}_2$

FIGURE 3.4 – Deux automates sur des monoïdes différents.

de l'étiquette. La figure 3.3 illustre ce principe : α, β, γ et κ sont des éléments du monoïde et les transitions (p, α, r_1), (p, β, r_2) et (p, γ, r_3) sont les uniques transitions sortantes de p. On peut remplacer les transitions grisées partant de q dans la figure 3.3a par l'unique transition spontanée partant de q (en grisé) dans la figure 3.3b.

Le caractère particulier de l'union ensembliste, qui va ici servir de somme, permet une nouvelle approche de la reconnaissance par automate en découpant l'élément (de $\mathfrak{P}(M)$) reconnu par ce dernier en terme d'élément accepté. En conservant les mêmes définitions que pour le cas des automates génériques[7] mais en ajoutant cette nouvelle notion d'acceptance, il est possible de définir de nouvelles propriétés.

Définition 3.14 Soit $\mathcal{A}$ un automate sur un monoïde M. Un élément m de M est *accepté* par $\mathcal{A}$ s'il existe un calcul réussi c dans $\mathcal{A}$ tel que $|c| = m$. L'élément de $\mathfrak{P}(M)$ reconnu par $\mathcal{A}$ correspond à l'ensemble des éléments acceptés par $\mathcal{A}$:

$$|\mathcal{A}| = \{m \in M \mid \exists i \in I, f \in T, i \xrightarrow{m} f\}$$

EXEMPLE 3.4 L'ensemble des éléments acceptés par l'automate $\mathcal{A}_1$ sur $\mathfrak{P}(A^*)$ décrit par la figure 3.4a est l'ensemble des mots sur A qui possèdent au moins une occurrence de la lettre a. L'automate $\mathcal{A}_2$ sur $\{a, b\}^* \times \{c\}^*$ décrit par la figure 3.4b reconnait l'ensemble $\{(a^m b^{n+1}, c^n) \mid n, m \in \mathbb{N}\}$.

Ce nouveau concept d'élément accepté va être très important dans la suite et va nous permettre d'exprimer de nouvelles propriétés sur les automates que nous étudions. Un des premiers exemples est le concept d'ambiguïté qui sera très largement utilisé, notamment dans le chapitre traitant des automates bidirectionnels.

7. Cela inclut le fait de pouvoir manipuler les éléments E,I et T comme des ensembles plutôt que sous leur forme linéaire.

Définition 3.15 Un automate est dit *non-ambigu* si pour tout élément m de M il existe au plus un calcul réussi dont l'étiquette est m.

Pour le moment il n'est pas possible d'entrer plus dans les détails du fait du caractère quelconque du monoïde M. On peut seulement déduire que bien qu'il puisse y avoir plusieurs factorisations possibles de m, dans un automate non-ambigu il ne peut y avoir qu'une unique factorisation qui étiquette un calcul étiqueté par m (si m est accepté par $\mathcal{A}$ évidemment). Cette notion d'ambiguïté, qui sera par la suite étendue dans le cas du monoïde libre en des notions plus précises, est très importante dans les modèles informatiques car elle introduit l'unicité d'un calcul et par la suite le déterminisme.

EXEMPLE 3.5 L'automate décrit par la figure 3.4a est ambigu. En effet, pour tout mot u de A^*, il existe $|u|_a$ calculs réussis qui reconnaissent u.
En revanche celui de la figure 3.4b est non-ambigu : on lit d'abord des a puis des b (couplés au c) sans aucun choix possible pour une étiquette donnée..

On souhaite également découper les calculs d'un automate en fonction des états qu'ils empruntent, ce qui nous amène aux définitions suivantes...

Définition 3.16 Soit $\mathcal{A} = \langle Q, M, E, I, T\rangle$ un automate sur M. Le *passé* d'un état p dans $\mathcal{A}$, noté $\mathsf{Past}_{\mathcal{A}}(p)$, correspond à l'ensemble des éléments qui étiquettent un calcul d'un état initial i à p, autrement dit, on a :

$$\mathsf{Past}_{\mathcal{A}}(p) = \{m \in M \mid \exists i \in I, i \xrightarrow{m} p\}$$

Le *futur* d'un état p dans $\mathcal{A}$, noté $\mathsf{Fut}_{\mathcal{A}}(p)$, correspond à l'ensemble des éléments qui étiquettent un calcul de p à un état final, autrement dit, on a :

$$\mathsf{Fut}_{\mathcal{A}}(p) = \{m \in M \mid \exists f \in T, p \xrightarrow{m} f\}$$

L'ensemble $\mathsf{Trans}_{\mathcal{A}}(p, q)$ correspond à l'ensemble des éléments de M qui étiquettent un calcul de p à q dans $\mathcal{A}$, autrement dit, on a :

$$\mathsf{Trans}_{\mathcal{A}}(p, q) = \{m \in M \mid p \xrightarrow{m} q\}$$

On peut également voir ces notions comme des redéfinitions des états initiaux et finals : le passé d'un état p correspond à l'ensemble des éléments acceptés par l'automate lorsque l'ensemble des états finals est réduit à l'unique élément p tandis que le futur d'un état correspond à l'ensemble des éléments acceptés par l'automate lorsque l'ensemble des états initiaux est réduit à

l'unique état p. L'ensemble $\mathsf{Trans}_{\mathcal{A}}(p,q)$ correspond à l'ensemble des éléments acceptés par l'automate lorsque $I = \{p\}$ et $T = \{q\}$.

EXEMPLE 3.6 Dans l'automate $\mathcal{A}_1$ représenté à la figure 3.4a, on a les ensembles suivants :

$$\mathsf{Past}_{\mathcal{A}}(1) = \mathsf{Trans}_{\mathcal{A}}(1,1) = \mathsf{Trans}_{\mathcal{A}}(2,2) = \mathsf{Fut}_{\mathcal{A}}(2) = A^*$$

$$\mathsf{Past}_{\mathcal{A}}(2) = \mathsf{Trans}_{\mathcal{A}}(1,2) = \mathsf{Fut}_{\mathcal{A}}(1) = A^*aA^*$$

Pour l'automate $\mathcal{B}$ de la figure 3.4b, on a les ensembles suivants :

$$\mathsf{Past}_{\mathcal{B}}(1) = \{(a^n, 1) \mid n \in \mathbb{N}\}, \qquad \mathsf{Fut}_{\mathcal{B}}(2) = \{(b^n, c^n) \mid n \in \mathbb{N}\}$$

$$\mathsf{Past}_{\mathcal{B}}(2) = \mathsf{Fut}_{\mathcal{B}}(1) = \{(a^m b^{n+1}, c^n \mid m, n \in \mathbb{N}\}$$

Avec ces définitions, pour tout p et q dans Q, $\mathsf{Past}_{\mathcal{A}}(p) \cdot \mathsf{Fut}_{\mathcal{A}}(p)$ et $\mathsf{Past}_{\mathcal{A}}(p) \cdot \mathsf{Trans}_{\mathcal{A}}(p,q) \cdot \mathsf{Fut}_{\mathcal{A}}(q)$ sont des sous-ensembles de l'ensemble des éléments acceptés par $\mathcal{A}$. On peut exprimer l'élément reconnu par $\mathcal{A}$ en terme de passé et de futur :

$$\begin{aligned} |\mathcal{A}| &= \bigcup_{i \in I} \mathsf{Fut}_{\mathcal{A}}(i) \\ &= \bigcup_{f \in T} \mathsf{Past}_{\mathcal{A}}(f) \\ &= \bigcup_{p \in Q} \mathsf{Past}_{\mathcal{A}}(p) \cdot \mathsf{Fut}_{\mathcal{A}}(p) \\ &= \bigcup_{p,q \in Q} \mathsf{Past}_{\mathcal{A}}(p) \cdot \mathsf{Trans}_{\mathcal{A}}(p,q) \cdot \mathsf{Fut}_{\mathcal{A}}(q) \end{aligned}$$

Il apparait clairement que l'ensemble reconnu par un automate sur M est rationnel étant donné qu'il est formé à partir de produits et d'unions d'ensembles eux-mêmes rationnels (les étiquettes des transitions). A l'inverse, il est possible de construire à partir de n'importe qu'elle expression rationnelle E sur M un automate qui reconnait $|E|$. Ce qui nous amène à une nouvelle définition d'ensemble rationnel.

Théorème 3.2 *Une partie P d'un monoïde M est rationnelle si et seulement il existe un automate fini sur M qui reconnait P.*

REMARQUE 3.5 Comme nous l'avons dit précédemment, il y a une différence entre ensemble reconnaissable (par morphisme) et ensemble rationnel. Prenons l'exemple du monoïde $(\mathbb{N}^2, +)$ et le langage $\{(n,n) \mid n \in \mathbb{N}\}$ reconnu par l'automate de la figure 3.5. Bien que ce langage, que l'on peut

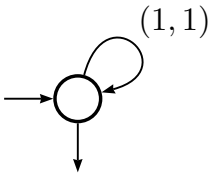


FIGURE 3.5

également exprimer par l'expression rationnelle $(1,1)^*$, soit donc bien rationnel, il n'est pas reconnaissable par morphisme. En effet, considérons le morphisme φ de $\mathbb{N}^2$ dans un monoïde $(M,\cdot)$ tel qu'il existe un sous-ensemble P de M avec $\varphi^{-1}(P) = (1,1)^*$. S'il existe deux entiers x et y distincts tels que $\varphi(x,0) = \varphi(y,0)$ alors on a

$$\begin{aligned}\varphi(x,y) &= \varphi(x,0)\cdot\varphi(y,0)\\ &= \varphi(y,0)\cdot\varphi(0,y)\\ &= \varphi(y,y) \in P\end{aligned}$$

Or (x,y) n'est pas dans $(1,1)^*$. Donc tous les éléments de la forme $(x,0)$ ont tous des images différentes par φ ce qui implique que M est infini. On en déduit que $(1,1)^*$ n'est pas reconnaissable par morphisme. Cela dit cette nouvelle approche nous permet d'envisager les ensembles rationnels sous un angle plus concret et plus facilement manipulable que par le biais des opérations rationnelles.

De façon analogue aux morphismes de monoïdes, les morphismes d'automates vont être des outils très précieux et vont nous permettre d'utiliser la puissance des méthodes algébriques pour analyser les ensembles décrits par les automates sur un monoïde. Á la différence des morphismes de graphes qui sont simplement des applications entre les sommets et qui impliquent une "conservation" des arcs, l'accent est mis sur le fait qu'un morphisme d'automates est avant tout une application entre les transitions.

Définition 3.17 Soient $\mathcal{A} = \langle Q, M, E, I, T\rangle$ et $\mathcal{B} = \langle R, M, F, J, U\rangle$ deux automates sur M. Un couple d'applications (toutes deux notées φ) $\varphi\colon R \to Q$ et $\varphi\colon F \to E$ est un *morphisme d'automates* de $\mathcal{B}$ dans $\mathcal{A}$ si

- $\forall (p,m,q) \in F$, $(\varphi(p), m', \varphi(q)) \in E$.
- $\varphi(J) \subseteq I$ et $\varphi(U) \subseteq T$.

Ces conditions permettent d'assurer la propriété principale d'un morphisme d'automates.

Proposition 3.3 *S'il existe un morphisme $\varphi\colon \mathcal{B} \to \mathcal{A}$ de $\mathcal{B}$ dans $\mathcal{A}$, alors l'élément reconnu par $\mathcal{B}$ est inclus dans celui reconnu par $\mathcal{A}$,* i.e. $|\mathcal{B}| \subseteq |\mathcal{A}|$.

REMARQUE 3.6 Les morphismes d'automates sont stables pour la composition (notée $\circ$). Ainsi si $\varphi\colon \mathcal{B} \to \mathcal{A}$ et $\mu\colon \mathcal{A} \to \mathcal{C}$ sont des morphismes d'automates, alors $\mu \circ \varphi\colon \mathcal{B} \to \mathcal{C}$ est également un morphisme d'automates.

Il semble, a priori, lourd de charger la définition d'un morphisme d'automates avec une application sur les transitions alors qu'une définition ne comportant qu'une application sur les états serait plus simple cela dit, comme nous allons le voir, cette surcharge est nécessaire car elle va nous permettre de travailler de façon plus subtile. En effet, en précisant des propriétés sur les transitions qui seraient difficilement abordables sans cela, on va pouvoir définir des classes de morphismes avec certaines particularités intéressantes... Avant d'en disposer, nous avons besoin de définir un certain nombre de propriétés locales sur les morphismes d'automates.

Dans ce qui suit, $\mathcal{A} = \langle Q, M, E, I, T\rangle$ et $\mathcal{B} = \langle R, M, F, J, U\rangle$ sont deux automates sur M et $\varphi\colon \mathcal{B} \to \mathcal{A}$ est un morphisme d'automates.

Définition 3.18 Pour tout état p de $\mathcal{A}$, on note $Out_{\mathcal{A}}(p)$ l'ensemble des transitions sortantes de p, autrement dit

$$Out_{\mathcal{A}}(p) = \{(p, m, q) \mid (p, m, q) \in E\}$$

Pour tout état p de $\mathcal{A}$, on note $In_{\mathcal{A}}(p)$ l'ensemble des transitions arrivant sur p, autrement dit

$$In_{\mathcal{A}}(p) = \{(q, m, p) \mid (q, m, p) \in E\}$$

Définition 3.19 Le morphisme φ de $\mathcal{B}$ dans $\mathcal{A}$ est dit *localement surjectif* si quel que soit r dans R, la restriction de φ à $Out_{\mathcal{B}}(r)$ est surjective de $Out_{\mathcal{B}}(r)$ dans $Out_{\mathcal{A}}(\varphi(r))$. De même, φ est dit *localement injectif* si quel que soit r dans R, la restriction de φ à $Out_{\mathcal{B}}(r)$ est injective de $Out_{\mathcal{B}}(r)$ dans $Out_{\mathcal{A}}(\varphi(r))$. Enfin, φ est dit *localement bijectif* s'il est à la fois localement surjectif et localement injectif.

Le fait d'être localement surjectif implique en particulier (sans rentrer dans les détails) que $\varphi^{-1}(T) = U$ et que si un état p de $\mathcal{A}$ est initial alors il y a au moins un état initial dans $\varphi^{-1}(p)$. La locale injectivité de φ implique,

quant à elle, que si p est un état initial de $\mathcal{A}$ alors il y a au plus un état initial dans $\varphi^{-1}(p)$.
Les définitions de morphisme localement co-surjectif, localement co-injectif et localement co-bijectif sont les versions duales des précédentes en se basant non plus sur les transitions sortantes mais sur les transitions entrantes (sur In au lieu de Out). Le fait d'être localement co-surjectif implique que $\varphi^{-1}(I) = J$ et que quel que soit t dans T, il existe au moins un u dans U tel que $\varphi(u) = t$. Alors que le fait d'être localement co-injectif implique qu'il existe au plus un u dans U tel que $\varphi(u) = t$.

La propriété de locale surjectivité va entraîner l'inclusion de l'élément reconnu par $\mathcal{A}$ dans celui reconnu par $\mathcal{B}$ et donc leur égalité (d'après 3).

Proposition 3.4 *Si $\varphi\colon \mathcal{B} \to \mathcal{A}$ est un morphisme localement surjectif, alors $|\mathcal{A}| = |\mathcal{B}|$.*

En utilisant, en plus, la surjectivité "classique" (globale), on obtient un type particulier de morphisme :

Définition 3.20 Si $\varphi\colon R \to Q$ est surjective (c'est à dire si $\varphi(R) = Q$), on dira que le morphisme d'automates $\varphi\colon \mathcal{B} \to \mathcal{A}$ est *globalement surjectif*.
Un automate $\mathcal{A}$ est un *quotient* de $\mathcal{B}$ s'il existe un morphisme d'automates $\varphi\colon \mathcal{B} \to \mathcal{A}$ à la fois localement surjectif et globalement surjectif.

Cette nouvelle notion amène à l'idée d'automate quotient minimum : un automate $\mathcal{A}$ est le quotient minimum d'un automate $\mathcal{B}$ s'il est un quotient de tous les automates qui sont eux-mêmes quotients de $\mathcal{B}$. Le quotient minimum d'un automate est donc canoniquement associé à cet automate.

REMARQUE 3.7 Par la suite, on pourra associer canoniquement un automate, dit minimal, à un langage donné grâce au principe d'automate quotient mais cela va nécessiter une propriété de non-ambiguïté plus forte sur les automates, le déterminisme, qui n'aura de sens que pour des automates sur le monoïde libre.

Définition 3.21 Un morphisme d'automate est un *revêtement* (*resp.* un *co-revêtement*) s'il est à la fois localement bijectif (*resp.* co-bijectif) et globalement surjectif.

L'idée derrière ce type de morphismes est qu'il consiste en une "fusion" d'états dont le comportement est "similaire" si l'on regarde leurs transitions (sortantes pour un revêtement et entrantes pour un co-revêtement). Cela dit cette notion est déjà présente dans le principe d'automate quotient à ceci prêt que dans un morphisme qui est à la fois localement et globalement surjectif il

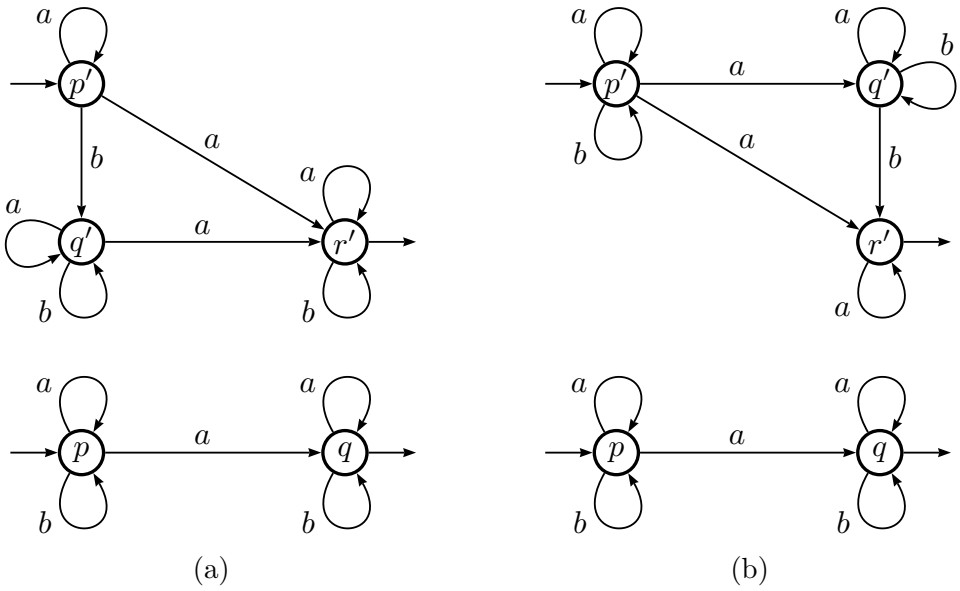


FIGURE 3.6 – Un revêtement et un co-revêtement de l'automate $\mathcal{A}_1$

n'y a pas (forcément) de correspondance biunivoque entre les calculs réussis.

Proposition 3.5 *Si $\varphi\colon \mathcal{B} \to \mathcal{A}$ est un revêtement (resp. un co-revêtement), les calculs réussis de $\mathcal{A}$ sont en bijection avec les calculs réussis de $\mathcal{B}$.*

On dira, par abus de langage, que $\mathcal{B}$ est un revêtement (*resp.* un co-revêtement) de $\mathcal{A}$ s'il existe un morphisme de $\mathcal{B}$ dans $\mathcal{A}$ qui a cette propriété. On peut également noter que, par définition, si $\mathcal{B}$ est un revêtement (*resp.* un co-revêtement) de $\mathcal{A}$ alors $\mathcal{A}$ est un quotient (*resp.* un co-quotient) de $\mathcal{B}$.

EXEMPLE 3.7 Le morphisme φ de la figure 3.6a défini par $\varphi(p') = p$, $\varphi(q') = p$ et $\varphi(r') = r$ est un revêtement. Celui de la figure 3.6b défini par $\varphi(p') = p$, $\varphi(q') = q$ et $\varphi(r') = q$ est un co-revêtement.

3.3 Automates classiques

En théorie des automates, on utilise classiquement des automates sur le monoïde libre A^*. Il s'agit donc d'automates dont les étiquettes des transitions sont des langages de A^* (autrement dit dans $\mathfrak{P}(A^*)$). Mais il ne s'agit pas exactement des automates les plus couramment utilisés, il s'agit d'une généralisation (d'où leur nom d'automates généralisés) des automates dit "classiques" dans lesquels les étiquettes des transitions correspondent à des lettres (les générateurs de A^*). C'est cette restriction des automates sur A^*

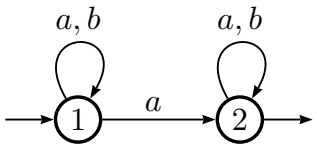


FIGURE 3.7 – L'automate $\mathcal{A}_1$ en version "classique".

que nous allons manipuler.

Définition 3.22 Un automate sur A^* est un tuple $\langle Q, A, E, I, T\rangle$ tel que Q est l'ensemble fini des états, A est l'alphabet, $E \subseteq Q\times A\times Q$ est l'ensemble des transitions, $I \subseteq Q$ est l'ensemble des états initiaux et $T \subseteq Q$ est l'ensemble des états finals.

Les définitions données dans le cas des automates génériques sont toujours valables mais on leurs préfère les définitions plus intuitives suivantes.

Définition 3.23 Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate sur un alphabet A. Un calcul dans $\mathcal{A}$ est une suite de transitions telles que l'état d'arrivée de l'une est égal à l'état d'origine de la suivante, c'est à dire telles que les arcs correspondant à ces transitions forment un chemin dans le graphe sous-jacent. Comme précédemment, l'étiquette d'un calcul correspond au produit (ici la concaténation) des étiquettes de ses transitions et un calcul est dit réussi si son état de départ est initial et si son état d'arrivée est final. Un mot de A^* est accepté par $\mathcal{A}$ s'il est l'étiquette d'au moins un calcul réussi dans $\mathcal{A}$.

REMARQUE 3.8 Comme le montre la figure 3.7 qui reprend l'automate $\mathcal{A}_1$ de la figure 3.4a, les transitions ne sont plus des parties de A^* mais uniquement des éléments de son ensemble générateur. Cela dit, on aura souvent tendance à "regrouper" des transitions entre deux états en une seule transition (a priori) en séparant les étiquettes des transitions par des virgules.

Définition 3.24 Le *langage reconnu* par un automate $\mathcal{A} = \langle Q, A, E, I, T\rangle$, que l'on notera $|\mathcal{A}|$, est l'ensemble des mots qui sont acceptés par $\mathcal{A}$, autrement dit

$$|\mathcal{A}| = \{u \in A^* \mid \exists i \in I, \exists t \in T, i \xrightarrow{u} t\}$$

Deux automates sont équivalents s'ils reconnaissent le même langage.

On dispose de différentes opérations sur les automates finis ainsi définis comme, par exemple, le "complémentaire" d'un automate (qui reconnaît le langage complémentaire), l'union de deux automates ou encore leur produit.

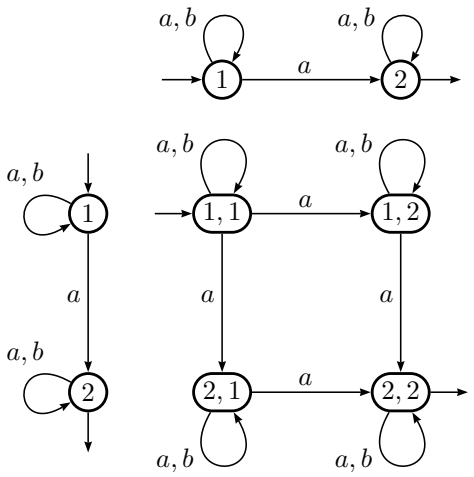


FIGURE 3.8 – Le carré de l'automate $\mathcal{A}_1$.

Proposition 3.6 *Soit $\mathcal{A}$ un automate sur A^*. Il existe un automate fini qui reconnait $\overline{|\mathcal{A}|}$.*

A partir d'un automate $\mathcal{A}$, il suffit de le déterminiser [8], de le rendre complet et ensuite de transformer tous ses états finals en non-finals et inversement.

Proposition 3.7 *Soient $\mathcal{A}$ et $\mathcal{B}$ deux automates finis sur A^*. Il existe un automate fini qui reconnait $|\mathcal{A}| \cup |\mathcal{B}|$.*

En effet, l'union de deux automates $\mathcal{A} = \langle Q, A, E, I, T\rangle$ et $\mathcal{B} = \langle R, A, F, J, U\rangle$ définie par $\mathcal{A} \cup \mathcal{B} = \langle Q \cup R, A, E \cup F, I \cup J, T \cup U\rangle$ reconnait bien l'union du langage de $\mathcal{A}$ et du langage $\mathcal{B}$.

Proposition 3.8 *Soient $\mathcal{A}$ et $\mathcal{B}$ deux automates finis sur A^*. Il existe un automate fini qui reconnait $|\mathcal{A}| \cap |\mathcal{B}|$.*

Ce que l'on appelle le produit cartésien, ou tout simplement produit, de deux automates $\mathcal{A} = \langle Q, A, E, I, T\rangle$ et $\mathcal{B} = \langle R, A, F, J, U\rangle$ est l'automate $\mathcal{A} \times \mathcal{B} = \langle Q \times R, A, E', I \times J, T \times R\rangle$ avec $E' = \{((p, p'), a, (q, q') \mid (p, a, q) \in E$ et $(p', a, q') \in F\}$. Avec une telle définition, un chemin réussi dans $\mathcal{A} \cap \mathcal{B}$ est un chemin réussi à la fois dans $\mathcal{A}$ et dans $\mathcal{B}$.

EXEMPLE 3.8 La figure 3.8 illustre le produit de l'automate $\mathcal{A}_1$ par lui-même qu'on appelle carré de l'automate.

8. nous allons voir cette opération un peu plus loin

REMARQUE 3.9 On peut également construire le produit en utilisant les opérations de complémentation et d'union définie précédemment en se basant sur l'observation suivante : $\mathcal{A} \cap \mathcal{B} = \overline{\overline{|\mathcal{A}|} \cup \overline{|\mathcal{B}|}}$.
Soit $\mathcal{A}$ un automate et soit $\mathcal{A} \times \mathcal{A}$ son carré. Si la diagonale de $\mathcal{A} \times \mathcal{A}$ constitue sa partie émondée alors $\mathcal{A}$ est non ambigu.

Comme on l'a vu, il est possible de représenter un automate de différentes façons mais il s'avère que dans le cas des automates classiques, la définition la plus pratique (et aussi la plus courante) est celle sous forme d'ensembles. Cela dit, il est également pratique de pouvoir étudier les automates sous un angle plus fonctionnel. De ce fait, l'ensemble des transitions $E \subseteq Q \times A \times Q$ peut être vue comme une application δ de $Q \times A$ dans $\mathfrak{P}(Q)$, appelée fonction de transition :

$$\forall (p, a) \in Q \times A, \qquad \delta(p, a) = \{q \in Q \mid (p, a, q) \in E\}$$

Il s'agit en réalité d'une fonction exprimant les successeurs directs. Pour plus de concision, pour p dans Q et a dans A, nous noterons $\delta(p, a) = p \cdot a$.
Cette fonction peut facilement s'étendre non plus à un état unique mais à une partie de Q par addition :

$$\forall P \subseteq Q, \forall a \in A, \qquad P \cdot a = \bigcup_{p \in P} p \cdot a$$

Ce qui en fait une application de $\mathfrak{P}(Q) \times A$ dans $\mathfrak{P}(Q)$ que l'on peut encore étendre par récurrence sur la longueur des mots en une application de $\mathfrak{P}(Q) \times A^*$ dans $\mathfrak{P}(Q)$:

$$\begin{aligned} \forall P \subseteq Q, & \qquad P \cdot 1_{A^*} = P \\ \forall P \subseteq Q, \forall u = u'a \in A^*, & \qquad P \cdot u = (P \cdot u') \cdot a \end{aligned}$$

De façon symétrique, il existe une fonction de $A^* \times \mathfrak{P}(Q)$ dans $\mathfrak{P}(Q)$ qui exprime l'ensemble des ancêtres d'un ensemble d'états. Ces nouvelles définitions sont cohérentes avec celle de calcul. En effet, il existe un calcul de p à q étiqueté par un mot u si et seulement si l'image de (p, u) par δ contient q :

$$\begin{aligned} \forall P \subseteq Q, \forall u \in A^*, & \qquad P \cdot u = \{q \in Q \mid \exists p \in P, p \xrightarrow{u} q\} \\ \forall P \subseteq Q, \forall u \in A^*, & \qquad u \cdot P = \{p \in Q \mid \exists q \in P, p \xrightarrow{u} q\} \end{aligned}$$

On arrive donc logiquement à une nouvelle définition du langage reconnu par un automate $\mathcal{A} = \langle Q, A, E, I, T \rangle$ à partir de sa fonction de transition :

$$|\mathcal{A}| = \{u \in A^* \mid (I \cdot u) \cap T \neq \emptyset\} = \{u \in A^* \mid I \cap (u \cdot T) \neq \emptyset\}$$

(a) non déterminisme (b) non co-déterminisme

FIGURE 3.9

L'utilisation du monoïde libre et le fait de ne considérer que des lettres uniques pour les étiquettes des transitions ajoute en simplicité et va permettre de définir des nouveaux concepts qui n'auraient pas de sens autrement tout en ne diminuant pas la puissance des automates (les ensembles reconnus sont les mêmes).

Définition 3.25 Un automate $\mathcal{A} = \langle Q, A, E, I, T \rangle$ est *déterministe* (*resp. co-déterministe*) s'il n'a qu'un seul état initial (*resp.* final) et si, pour chaque état p de Q et pour chaque lettre a de A, il existe au plus une transition sortante de (*resp.* arrivant en) p étiquetée par a.
Un automate est dit *complet* (*resp. co-complet*) si, pour chaque état p de Q et pour chaque lettre a de A, il existe au moins une transition sortante (*resp.* entrante) de p étiquetée par a.

La notion de déterminisme est extrêmement importante car elle fournit un modèle réaliste et plausible des machines à calculer : à partir d'un état donné de notre machine et pour une action donnée (représentée ici par les lettres) on arrive dans un nouvel état (unique). Ces nouvelles propriétés de déterminisme et de co-déterminisme sont plus fortes que la notion de base de non-ambiguïté. Les figures 3.9a et 3.9b montrent respectivement les cas de non déterminisme et de non co-déterminisme.
Ces notions s'expriment également d'un point de vue fonctionnel. De ce fait, si un automate $\mathcal{A} = \langle Q, A, \delta, I, T \rangle$ est déterministe alors sa fonction de transition est une fonction partielle de $Q \times A$ dans Q, c'est à dire une application δ de $Q \times A$ dans $\mathfrak{P}(Q)$ telle que les images par δ sont soit vides soit des singletons. Si un automate est co-déterministe (*resp.* complet), alors sa fonction de transition est injective (*resp.* surjective).

Théorème 3.9 *Tout automate est équivalent à un automate déterministe (complet).*

En effet, il est possible de construire, à partir de n'importe quel automate, un automate déterministe équivalent. Une opération qui consiste à passer d'un automate non-déterministe à un automate déterministe s'appelle une

déterminisation (elle n'est pas unique). Soit $\mathcal{A} = \langle Q, A, \delta, I, T\rangle$ un automate non-déterministe (a priori) et soit $\mathcal{B} = \langle \mathfrak{P}(Q), A, \delta, \{I\}, U\rangle$ l'automate tel que :

$$\forall P \in \mathfrak{P}(Q), \forall a \in A, \quad P \cdot a = \bigcup_{p \in P} p \cdot a$$

et

$$U = \{S \subseteq Q \mid S \cap T \neq \emptyset$$

De part sa définition, $\mathcal{B}$ est clairement déterministe. D'autre part, $\mathcal{B}$ est équivalent à $\mathcal{A}$ car on a la suite d'équivalences suivantes :

$$\begin{aligned} \forall u \in A^*, f \in |\mathcal{A}| &\iff \exists i \in I, (i \cdot u) \cap T \neq \emptyset \\ &\iff (I \cdot u) \cap T \neq \emptyset \\ &\iff I \cdot u \in U \\ &\iff u \in |\mathcal{B}| \end{aligned}$$

Cela dit, avec une telle définition, il n'est pas assuré que tous les états de l'automate ainsi obtenu soient accessibles. Pour éviter cela, il est possible de construire directement un automate déterministe et accessible équivalent par le biais d'une construction appelée déterminisation "par la méthode des sous-ensembles".

Définition 3.26 Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate. Le déterminisé de $\mathcal{A}$ est l'automate $\mathcal{A}_{\text{det}} = \langle Q', A, E', J, U\rangle$ tel que :

$$\begin{aligned} Q' &= \{I \cdot u \mid u \in A^*\} \\ J &= \{I\} \\ U &= \{S \subseteq Q' \mid S \cap T \neq \emptyset\} \\ E' &= \{(X, a, Y) \in Q' \times A \times Q' \mid X \cdot a = Y\} \end{aligned}$$

On constate facilement qu'un mot donné ne peut appartenir au passé de deux états à la fois, autrement dit que les passés des états d'un automate déterministe sont disjoints deux à deux. Il existe la version duale pour le co-déterminisme de cette construction :

Définition 3.27 Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate. Le co-déterminisé de $\mathcal{A}$ est l'automate $\mathcal{A}_{\text{co-det}} = \langle Q', A, E', J, U\rangle$ tel que :

$$\begin{aligned} Q' &= \{u \cdot T \mid u \in A^*\} \\ J &= \{X \in Q' \mid X \cap I \subseteq \emptyset\} \\ U &= \{T\} \\ E' &= \{(X, a, Y) \in Q' \times A \times Q' \mid X = a \cdot Y\} \end{aligned}$$

FIGURE 3.10 – Le déterminisé et le co-déterminisé de $\mathcal{A}_1$

En effet, en partant du seul état obligatoirement accessible (*resp.* co-accessible), l'unique état initial $\{I\}$ (*resp.* l'unique état final $\{T\}$), et en calculant les états accessibles (*resp.* co-accessibles) de proche en proche (autrement dit, on calcul les états qui sont des images par δ d'états déjà calculés), on obtient un automate qui possède les propriétés attendues.

EXEMPLE 3.9 La figure 3.10 montre le déterminisé ainsi que le co-déterminisé de l'automate $\mathcal{A}_1$.

REMARQUE 3.10 On voit très clairement le lien entre les passés et futurs dans le déterminisé $\mathcal{A}_{\text{det}}$ et ceux dans l'automate de départ $\mathcal{A}$:

$$\begin{aligned} \mathsf{Past}_{\mathcal{A}_{\text{det}}}(X) &\subseteq \bigcap_{p\in X} \mathsf{Past}_{\mathcal{A}}(p) \\ \mathsf{Fut}_{\mathcal{A}_{\text{det}}}(X) &= \bigcup_{p\in X} \mathsf{Fut}_{\mathcal{A}}(p) \end{aligned}$$

On peut faire la même observation dans le cas du co-déterminisé $\mathcal{A}_{\text{co-det}}$:

$$\begin{aligned} \mathsf{Past}_{\mathcal{A}_{\text{co-det}}}(X) &= \bigcup_{p\in X} \mathsf{Past}_{\mathcal{A}}(p) \\ \mathsf{Fut}_{\mathcal{A}_{\text{co-det}}}(X) &\subseteq \bigcap_{p\in X} \mathsf{Fut}_{\mathcal{A}}(p) \end{aligned}$$

Comme nous l'avons déjà évoqué, nous voulons associer à un langage L donné un automate qui lui est canoniquement associé. Dans cette optique, nous avons présenté le concept d'automate quotient minimal mais qui était canoniquement associé à l'automate de départ et non pas au langage. Pour arriver au résultat escompté, il faut appliquer cela mais en partant d'un automate déterministe. De cette façon, on obtient le plus petit [9] automate déterministe qui reconnait le langage L et qu'on appelle l'automate minimal de L.

9. en nombre d'états

Le principe qui se cache sous cette construction repose en fait sur les quotients du langage qui sont très fortement liés au futur des états d'un automate déterministe.

Proposition 3.10 *Soit $\mathcal{A}$ un automate déterministe qui reconnait un langage L. Alors pour tout état p de cet automate et pour tout mot u dans $\mathsf{Past}_{\mathcal{A}}(p)$, on a $\mathsf{Fut}_{\mathcal{A}}(p) = u^{-1}L$.*

De cette façon, on comprend bien que dans un automate déterministe quelconque, deux états distincts peuvent avoir le même futur et donc correspondre au même quotient. Dans l'automate minimal tel qu'on va le redéfinir maintenant, les états seront en bijection avec les quotients du langage.

Définition 3.28 L'*automate minimal* d'un langage L de A^*, également appelé automate des quotients et noté $\mathcal{A}_L$, est l'automate défini par le tuple $\langle Q, A, E, I, T\rangle$ tel que :

$$\begin{aligned} Q &= \{u^{-1}L \mid u \in A^*\} \\ I &= \{L\} \\ T &= \{p \in Q \mid 1_{A^*} \in p\} \\ E &= \{(p, a, q) \mid a^{-1}p = q\} \end{aligned}$$

L'automate minimal $\mathcal{A}_L$ ainsi défini est clairement déterministe (par construction). De même, toujours par construction, le futur de tout état $u^{-1}L$ de Q est exactement égal au quotient de L par u. De ce fait, comme l'état initial de $\mathcal{A}_L$ est L, on obtient la proposition suivante :

Proposition 3.11 *Pour tout langage L de A^*, l'automate $\mathcal{A}_L$ reconnait le langage L.*

REMARQUE 3.11 L'automate ainsi défini est complet mais pas émondé car il contient l'état puits $\emptyset$ qui n'est pas co-accessible.

Proposition 3.12 *Soit $\mathcal{A}$ un automate déterministe (complet) qui reconnait le langage L. L'automate minimal $\mathcal{A}_L$ de L est un quotient de $\mathcal{A}$.*

Il existe beaucoup de méthodes différentes pour calculer l'automate minimal d'un langage mais elles ne seront pas présentées dans ce rapport.

On va maintenant revenir à une version plus algébrique des automates pour faire le lien entre la reconnaissance par morphisme et par automate. La propriété "libre" du monoïde A^* et surtout les générateurs de ce dernier permettent de définir une nouvelle représentation des automates sur A^*, basée

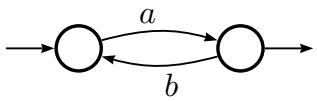


FIGURE 3.11 – Un automate reconnaissant le langage $(ab)^*a$.

sur les fonctions définies par les lettres de A.

Définition 3.29 Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate sur A^*. On appelle *représentation linéaire* de $\mathcal{A}$ le triplet (λ, μ, γ) tel que :

- λ est un vecteur (colonne) de $\mathbb{B}^Q$ tel que, quel que soit p de Q,

$$\lambda_p = 1_{\mathbb{B}} \Longleftrightarrow p \in I$$

- γ est un vecteur (ligne) de $\mathbb{B}^Q$ tel que, quel que soit p de Q,

$$\gamma_p = 1_{\mathbb{B}} \Longleftrightarrow p \in T$$

- μ est une application de A dans $\mathbb{B}^{Q\times Q}$ tel que, quels que soient p et q de Q et quel que soit a de A,

$$\mu a_{p,q} = 1_{\mathbb{B}} \Longleftrightarrow (p, a, q) \in E$$

Le monoïde fini engendré par $\mu(A)$ est appelé *monoïde de transition* de $\mathcal{A}$ et il reconnait le même langage.

En effet, l'application μ s'étend en un morphisme μ de A^* dans $\mathbb{B}^{Q\times Q}$ telle que, pour tout mot u accepté par $\mathcal{A}$, il existe i dans I et t dans T tels que $\mu(u)_{i,t} = 1_{\mathbb{B}}$.

EXEMPLE 3.10 La représentation linéaire de l'automate $\mathcal{A}_1$ est le triplet (λ, μ, γ) tel que :

$$\lambda = \begin{bmatrix}1\\0\end{bmatrix}, \quad \mu(a) = \begin{bmatrix}1 & 1\\0 & 1\end{bmatrix} = \alpha, \quad \mu(b) = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \beta, \quad \gamma = \begin{bmatrix}0 & 1\end{bmatrix}$$

Le monoïde de transition de $\mathcal{A}_1$ est donc le monoïde défini sur $\{\alpha, \beta\}$ avec $\alpha = \mu(a)$ et $\beta = \mu(b)$ par $\alpha = \alpha^2 = \alpha\beta = \beta\alpha$ et $\beta = \beta^2$. Le langage $|\mathcal{A}_1|$ est bien reconnu par le morphisme $\mu\colon A^* \to M$ car il correspond exactement à l'image inverse de α par μ.

EXEMPLE 3.11 Le monoïde de transition de l'automate de la figure 3.11 est le monoïde (fini) M de matrices d'entiers défini par l'ensemble générateur $\alpha = \varphi(a) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ et $\beta = \varphi(b) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$. Il contient 5 éléments : α et β ainsi que les éléments $\alpha\beta = \varphi(ab) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $\beta\alpha = \varphi(ba) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ et $0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. Les relations entre ces éléments sont les suivantes : $\alpha\alpha = \beta\beta = 0$, $\alpha\beta\alpha = \alpha$ et $\beta\alpha\beta = \beta$. Ainsi on obtient les images inverses suivantes $\varphi^{-1}(\alpha) = (ab)^*a$, $\varphi^{-1}(\beta) = (ba)^*b$, $\varphi^{-1}(\alpha\beta) = (ab)^+$, $\varphi^{-1}(\beta\alpha) = (ba)^+$ et $\varphi^{-1}(0) = A^*aaA^* \cup A^*bbA^*$. Comme on le voit, le langage $(ab)^*a$ reconnu par l'automate est bien l'image inverse d'une partie de M (ici $\{\alpha\}$).

Avec cette nouvelle notion de monoïde de transition, on est amené à se pencher sur le monoïde de transition de l'automate minimal du langage.

Définition 3.30 Soit $\mathcal{A}_L$ l'automate minimal d'un langage L. Le monoïde de transition de $\mathcal{A}_L$, appelé *monoïde syntaxique* de L, est un quotient de tout monoïde reconnaissant le langage L.

REMARQUE 3.12 On appellera morphisme syntaxique de L le morphisme canonique de A^* dans le monoïde syntaxique.

Le schéma de la figure 3.12 résume tout cela. Il contient les éléments suivants : M un monoïde reconnaissant un langage L, $\mathcal{A}$ l'automate déterministe dont le monoïde de transition est M, M_L et $\mathcal{A}_L$ sont respectivement le monoïde syntaxique et l'automate minimal de L, ϕ et φ sont tous deux des morphismes (*resp.* de monoïde et d'automates) surjectifs.

La définition 29 entraine que l'ensemble des langages reconnaissables par automate est inclus dans l'ensemble des langages reconnaissables par morphisme (de monoïdes).
Pour la réciproque, on utilise une construction qui vient de la théorie des graphes et que l'on va adapter aux automates : le graphe de Cayley. Mais pour cela, il nous faut d'abord définir la notion d'action d'un monoïde sur un ensemble [10].

10. C'est cette même notion qui permet d'étendre le concept de déterminisme aux automates sur un monoïde quelconque.

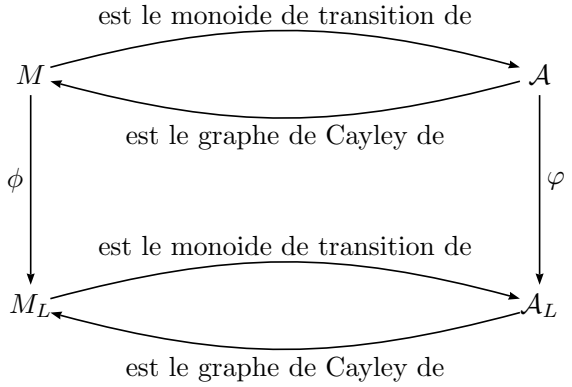


FIGURE 3.12 – Les relations entre automates et monoïdes reconnaissant un langage L.

Définition 3.31 Une action (à droite) d'un monoïde $(M, \cdot)$ sur un ensemble S est une application δ de $S \times M$ dans S telle que :

$$\begin{array}{ll} \forall s \in S & \delta(s, 1_M) = s \\ \forall s \in S, \forall m, n \in M & \delta((\delta(s, m), n) = \delta(s, m \cdot m'). \end{array}$$

On peut désormais construire le graphe de Cayley d'un monoïde reconnaissant un langage L. Soit $(M, .)$ un monoïde et soit $\varphi\colon A^* \to M$ un morphisme de monoïdes qui reconnait L et $\cdot$ une action de A sur M : $m \cdot a = m.\varphi(a)$.
Le graphe de Cayley (M, A) est un graphe orienté étiqueté dans lequel les sommets sont les éléments de M et tel que, pour toute lettre a de A et tout élément m de M, il y a un arc étiqueté par a de m à $m \cdot a$. En regardant ce graphe comme un automate en positionnant correctement l'état initial et les états finals, on obtient un automate (déterministe) qui reconnait L :

$$\mathcal{A} = \langle M, A, E, 1_M, \varphi L \rangle$$

avec

$$E = \{(m, a, n) \mid n = m \cdot a\}$$

Proposition 3.13 *Un langage est reconnu par un morphisme de monoïdes fini si et seulement si il est reconnu par un automate (déterministe) fini.*

Cette proposition couplée à la suivante va nous amener au théorème qui est à la base de la théorie des automates : le théorème de Kleene.

Proposition 3.14 *Un langage est rationnel si et seulement s'il est reconnu par un automate fini.*

Il existe plusieurs méthodes qui permettent de construire un automate fini qui reconnait le langage décrit par une expression rationnelle : la méthode de Thompson [47] , la méthode de Glushkov [24] ou encore la méthode de Brzozowski [6].

A l'inverse, il existe plusieurs façons de calculer le langage reconnu par un automate fini : l'algorithme de McNaughton et Yamada [37], celui de Conway [20] ou encore celui de Brzozowski et McCluskey [5].

Cela nous amène logiquement au théorème de référence :

Théorème 3.15 *[Kleene] Un langage est rationnel si et seulement s'il est reconnaissable (par morphisme).*

Chapitre III

Automate universel d'Oméga-langages

Dans la suite de cette étude, nous manipulons un certains nombre d'extensions des automates élémentaires comme par exemple les automates à multiplicité ou encore les automates bidirectionnels. Une autre extension que nous allons aborder dans ce chapitre consiste à ne plus traiter des séquences finies de lettres mais des séquences infinies. Les mots infinis apparaissent naturellement en informatique pour décrire le comportement de processus (potentiellement) infinis et de modéliser des calculs qui ne sont pas censés se terminer comme par exemple des programmes de contrôle, des systèmes d'exploitation ou encore des protocoles réseaux ; ils constituent une extension naturelle des mots finis. Comme nous l'avons déjà vu, quand il s'agit de langages rationnels, différents types de formalismes peuvent être étudiés : les automates, les monoïdes, les expressions rationnelles, ... Il existe des notions similaires pour les langages rationnels sur les mots infinis, également appelés langages ω-rationnels. En effet, les monoïdes classiques ainsi que les expressions rationnelles ont été étendues aux ω-semigroupes et aux expressions ω-rationnelles respectivement. Bien que ces deux notions soient sans équivoque, pour ce qui est du modèle des automates, il existe plusieurs extensions pour les mots infinis : les automates de Büchi, les automates de Muller, les automates de Rabin, ... Le formalisme le plus intuitif, qui correspond aux automates de Büchi et qui s'avère être le modèle que nous allons utiliser dans ce chapitre, possède certains inconvénients.

Quelques uns des résultats bien connus sur les mots finis se transposent également sur les mots infinis comme par exemple le célèbre théorème de Kleene que nous avons déjà vu sur les mots finis et qui affirme, sur les mots infinis, que les langages ω-rationnels sont exactement les langages qui sont reconnus par les automates de Büchi et, par extension, par les ω-semigroupes.

Cela dit, il y a d'autres propriétés qui ne s'appliquent pas au cas des mots infinis comme par exemple l'existence d'un automate déterministe minimal canonique : les automates de Büchi déterministes sont strictement moins expressifs que les non-déterministes et de ce fait, il n'y a pas de notion d'automate de Büchi minimal. Carton et Michel [16] ont prouvé que les automates prophétiques (qui correspondent, de façon pertinente, à la notion de déterminisme "droite/gauche") sont aussi expressifs que les automates de Büchi, mais il n'existe pas d'automate prophétique minimal unique pour un ω-langage donné. Cela dit, l'automate minimal n'est pas le seul automate canonique associé à un langage rationnel...

Dans ce chapitre, nous étudions une extension d'un autre automate canonique associé à chaque langage rationnel. En 1971, Conway [20] introduit la notion de factorisation d'un langage rationnel (par le biais de sa matrice des facteurs). Ce concept a conduit à la définition d'un nouvel objet appelé l'automate universel d'un langage [35, 45]. Il possède plusieurs propriétés importantes étant donné que tout automate qui reconnaît le même langage L a une image par morphisme qui se trouve être un sous-automate de l'automate universel de L. Par exemple, il peut être utilisé pour calculer un automate fini (non déterministe) de taille minimal [42], ou dans des preuves théoriques sur l'existence d'automates possédant des propriétés particulières (hauteur d'étoile [34], réversibilité [32],*etc.*).

Nous avons étendu le concept de factorisation aux langages ω-rationnels. Avec ces ω-factorisations, nous avons construit l'automate universel d'un langage ω-rationnel et nous avons prouvé que, moyennant une conversion sous forme normale, tout automate de Büchi a une image par morphisme dans cet automate (propriété d'universalité), et que cet automate est minimal pour cette propriété.

Dans la première partie, nous rappelons quelques définitions basiques à propos des langages sur mots infinis et des automates de Büchi. De même, pour les besoins de la construction de l'automate universel, nous introduisons la forme normale d'un automate de Büchi. Finalement, nous rappelons la définition d'ω-semigroupe donnée par [41] et le principe de reconnaissance (par morphisme) d'un langage ω-rationnel par un ω-semigroupe.

Dans la seconde partie, nous définissons les ω-factorisations ainsi que les ω-factorisations pures d'un ω-langage $\mathfrak{L}$ qui sont utilisées pour définir l'automate universel de ce dernier. Ensuite, nous expliquons comment les calculer en utilisant l'ω-semigroupe de transition d'un des automates de Büchi reconnaissant $\mathfrak{L}$. Pour le cas des ω-factorisations, nous décrivons un autre moyen de calcul basé sur l'*automate prophétique*.

La dernière partie est consacrée à la définition de l'automate universel d'un langage ω-rationnel $\mathfrak{L}$. Cela fait intervenir aussi bien les ω-factorisations pures que les standard ainsi que des factorisations dites *positives* sur les mots finis. Enfin, nous exposons les principales propriétés de cet automate : il accepte exactement $\mathfrak{L}$, il a la propriété d'universalité et il est minimal parmi les automates universels de $\mathfrak{L}$.

L'ensemble de ce chapitre a fait l'objet d'un article publié en version courte [9] et en version longue (accepté et en cours de publication au moment de l'écriture de ces lignes).

1 Mots infinis

La plupart des définitions qui vont être évoquées dans cette section sont tirées de [41] sauf mention explicite. Dans ce qui suit, sauf mention explicite, on considère que A est un alphabet sur deux lettres a et b.

1.1 ω-langages

L'appréhension de cette nouvelle facette de la théorie des langages nécessite son propre vocabulaire et ses propres opérations. Un mot infini (ou ω-mot) u sur l'alphabet A est une séquence infinie d'éléments de A que l'on note

$$u = a_0 a_1 a_2 \dots a_n \dots$$

et que l'on peut voir comme une application de $\mathbb{N}$ dans A définie, pour tout n de $\mathbb{N}$, par $u(n) = a_n$. De la même façon que l'on dénote par A^* l'ensemble des mots finis sur A, celui des mots infinis sur A est noté A^ω et on regroupera ces deux ensembles sous la notation $A^\infty = A^* \cup A^\omega$. Avec ces nouvelles notations, on définit également une extension du produit de concaténation sur mots finis au ω-mots, appelé également produit mixte ou simplement produit, d'un mot $u = a_0 a_1 \dots a_n$ et d'un ω-mot $v = b_0 b_1 \dots$, comme l'unique ω-mot $uv = a_0 a_1 \dots a_n b_0 b_1 \dots$. Les définitions de préfixe et de facteur d'un ω-mot u sont les mêmes que pour les mots finis, tandis qu'un suffixe est naturellement un ω-mot w tel que $u = vw$. On appelle ω-langage, tout sous-ensemble de A^ω.

Comme pour les langages de mots finis, on définit également la notion d'ensemble ω-rationnel par le biais de 4 opérations (ω-rationnelles) : les deux premières sont l'union ensembliste et l'étoile de Kleene (qu'on appelle également itération finie) déjà présentent pour les mots finis, les deux autres sont

le produit mixte et l'ω-puissance (appelée également itération infinie) que l'on définit de la façon suivante :

Définition 1.1 Soient X un sous-ensemble de A^* et Y un sous-ensemble de A^ω alors

$$XY = \{xy \mid x \in X \text{ et } y \in Y\}$$

On définit l'opération d'itération infinie d'un ensemble X de A^* comme

$$X^\omega = \{x_0 x_1 \cdots \mid \forall i \in \mathbb{N}, x_i \in X \setminus \{1_{A^*}\}\}$$

et on note

$$X^\infty = X^* \cup X^\omega$$

Ces nouvelles définitions s'accompagnent d'un certain nombre de propriétés remarquables :

Proposition 1.1 *Soient X et Y deux sous-ensembles de A^*. On a les identités suivantes :*

- $(X + Y)^\omega = (X^*Y)^\omega + (X + Y)^* X^\omega$
- $(XY)^\omega = X(YX)^\omega$
- $\forall n > 0 \in \mathbb{N}$, $(X^n)^\omega = (X^+)^\omega = X^\omega$
- $XX^\omega = X^+X^\omega = X^\omega$

Avec ces opérations, il est désormais possible de définir les ensembles ω-rationnels qui nous intéressent.

Définition 1.2 L'ensemble des langages ω-rationnels sur A est le plus petit ensemble qui contient l'ensemble vide, toutes les ω-puissances de langage sur A, qui est stable pour l'union et qui est également stable pour le produit mixte avec des langages rationnels.

Il est clair que, pour tout langage ω-rationnel $\mathfrak{L}$, il existe un ensemble fini de paires de langages rationnels $(X_i, Y_i)_{i \in I}$ tel que

$$\mathcal{L} = \bigcup_{i \in I} X_i Y_i^\omega. \tag{1.1}$$

1.2 Reconnaissance par semigroupe

L'approche de reconnaissance par morphisme de monoïdes finis pour les langages sur A^* s'étant révélée très productive, il y a eu beaucoup de tentatives pour l'étendre aux langages ω-rationnels. Il apparaît que la notion appropriée de reconnaissance des langages ω-rationnels par une structure algébrique finie nécessite une structure plus complexe que celle des monoïdes

classiques dans laquelle il manque l'analogue de l'itération infinie : il s'agit des ω-semigroupes dans laquelle on autorise le produit infini.

Définition 1.3 [Perrin-Pin] Un ω-semigroupe est une paire d'ensembles $S = (S_+, S_\omega)$ dans lesquels on définit les opérations suivantes :
- une opération binaire (associative) définie sur S_+.
- une application de $S_+ \times S_\omega$ dans S_ω appelé *produit mixte* et tel que, pour tout (u, v, w) dans $S_+ \times S_+ \times S_\omega$, $(uv)w = u(vw)$.
- Une application (surjective) de S_+^ω dans S_ω appelé *produit infini* qui est compatible avec l'associativité finie et le produit mixte.

REMARQUE 1.1 Le produit infini n'est en aucun cas défini comme une limite. C'est ce que montre l'exemple suivant.

EXEMPLE 1.1 Soit l'ω-semigroupe $S_1 = (\{0, 1\}, \{a\})$ défini de la façon suivante : $0 \otimes 0 = 0 \otimes 1 = 1 \otimes 0 = 0$, $1 \otimes 1 = 1$ (0 et 1 sont donc des idempotents) et le produit infini de n'importe quel élément est a. La table de cet ω-semigroupe est la suivante :

	0	1 ‖	a
0	0	0 ‖	a
1	0	1 ‖	a

De cette façon, on voit bien que, quel que soit n dans $\mathbb{N}$, $0^n \neq 1^n$ alors que $1^\omega = 0^\omega = a$.

REMARQUE 1.2 Pour des raisons de lisibilité, on aura tendance à représenter les relations d'un ω-semigroupe avec une table comme on le ferait pour un monoïde mais en séparant les deux parties (finie et infinie) par une double barre.

Définition 1.4 Soit $S = (S_+, S_\omega)$ et $T = (T_+, T_\omega)$ deux ω-semigroupes. Un *morphisme d'ω-semigroupe* est une paire $\varphi = (\varphi_+, \varphi_\omega)$ qui consiste en un morphisme de semigroupe $\varphi_+ : S_+ \to T_+$ et une application $\varphi_\omega : S_\omega \to T_\omega$.

Les schémas de la figure 1.1 illustrent respectivement la structure d'ω-semigroupe et le principe de morphisme d'ω-semigroupes. Ces notions permettent de définir la notion de reconnaissance par morphisme (d'ω-semigroupes) définie dans [41, 17].

Définition 1.5 Soit T un ω-semigroupe et soit $\varphi : A^\infty = (A^+, A^\omega) \to T$ un morphisme d'ω-semigroupe surjectif. Un ensemble $\mathfrak{L}$ d'ω-mots sur A est *reconnu* par une paire (T, φ) si $\mathfrak{L} = \varphi_\omega^{-1}(\varphi_\omega(\mathfrak{L}))$.

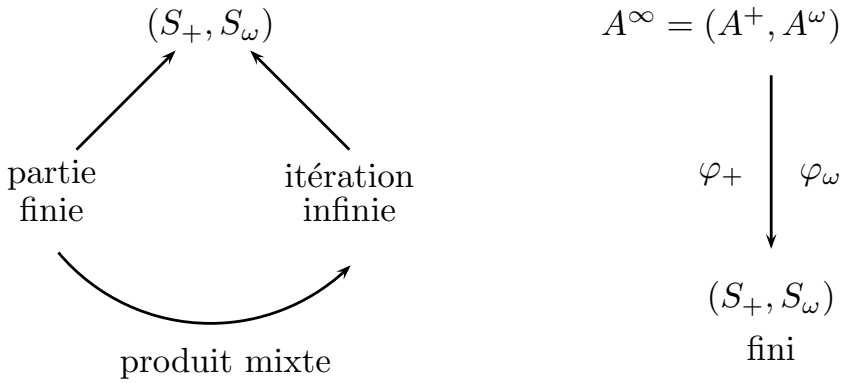


FIGURE 1.1 – ω-semigroupes et morphismes.

Un ensemble d'ω-mots sur A est *reconnaissable* s'il existe un ω-semigroupe fini T et un morphisme d'ω-semigroupe surjectif φ tel que $\mathfrak{L}$ est *reconnu* par (T, φ).

Théorème 1.2 *[41] Un ensemble $\mathfrak{L}$ d'ω-mots sur A est un langage ω-rationnel si et seulement si il est* reconnaissable.

EXEMPLE 1.2 Soit l'ω-semigroupe $S_2 = (\{\alpha, \beta\}, \{\alpha^\omega, \beta^\omega\})$ défini par la table suivante :

1	α	β	α^ω	β^ω
α	α	α	α^ω	β^ω
β	α	β	α^ω	β^ω

Soit le morphisme φ de A^∞ dans S définit par $\varphi(a) = \alpha$ et $\varphi(b) = \beta$. On obtient les images inverses suivantes :

$$\begin{array}{lll}
\varphi^{-1}(\alpha) & = A^*aA^* & \text{(les mots finis qui contiennent au moins un } a\text{)}, \\
\varphi^{-1}(\beta) & = b^+ & \text{(les mots finis qui ne contiennent aucun } a\text{)}, \\
\varphi^{-1}(\beta^\omega) & = A^*b^\omega & \text{(les } \omega\text{-mots qui possèdent un nombre finis de } a\text{)}, \\
\varphi^{-1}(\alpha^\omega) & = A^\omega \setminus A^*b^\omega & \text{(les } \omega\text{-mots qui possèdent une infinité de } a\text{)}
\end{array}$$

Ainsi, n'importe lequel de ces ensembles (ainsi que toute union de ces ensembles) est reconnu par φ et est donc reconnaissable.

1.3 Automates de Büchi

Un automate de Büchi se présente sous la forme d'un tuple $\langle Q, A, E, I, T\rangle$ comme c'est le cas des automates classiques (sur mots finis) avec les mêmes définitions d'ensembles : Q est l'ensemble des états, A l'alphabet, $E \subseteq Q \times A \times Q$ l'ensemble des transitions, $I \subseteq Q$ l'ensemble des états initiaux et $T \subseteq Q$ l'ensemble des états finals. Ils ont donc pratiquement la même allure qu'un automate sur mots finis à l'exception des états finals que l'on préfère représenter avec un double cercle plutôt qu'avec une transition sortante. En effet, un mot infini a un début, ce qui explique que l'on conserve la représentation des états initiaux, mais il n'a pas de fin et donc la nouvelle notation exprime mieux la notion d'infini.
Nous reprenons les définitions déjà rencontrées pour les automates sur mots finis et nous les adaptons aux automates de Büchi et au nouveau formalisme des mots infinis.

Définition 1.6 Soit $\mathfrak{A} = \langle Q, A, E, I, T\rangle$ un automate de Büchi sur A. Un calcul de $\mathfrak{A}$ est une séquence infinie de transition $c = (e_0, e_1, \dots)$ de E telle que l'état d'origine de e_{i+1} est égale à l'état d'arrivée de e_i pour tout i dans $\mathbb{N}$. Un calcul est initial si l'état d'origine de e_0 est initial. Un calcul est final si au moins un des états finals apparaît infiniment souvent dans ce calcul. Un calcul est réussi s'il est à la fois initial et final. Le langage ω-rationnel reconnu par $\mathfrak{A}$, noté $\mathfrak{L}(\mathfrak{A})$, est l'ensemble des ω-mots qui étiquettent un calcul réussi dans $\mathfrak{A}$.

Définition 1.7 Soit $\mathfrak{A}$ un automate de Büchi sur A. On appelle passé d'un état p de $\mathfrak{A}$, que l'on note $\mathsf{Past}_{\mathfrak{A}}(p)$, le langage des mots (finis) qui étiquettent un calcul initial de $\mathfrak{A}$ arrivant en p. De même, on note $\mathsf{Trans}_{\mathfrak{A}}(p, q)$ le langage des mots (finis) qui étiquettent un calcul dans $\mathfrak{A}$ partant de p et arrivant en q. Le futur d'un état p de $\mathfrak{A}$, que l'on note $\mathsf{Fut}_{\mathfrak{A}}(p)$, est le langage des ω-mots qui étiquettent un calcul final partant de p. Le langage reconnu par $\mathfrak{A}$ est l'union des futurs de ses états initiaux.

EXEMPLE 1.3 L'automate de Büchi de la figure 1.2 reconnaît le langage des mots infinis sur l'alphabet $\{a, b, c\}$ qui ne possèdent pas 2 occurences consécutives d'une même lettre.

Comme on peut le voir, la plupart des définitions sont semblables à celle pour les automates sur mots finis et les seules qui changent sont celles qui font intervenir la condition d'acception de l'automate. Cette condition d'acceptation pour un automate de Büchi est la conjonction de conditions sur les préfixes finis d'un mot et de conditions concernant le comportement infini.

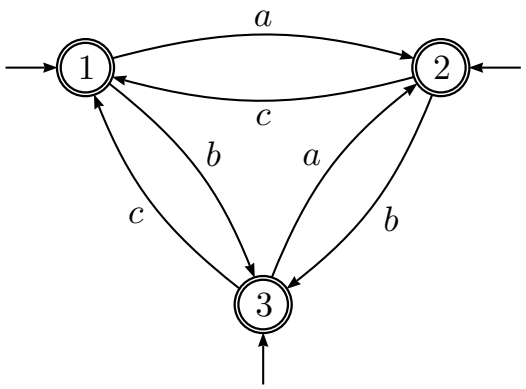


FIGURE 1.2 – Un automate de Büchi.

De ce fait, un automate de Büchi est déterministe dans les mêmes conditions que pour les automates de mots finis tandis qu'il est co-déterministe si ses transitions le sont et si un ω-mot est l'étiquette d'au plus un calcul final. De façon plus générale, un automate de Büchi est non-ambigu si un ω-mot est l'étiquette d'au plus un calcul réussi.

REMARQUE 1.3 A la différence des mots finis et comme nous l'avons déjà mentionné, l'ensemble des ω-langages reconnu par des automates de Büchi déterministes est strictement inclus dans celui des ω-langages reconnus par des automates de Büchi non-déterministes.

Comme pour les mots finis, on souhaite pouvoir construire un ω-semigroupe reconnaissant un certain ω-langage $\mathfrak{L}$ à partir d'un automate de Büchi $\mathfrak{A}$ qui reconnaît $\mathfrak{L}$. De fait, le passage d'un automate à sa représentation linéaire était quasi direct pour les mots finis du fait de la simplicité de la condition d'acceptation qui pouvait être aisément exprimée par des valeurs booléennes : un mot est accepté ou non. Pour les mots infinis il va falloir étendre ce procédé pour permettre d'exprimer le caractère infini d'un calcul en attribuant une "multiplicité" à chaque calcul fini qui exprimera 3 possibilités :

- le calcul n'existe pas,
- le calcul existe mais ne contient pas d'état final récurent,
- le calcul existe et contient au moins un état final récurent.

Pour cela, il est nécessaire d'utiliser le semi-anneau $\mathbb{K} = \{-\infty, 0, 1\}$ avec pour addition, le maximum pour l'ordre $-\infty < 0 < 1$, et pour produit une extension de l'addition booléenne :

	$-\infty$	0	1
$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	$-\infty$	0	1
1	$-\infty$	1	1

Avec cela, comme pour les mots finis, on définit une application μ de A dans l'ensemble des matrices carrées sauf que cette fois ci, ces matrices sont dans $\mathbb{K}^{Q\times Q}$ et plus dans $\mathbb{B}^{Q\times Q}$:

$$\mu(a)_{p,q} = \begin{cases} -\infty & \text{si } (p,a,q) \notin E \\ 0 & \text{si } (p,a,q) \in E,\ p \notin F,\ q \notin F \\ 1 & \text{si } (p,a,q) \in E,\ p \in F \text{ ou } q \in F \end{cases}$$

On peut naturellement étendre μ en un morphisme de A^+ dans $\mathbb{K}^{Q\times Q}$ tel que, pour tout mot u de A^+, $\mu(u)_{p,q}$ vaut : $-\infty$ s'il n'existe pas de calcul de p à q étiqueté par u, 0 s'il existe au moins un calcul de p à q étiqueté par u mais aucun de ces calculs ne contient d'état final, 1 s'il existe au moins un calcul de p à q étiqueté par u et qui contient au moins un état final.
Cela dit, l'objectif étant de construire un ω-semigroupe reconnaissant l'ω-langage, on dispose de la partie finie ($S^+ = \mathbb{K}^{Q\times Q}$) qui est le semigroupe fini engendré par $\mu(A)$) mais il n'y a toujours pas de moyen d'encoder l'existence ou non d'un calcul infini (S_ω) étant donné qu'il n'y a pas d'état d'arrivée. La solution consiste à manipuler pour S_ω, des vecteurs (colonnes) dans $\{-\infty, 1\}$ pour exprimer l'existence ou non d'un calcul infini étiqueté par le produit infini d'un élément de S_+. Pour cela, pour chaque élément s de S_+, on définit un s-calcul infini partant de p, c'est à dire une séquence $(p, p_1, \dots)$ d'éléments de Q telle que $s_{p_i,p_{i+1}} \neq -\infty$, pour tout i de 0 à $n-1$. Ce s-calcul est dit réussi si $s_{p_i,p_{i+1}} = 1$ pour un nombre infini de coefficients. A l'aide de ces nouvelles définitions, pour tout p de Q, on définit chaque élément s^ω de S_ω par :

$${s^\omega}_p = \begin{cases} 1 & \text{s'il existe un } s\text{-calcul réussi partant de } p \\ -\infty & \text{sinon} \end{cases}$$

Ce qui fait de μ un morphisme d'ω-semigroupe de A^∞ dans $S = (S_+, S_\omega)$ qui possède la propriété recherchée.

Proposition 1.3 *Le morphisme μ ainsi défini reconnaît $\mathfrak{L}(\mathfrak{A})$.*

EXEMPLE 1.4 Soit $\mathfrak{L}_1 = A^*b^\omega$ le langage reconnu par l'automate de Büchi de la Figure 1.3. A partir de cet automate, on obtient l'ω-semigroupe S dont les éléments sont :

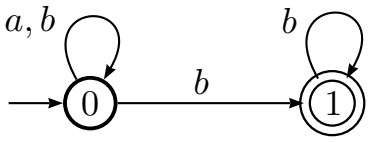


FIGURE 1.3

$$S_+ \longrightarrow \alpha = \begin{bmatrix} 0 & -\infty \\ -\infty & -\infty \end{bmatrix}, \quad \beta = \begin{bmatrix} 0 & 1 \\ -\infty & 1 \end{bmatrix}, \quad \alpha\beta = \begin{bmatrix} 0 & 1 \\ -\infty & -\infty \end{bmatrix}$$

$$S_\omega \longrightarrow \alpha^\omega = \begin{bmatrix} -\infty \\ -\infty \end{bmatrix}, \quad \beta^\omega = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \alpha\beta^\omega = \begin{bmatrix} 1 \\ -\infty \end{bmatrix}$$

La table des relations de ce dernier est la suivante :

1	α	β	$\alpha\beta$	α^ω	β^ω	$\alpha\beta^\omega$
α	α	$\alpha\beta$	$\alpha\beta$	α^ω	$\alpha\beta^\omega$	$\alpha\beta^\omega$
β	α	β	$\alpha\beta$	α^ω	β^ω	$\alpha\beta^\omega$
$\alpha\beta$	α	$\alpha\beta$	$\alpha\beta$	α^ω	$\alpha\beta^\omega$	$\alpha\beta^\omega$

L'image inverse par μ de $\{\beta^\omega, \alpha\beta^\omega\}$ est exactement l'ω-langage $\mathfrak{L}_1$.

A l'inverse, il a été prouvé par [41] (en utilisant les ω-semigroupe ordonnés) que tout ω-langage reconnu par un morphisme d'ω-semigroupe fini est ω-rationnel. Ce qui nous amène à la version sur mots infinis du théorème de Kleene.

Proposition 1.4 *Un langage est ω-rationnel si et seulement s'il est reconnaissable (par un automate de Büchi).*

De plus, à l'instar de ce qui se passe sur les mots finis, il est toujours possible de quotienter les ω-semigroupes que l'on manipule pour obtenir un ω-semigroupe minimal, appelé ω-semigroupe syntaxique. Le morphisme qui permet de passer d'un ω-semigroupe quelconque à l'ω-semigroupe syntaxique est appelé morphisme syntaxique.

EXEMPLE 1.5 Soit l'automate de la figure 1.4 reconnaissant l'ω-langage

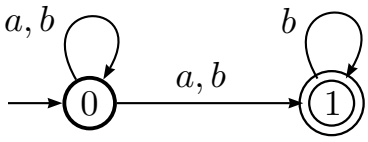


FIGURE 1.4

$\mathfrak{L}_1 = A^*b^\omega$. L'$\omega$-semigroupe que l'on obtient contient les éléments suivants :

$$S_+ \quad \longrightarrow \quad \alpha = \begin{bmatrix} 0 & 1 \\ -\infty & -\infty \end{bmatrix}, \qquad \beta = \begin{bmatrix} 0 & 1 \\ -\infty & 1 \end{bmatrix}$$

$$S_\omega \quad \longrightarrow \quad \alpha^\omega = \begin{bmatrix} -\infty \\ -\infty \end{bmatrix}, \qquad \beta^\omega = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

et dont les relations sont les suivantes :

1	α	β	α^ω	β^ω
α	α	α	α^ω	β^ω
β	α	β	α^ω	β^ω

L'image inverse de β^ω par μ est bien le langage $\mathfrak{L}_1$. L'ω-semigroupe obtenu est un quotient de celui de l'exemple précédent et il est également l'ω-semigroupe syntaxique du langage.

Maintenant que nous avons rappelé les principales définitions et propriétés déjà existantes qui vont être utilisées dans ce chapitre, nous allons en présenter de nouvelles qui vont être nécessaires pour faire le lien entre les automates de Büchi et l'automate universel dont nous rappellerons le concept plus loin.
Pour commencer, nous introduisons une forme normale pour les automates de Büchi qui consiste à séparer l'automate en deux partie : un partie de type *transient* dans laquelle les préfixes finis sont lus, et des composantes *finales* qui lisent la partie infinie.

Définition 1.8 Un état d'un automate de Büchi est dit *transient* s'il n'est accessible par aucun des états finals. Une composante fortement connexe (CFC) est *finale* si elle n'est pas triviale (au moins une transition), elle contient un *unique* état final, et si tout état accessible à partir de cet état final est dans la même CFC. Un automate de Büchi est en forme normale si

i) tout état initial est transient ;
ii) tout état final est dans une CFC finale ;
iii) pour tout état non final q d'une CFC finale S, tous les ancêtres de q sont dans S.

Définition 1.9 Soit $\mathfrak{B} = (Q, A, E, I, F)$ un automate de Büchi. Soit G l'ensemble des états finals de $\mathfrak{B}$ qui appartiennent à une CFC non triviale, et soit $\mathcal{S}$ la fonction qui associe à chaque état f de G sa CFC. La normalisation de $\mathfrak{B}$ est l'automate $\mathcal{B}_{\mathsf{nf}} = (Q', A, E', I', F')$ définit comme suit :

- $Q' = Q \cup \{(p, f) \mid f \in G, p \in \mathcal{S}(f)\}$;
- $I' = I$, $F' = \{(f, f) \mid f \in G\}$;
- $E' = E \cup \{p \xrightarrow{a} (f, f) \mid p \xrightarrow{a} f \in E, f \in G\}$
  $\cup \{(p, f) \xrightarrow{a} (q, f) \mid p \xrightarrow{a} q \in E, f \in G, p, q \in \mathcal{S}(f)\}$.

Proposition 1.5 *Soit $\mathfrak{B}$ un automate de Büchi. La normalisation $\mathcal{B}_{\mathsf{nf}}$ de $\mathfrak{B}$ est un automate de Büchi équivalent en forme normale.*

Démonstration. Soit w un mot accepté par $\mathfrak{B}$ et soit π un calcul acceptant étiqueté par w : $\pi = p_0 \xrightarrow{w_1} p_1 \xrightarrow{w_2} \ldots$. Il existe un état final f tel qu'il y a une infinité de p_i égaux à f (donc f est dans G). Soit k le plus petit $i \geq 1$ tel que $p_i = f$. Pour tout $i \geqslant k$, p_i est dans $\mathcal{S}(f)$. Soit φ la fonction qui associe p_i à p_i si $i < k$ et à (p_i, f) sinon. L'image de π par φ est un calcul acceptant de $\mathcal{B}_{\mathsf{nf}}$.

Réciproquement, la projection qui associe à chaque état (p, f) ou p de $\mathcal{B}_{\mathsf{nf}}$, l'état p de $\mathfrak{B}$, associe également chaque calcul acceptant de $\mathcal{B}_{\mathsf{nf}}$ à un calcul acceptant de $\mathfrak{B}$.

Par définition de $\mathcal{B}_{\mathsf{nf}}$, pour tout état p de Q, tous les prédécesseurs de p dans $\mathcal{B}_{\mathsf{nf}}$ sont aussi dans Q. Les ancêtres de p sont donc des états de Q, et comme aucun état de Q n'est final dans $\mathcal{B}_{\mathsf{nf}}$, l'état p est transient.

Soit (f, f) un état final de $\mathcal{B}_{\mathsf{nf}}$; alors (f, f) est l'unique état final de la copie de la CFC indexée par f. Par définition, toute transition sortante de n'importe lequel des états de cette copie appartient à cette CFC : la CFC est finale. De plus, les transitions arrivant sur cette copie arrivent forcément sur l'état final (f, f). Donc $\mathcal{B}_{\mathsf{nf}}$ est en forme normale. □

EXEMPLE 1.6 La figure 1.5 présente un automate de Büchi ainsi que sa normalisation.

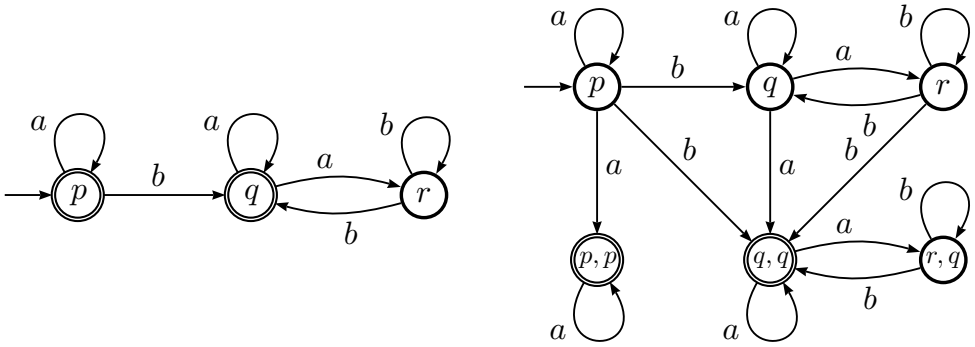


FIGURE 1.5 – Un automate de Büchi et sa normalisation.

2 Automate universel sur mots finis

2.1 Factorisations

Les factorisations de langages ont été introduites par Conway dans [20] et permettaient, au départ, de décider si un langage rationnel appartient à la clôture rationnelle d'une famille finie de langage rationnels. Conway a montré que tout langage rationnel avait un nombre fini de factorisations et que cette propriété caractérisait les langages rationnels. Par la suite, Lombardy et Sakarovitch [35] ont utilisé ce concept de factorisation pour définir un automate canonique d'un langage rationnel.

Définition 2.1 Soit $\mathcal{L}$ un langage sur A. Une k-sous-factorisation de $\mathcal{L}$ est un k-tuple de langages non vides, appelés facteurs, $X = (X_1, X_2, \dots X_k)$ telle que $X_1X_2 \dots X_k$ est un sous-ensemble de $\mathcal{L}$. Une k-sous-factorisation positive est une k-sous-factorisation telle que les langages qui la composent sont dans A^+.

Les k-sous-factorisations d'un langages donné $\mathcal{L}$ sont partiellement ordonnées : $(X_1, X_2 \dots, X_k)$ est inférieure à $(X'_1, X'_2, \dots, X'_k)$ si $X_i \subseteq X'_i$ pour tout i dans $[1; k]$. Dans le cas contraire, elles sont incomparables. Cela implique également qu'il existe des k-sous-factorisations maximales pour cet ordre partiel : les k-factorisations. L'idée derrière cette notion de maximalité est qu'il est impossible de rajouter le moindre élément à n'importe lequel des facteurs d'une k-factorisation sans que l'on ne sorte du langage, *i.e.* sans que le produit de ses facteurs ne contienne un élément qui n'est pas dans le langage.
Pour notre part, nous ne manipulerons que des 2-sous-factorisations, *i.e.* des paires de langages non vides (L, R) où L est appelé facteur gauche et R

facteur droit, que nous appellerons simplement sous-factorisations. De plus, nous noterons $\mathfrak{F}(\mathcal{L})$ l'ensemble des factorisations de $\mathcal{L}$ et nous noterons $\mathfrak{F}_+(\mathcal{L})$ l'ensemble des factorisations positives de $\mathcal{L}$.

EXEMPLE 2.1 Soit $\mathcal{L}_1 = A^*(aa+bb)A^*$ le langage des mots qui contiennent un facteur aa ou un facteur bb et soit $\mathcal{L}_2 = A^*(ab+ba)A^*$ le langage des mots qui contiennent un facteur ab ou un facteur ba. Les factorisations de ces langages sont les suivantes :

$$\mathfrak{F}(\mathcal{L}_1) = \{(A^*, \mathcal{L}_1),\ (\mathcal{L}_1 + A^*a, aA^* + \mathcal{L}_1),\ (\mathcal{L}_1 + A^*b, bA^* + \mathcal{L}_1),\ (\mathcal{L}_1, A^*)\}$$
$$\mathfrak{F}(\mathcal{L}_2) = \{(A^*, \mathcal{L}_2),\ (A^*aA^*, A^*bA^*),\ (A^*bA^*, A^*aA^*),\ (\mathcal{L}_2, A^*)\}$$

Les factorisations positives de ces langages sont les mêmes sauf la première et la dernière de chaque ensemble dans lesquelles il faut modifier le facteur A^* en A^+.

REMARQUE 2.1 Étant donné que nous ne manipulons que des factorisations (des sous-factorisations maximales), chacun des facteurs est maximal par rapport à l'autre ce qui implique qu'une factorisation (L, R) d'un langage $\mathcal{L}$ est entièrement définie par son facteur gauche (ou droit) :

$$L = max\{X \subseteq A^* \mid X.R \subseteq \mathcal{L}\}, \qquad R = max\{X \subseteq A^* \mid L.X \subseteq \mathcal{L}\}$$

Pour notre part, pour des raisons de lisibilité, lorsque nous aurons besoin de "réduire" l'écriture d'une factorisation, nous la désignerons par son facteur gauche.

Proposition 2.1 *Soit $\varphi\colon A^* \to M$ un morphisme qui reconnaît un langage $\mathcal{L}$ de A^*,* i.e. *il existe un sous-ensemble N de M tel que $\mathcal{L} = \varphi^{-1}(N)$. Alors les factorisations de $\mathcal{L}$ sont également reconnues par φ,* i.e. *si (L, R) est un factorisation de $\mathcal{L}$ alors $(\varphi(L), \varphi(R))$ est une factorisation de N.*

REMARQUE 2.2 D'après la proposition 1 et comme le montre l'exemple 36, il est tout à fait possible de définir les factorisations non pas dans A^* mais dans n'importe quel monoïde. Il en va de même pour la définition de l'automate universel qui peut également se faire dans un monoïde quelconque, cela dit, pour notre part, nous ne manipulerons cet outil que dans A^*.

EXEMPLE 2.2 On reprend le langage $\mathcal{L}_1 = A^*(aa+bb)A^*$. Soit $\varphi\colon A^* \to M$ un morphisme de A^* dans un monoïde fini $M = \{\alpha, \beta, 0\}$, dont la table est représentée sur la figure 2.1, défini par $\varphi(a) = \alpha$ et $\varphi(b) = \beta$. Le langage $\mathcal{L}_1$ est bien reconnu par ce morphisme car $\mathcal{L}_1 = \varphi^{-1}(\{0\})$. Dans la figure 2.1, on

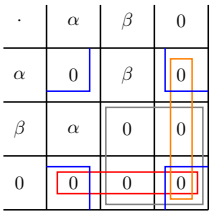


FIGURE 2.1 – La table du monoïde M et les factorisations de $\{0\}$.

a effectué un "regroupement géométrique" des éléments égaux à 0 dans M ce qui permet de repérer les factorisations de ce langage. Le schéma suivant montre la correspondance entre les factorisations de $\mathcal{L}_1$ et leur images par φ dans M :

$$\begin{array}{rcccccc} \mathfrak{F}(\mathcal{L}_1) & = & \{(A^*, \mathcal{L}_1), & (\mathcal{L}_1 + A^*a, aA^* + \mathcal{L}_1), & (\mathcal{L}_1 + A^*b, bA^*\mathcal{L}_1), & (\mathcal{L}_1, A^*)\} \\ & & \downarrow & \downarrow & \downarrow & \downarrow \\ \mathfrak{F}(\{0\}) & = & \{(M, \{0\}), & (\{\alpha, 0\}, \{\alpha, 0\}), & (\{\beta, 0\}, \{\beta, 0\}), & (\{0\}, M)\} \end{array}$$

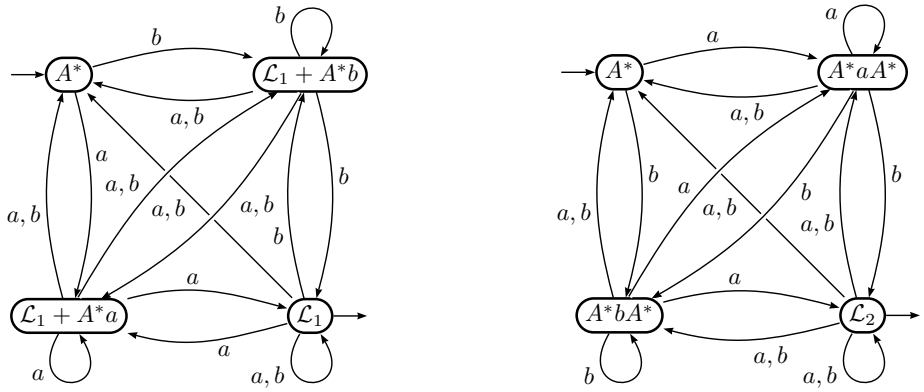
2.2 Définition de l'automate universel

Nous allons maintenant rappeler la définition de l'automate universel qui se base sur les factorisations du langage ainsi que quelques propriétés importantes dont nous omettrons les preuves. Le lecteur intéressé par ces dernières est renvoyé à [35] .

Définition 2.2 Soit $\mathcal{L}$ un langage de A^*. L'automate universel $\mathcal{U}_\mathcal{L}$ de $\mathcal{L}$ est l'automate défini par le tuple $\langle \mathfrak{F}(\mathcal{L}), A, E, I, T \rangle$ tel que :

$$\begin{aligned} I &= \{(L, R) \in \mathfrak{F}(\mathcal{L}) \mid 1_{A^*} \in L)\} \\ T &= \{(L, R) \in \mathfrak{F}(\mathcal{L}) \mid 1_{A^*} \in R)\} \\ E &= \{((L, R), a, (L', R') \in \mathfrak{F}(\mathcal{L}) \times A \times \mathfrak{F}(\mathcal{L}) \mid L.a.R' \subseteq \mathcal{L})\} \end{aligned}$$

Comme on le voit, les factorisations du langage jouent un rôle crucial dans la définition de l'automate universel. Non seulement elles servent d'étiquettes aux états mais en plus, et surtout, elles définissent la fonction de transition de l'automate universel. D'ailleurs ces conditions définissant les transitions peuvent être décrites de différentes façon étant donnée la maximalité des factorisations. De ce fait, un état est initial (*resp.* final) si son facteur droit (*resp.* gauche) est un sous-ensemble de $\mathcal{L}$. De la même façon, on aura une



(a) $\mathcal{U}_{\mathcal{L}_1}$ (b) $\mathcal{U}_{\mathcal{L}_2}$

FIGURE 2.2

transition étiquetée par une lettre a entre deux états (L, R) et (L', R') si $L.a \subseteq L'$ ou, de façon équivalente, si $a.R' \subseteq R$.

Proposition 2.2 *L'automate universel $\mathcal{U}_{\mathcal{L}}$ reconnaît le langage $\mathcal{L}$.*

EXEMPLE 2.3 $\mathcal{U}_{\mathcal{L}_1}$, l'automate universel du langage $\mathcal{L}_1 = A^*(aa + bb)A^*$, et $\mathcal{U}_{\mathcal{L}_2}$, celui du langage $\mathcal{L}_2 = A^*(ab + ba)A^*$, sont décrits sur la figure 2.2. Comme nous l'avons remarqué, une factorisation est entièrement définie par son facteur gauche (ou droit), c'est pourquoi les états ont pour label le facteur gauche de la factorisation uniquement.

Du fait de la définition des transitions qui dépend entièrement des étiquettes des états qui se trouvent être les factorisations du langage, il est logique de constater que le passé et le futur des états seront directement liés à ces étiquettes. De ce fait, on a la propriété remarquable que le passé (*resp.* le futur) d'un état est exactement égal à son facteur gauche (*resp.* droit).

Proposition 2.3 *Soit $\mathcal{U}_{\mathcal{L}}$ l'automate universel d'un langage $\mathcal{L}$. Pour tout état $p = (L, R)$ de $\mathcal{U}_{\mathcal{L}}$, on a les égalités suivantes :*

$$L = \mathsf{Past}_{\mathcal{A}}(p) \text{ et } R = \mathsf{Fut}_{\mathcal{A}}(p)$$

Avec ce lien entre le passé (*resp.* futur) d'un état et le facteur gauche (*resp.* droit) de la factorisation sous-jacente et en tenant compte du fait que ces facteurs sont tous différents de l'ensemble vide, les états sont donc tous

accessibles et co-accessibles.

Corollaire 2.4 *L'automate universel $\mathcal{U}_\mathcal{L}$ d'un langage $\mathcal{L}$ est émondé.*

Les propriétés suivantes sont les plus importantes concernant l'automate universel d'un langage. La première décrit la notion d'universalité pour un automate et la seconde indique que l'automate universel d'un langage est le plut petit automate reconnaissant $\mathcal{L}$ et possédant la propriété d'universalité.

Définition 2.3 Soit $\mathcal{A}$ un automate reconnaissant un langage $\mathcal{L}$. Si pour tout automate $\mathcal{A}'$ reconnaissant $\mathcal{L}$, il existe un morphisme φ de $\mathcal{A}'$ dans $\mathcal{A}$ alors on dit que $\mathcal{A}$ possède la propriété d'universalité.

Proposition 2.5 *Soit $\mathcal{A}$ un automate reconnaissant un langage $\mathcal{L}$ tel qu'il existe un morphisme φ de $\mathcal{U}_\mathcal{L}$ dans $\mathcal{A}$. Alors φ est injectif.*

Cette dernière proposition implique que tout automate qui possède la propriété d'universalité contient $\mathcal{U}_\mathcal{L}$, *i.e.* $\mathcal{U}_\mathcal{L}$ est le plus petit automate reconnaissant $\mathcal{L}$ et remplissant la propriété d'universalité.

L'automate universel sur mots finis possède également une autre propriété intéressante qui tient au fait qu'un langage rationnel possède un nombre fini de factorisations. Comme on le verra par la suite, cette propriété n'est pas vraie sur les mots infinis.

Proposition 2.6 *L'automate universel d'un langage a un nombre fini d'états si et seulement si ce langage est rationnel.*

3 Automate universel sur mots infinis

3.1 ω-factorisations

Nous étendons le concept de factorisation aux ω-semigroupes ; pour cela, nous avons défini deux types de factorisations : les ω-factorisations qui sont des extensions directes des factorisations sur les mots finis et les ω-factorisations pures qui expriment l'itération infinie.

Définition 3.1 Soit $S = (S_+, S_\omega)$ un ω-semigroupe et soit K un sous-ensemble de S_ω. Une ω-sous-factorisation de K est une paire $X = (X_1, X_2)$, avec $X_1 \subseteq S_+^1$ et $X_2 \subseteq S_\omega$ telle que X_1X_2 est un sous-ensemble de K. Une ω-sous-factorisation pure de K est une paire $Y = (Y_1, Y_2)$, avec Y_1 et Y_2 deux sous-ensembles de S_+ et $Y_2 \neq \emptyset$, telle que $Y_1Y_2^\omega$ est un sous-ensemble de K.

On utilise également la même notion de maximalité que pour les factori-

sations : une ω-factorisation (*resp.* une ω-factorisation pure) est une ω-sous-factorisation (*resp.* une ω-sous-factorisation pure) maximale et on note $\mathfrak{F}(K)$ (*resp.* $\mathfrak{F}_p(K)$) l'ensemble des ω-factorisations (*resp.* ω-factorisations pures). Si X est une ω-factorisation de K alors $X_1 = \{x \in S_+^1 \mid xX_2 \subseteq K\}$, et $X_2 = \{y \in S_\omega \mid X_1 y \subseteq K\}$. Si Y est une ω-factorisation pure de K alors $Y_1 = \{x \in S_+ \mid xY_2^\omega \subseteq K\}$ et $Y_2^+ = Y_2$, mais Y_2 n'est pas caractérisé par Y_1.

EXEMPLE 3.1 Considérons le langage $\mathcal{L}_1 = A^*(aa + bb)A^*$ et le langage ω-rationnel $\mathfrak{L}_2 = {\mathcal{L}_1}^\omega$. Les paires $(A^*, \mathfrak{L}_2)$ et $(\emptyset, A^\omega)$ sont les deux ω-factorisations de $\mathfrak{L}_2$. Les paires $(A^+, \mathcal{L}_1 + aA^*a + a)$ et $(A^+, \mathcal{L}_1 + bA^*b + b)$ sont les deux seules ω-factorisations pures de $\mathfrak{L}_2$. [1]

Proposition 3.1 *Soit $\mathfrak{L}$ un langage ω-rationnel. Si $Y = (Y_1, Y_2)$ est dans $\mathfrak{F}_p(\mathfrak{L})$, alors il existe $X = (X_1, X_2)$ dans $\mathfrak{F}(\mathfrak{L})$ telle que $Y_1 = X_1 \cap A^+$ et $Y_2^\omega \subseteq X_2$.*

Démonstration. Comme $Y_1.Y_2^\omega \subseteq \mathfrak{L}$, il existe X dans $\mathfrak{F}(\mathfrak{L})$ telle que $(Y_1, Y_2^\omega) \subseteq (X_1, X_2)$. Donc, $(X_1 \cap A^+)Y_2^\omega \subseteq (X_1 \cap A^+)X_2 \subseteq \mathfrak{L}$, et $(X_1 \cap A^+, Y_2)$ est une ω-factorisation pure. Par maximalité de Y, $X_1 \cap A^+$ est égal à Y_1. □

Proposition 3.2 *Soit $\mathfrak{L}$ un langage ω-rationnel sur A. Soit T un ω-semigroupe fini et soit $\varphi : A^\infty \to T$ tel que $\mathfrak{L}$ est reconnu par (T, φ). Alors, pour tout X dans $\mathfrak{F}(\mathfrak{L})$, l'ensemble $X_1 \cap A^+$ est reconnu par (T_+, φ_+), X_2 est reconnu par (T, φ), et pour tout Y dans $\mathfrak{F}_p(\mathfrak{L})$, Y_1 et Y_2 sont tous deux reconnus par (T_+, φ_+).*

Démonstration. Soit $\mathfrak{L}$ un langage ω-rationnel et soit $\varphi : A^\infty \to T$ un morphisme d'ω-semigroupes tel qu'il reconnaisse $\mathfrak{L}$, où T est un ω-semigroupe fini. Soit φ_* le morphisme de monoïdes de A^* dans T_+^1 qui est l'extension naturelle de φ_+.
Soit $X \in \mathfrak{F}(\mathfrak{L})$, alors $\varphi_*(X_1)\varphi_\omega(X_2) \subseteq \varphi_\omega(\mathfrak{L})$. Alors il existe α dans $\mathfrak{F}(\varphi(\mathfrak{L}))$ telle que $\varphi_*(X_1) \subseteq \alpha_1$ et $\varphi_\omega(X_2) \subseteq \alpha_2$. Comme $\mathfrak{L} = \varphi_\omega^{-1}(\varphi_\omega(\mathfrak{L}))$, alors $\varphi_*^{-1}(\alpha_1)\varphi_\omega^{-1}(\alpha_2) \subseteq \mathfrak{L}$, et ainsi, par maximalité de X, on a $X_1 = \varphi_*^{-1}(\alpha_1)$ et $X_2 = \varphi_\omega^{-1}(\alpha_2)$: les ω-factorisations de $\mathfrak{L}$ sont reconnues par φ.
Soit Y appartenant à $\mathfrak{F}_p(\mathfrak{L})$. La paire $\gamma = (\varphi_+(Y_1), \varphi_+(Y_2))$ est une ω-factorisation pure de $\varphi_\omega(\mathfrak{L})$, et donc il existe β dans $\mathfrak{F}_p(\varphi_\omega(\mathfrak{L}))$, avec $\beta \geqslant \gamma$. En conséquence, $Y_1 \leqslant (\varphi_+^{-1}(\beta_1), \varphi_+^{-1}(\beta_2))$, qui est une ω-factorisation pure de

1. A noter que l'image d'une ω-factorisations pure dans l'ω-semigroupe syntaxique n'est pas nécessairement l'union de *linked pairs* (*cf.* [41]) ; dans $(A^+, \mathcal{L}_1 + aA^*a + a)$, l'image de a n'est pas un idempotent.

$\mathfrak{L}$; par maximalité de Y, on a $Y = (\varphi_+^{-1}(\beta_1), \varphi_+^{-1}(\beta_2))$. Les ω-factorisations pures de $\mathfrak{L}$ sont donc reconnues par φ. □

Corollaire 3.3 *Soit $\mathfrak{L}$ un langage ω-rationnel sur A. Les ensembles $\mathfrak{F}(\mathfrak{L})$ et $\mathfrak{F}_p(\mathfrak{L})$ sont finis et chacun de leurs éléments est une paire de langages (ω-)rationnels. De plus,*

$$\mathfrak{L} = \bigcup_{X \in \mathfrak{F}(\mathfrak{L})} X_1 X_2 = \bigcup_{Y \in \mathfrak{F}_p(\mathfrak{L})} Y_1 Y_2^\omega \tag{3.1}$$

Démonstration. Pour tout mot w dans $\mathfrak{L}$, et pour toute factorisation (u, v) de w, il existe X appartenant à $\mathfrak{F}(\mathfrak{L})$ telle que $u \in X_1$ et $v \in X_2$. Donc w est dans $\bigcup_{X \in \mathfrak{F}(\mathfrak{L})} X_1 X_2$. De plus, pour tout X dans $\mathfrak{F}(\mathfrak{L})$, $X_1 X_2 \subseteq \mathfrak{L}$. En conséquence, $\mathfrak{L} = \bigcup_{X \in \mathfrak{F}(\mathfrak{L})} X_1 X_2$.
Par définition des langages ω-rationnels, il existe des paires $(Z^{(i)})_{i \in [1;n]}$ telles que $\mathfrak{L} = \bigcup_{i=1}^n Z_1^{(i)} Z_2^{(i)\omega} = \bigcup_{i=1}^n Z_1^{(i)} Z_2^{(i)} Z_2^{(i)\omega}$. Pour tout i, $(Z_1^{(i)} Z_2^{(i)}, Z_2^{(i)})$ est une ω-sous-factorisation pure de $\mathfrak{L}$, et donc il existe $Y^{(i)}$ dans $\mathfrak{F}_p(\mathfrak{L})$ telle que $Z_1^{(i)} Z_2^{(i)\omega} \subseteq Y_1^{(i)} Y_2^{(i)\omega} \subseteq \mathfrak{L}$. C'est pourquoi $\bigcup_{Y \in \mathfrak{F}_p(\mathfrak{L})} Y_1 Y_2^\omega = \mathfrak{L}$. □

Corollaire 3.4 *Soit $\mathfrak{B}$ un automate de Büchi et soit $\mathfrak{L}$ le langage ω-rationnel accepté par $\mathfrak{B}$. Les ensembles $\mathfrak{F}(\mathfrak{L})$ et $\mathfrak{F}_p(\mathfrak{L})$ sont effectivement calculables.*

Démonstration. L'ω-semigroupe de transition $S = (S_+, S_\omega)$ de $\mathfrak{B}$ est calculable (*cf.* [41]), ainsi que le morphisme φ tel que $\mathfrak{L}$ est reconnu par (S, φ). Par la Proposition 2, tout calcul d'ω-factorisation peut être fait dans S, qui est fini. □

Cette preuve induit une borne sur le nombre d'ω-factorisations. Comme toute ω-factorisation est caractérisée par l'un de ses facteurs, le nombre d'ω-factorisations est au plus de $\min(2^{|S_+|+1}, 2^{|S_\omega|})$. De façon similaire, le nombre d'ω-factorisations pures est au plus de $2^{|S_+|}$.
A l'inverse du cas sur les mots finis, la finitude de $\mathfrak{F}(\mathfrak{L})$ et de $\mathfrak{F}_p(\mathfrak{L})$ n'implique pas que $\mathfrak{L}$ est ω-rationnel.

EXEMPLE 3.2 Soit $\mathfrak{L}_3 = \{\prod_{i \geqslant 0} a^{f(i)} b \mid f : \mathbb{N} \to \mathbb{N} \text{ et } \forall i, f^{-1}(i) \text{ est fini}\}$; Ce langage n'est pas ω-rationnel et, pourtant, $\mathfrak{F}(\mathfrak{L}_3) = \{(A^*, \mathfrak{L}_3), (\emptyset, A^\omega)\}$ et $\mathfrak{F}_p(\mathfrak{L}_3) = \{(\emptyset, A^+)\}$ sont des ensembles finis.

3.2 Calcul d'ω-factorisations à partir des automates prophétiques

Dans cette partie, nous présentons une méthode alternative pour le calcul des ω-factorisations d'un langage ω-rationnel basée sur un automate prophétique reconnaissant ce langage.

Définition 3.2 [16] Un automate de Büchi $\mathfrak{B}$ sur A est prophétique si le futur de chacun de ses états est non vide et si ces futurs sont disjoints deux à deux.

Théorème 3.5 *[16] Pour tout langage ω-rationnel, il existe un automate prophétique qui le reconnaît.*

La transformation d'un automate de Büchi en automate prophétique est effective, mais compliquée. Néanmoins, quelques langages ω-rationnels naturels sont reconnu par des automates prophétiques très simples.

Pour calculer les ω-factorisations d'un langage à partir d'un automate prophétique le reconnaissant, nous utilisons la construction bien connue par la méthode des sous-ensembles (*subset construction*). L'objectif n'est pas d'obtenir un automate de Büchi équivalent (qui pourrait d'ailleurs ne pas exister...), mais de calculer un ensemble d'états qui seront utilisés dans la Proposition 6 pour caractériser les ω-factorisations. Cette construction est très largement inspirée de la construction dans [31] où le calcul des factorisations d'un langage rationnel se fait soit à l'aide des états du co-déterminisé de l'automate minimal du langage, soit, de façon plus générale, en utilisant les états du déterminisé ainsi que ceux du co-déterminisé d'un automate reconnaissant le langage.

Si $\mathfrak{B} = (Q, A, E, I, F)$ est un automate (de Büchi), pour tout mot u dans A^*, l'ensemble des états accessibles par u à partir d'un sous-ensemble K de Q est $K \cdot u = \{q \mid \exists p \in K \text{ et } u \in \mathsf{Trans}_{\mathcal{A}}(p, q)\}$.

En appliquant la construction par la méthode des sous-ensembles sur $\mathfrak{B}$, on obtient l'ensemble $P = \{K \subseteq Q \mid \exists u \in A^*,\ K = I \cdot u\}$. A noter que $I = I \cdot \varepsilon$ est toujours dans P et que P peut être calculé de façon incrémentale, étant donné que $I \cdot (ua) = (I \cdot u) \cdot a$.

Proposition 3.6 *Soit $\mathfrak{B}$ un automate prophétique avec Q l'ensemble de ses états et $\mathfrak{L} \subseteq A^\omega$ le langage reconnu. Soit P l'ensemble obtenu à l'aide la construction par la méthode des sous-ensembles de $\mathfrak{B}$ et soit $P_\cap$ le plus petit ensemble contenant l'élément Q ainsi que tout élément de P et clos par*

intersection. On pose, pour tout K dans $P_\cap$ et tout X dans $\mathfrak{F}(\mathfrak{L})$,

$$\varphi(K) = \left(\bigcap_{p\in K} \mathsf{Past}_{\mathcal{A}}(p), \bigcup_{p\in K} \mathsf{Fut}_{\mathcal{A}}(p) \right), \qquad \psi(X) = \bigcap_{u\in X_1} I \cdot u. \tag{3.2}$$

Ainsi, φ est une bijection entre $P_\cap$ et $\mathfrak{F}(\mathfrak{L})$ et $\psi = \varphi^{-1}$.

Démonstration. Soit $X \in \mathfrak{F}(\mathfrak{L})$ et soit $H = \psi(X)$, qui est clairement dans $P_\cap$, étant donné que pour tout mot u, l'ensemble $I \cdot u$ est dans P (si $X_1 = \emptyset$, alors $H = Q$).

Nous prouvons maintenant que la factorisation $Y = \varphi(H)$ est égale à X. On a $H = \bigcap_{u\in X_1} I \cdot u = \{p \in Q \mid X_1 \subseteq \mathsf{Past}_{\mathfrak{B}}(p)\}$, et donc $X_1 \subseteq \bigcap_{p\in H} \mathsf{Past}_{\mathfrak{B}}(p) = Y_1$. Pour tout v dans X_2, il existe un unique p dans Q tel que v est dans $\mathsf{Fut}_{\mathfrak{B}}(p)$; on a $X_1 v \subseteq \mathfrak{L}$, ainsi $X_1 \subseteq \mathsf{Past}_{\mathfrak{B}}(p)$ et p est dans H. En conséquence, $X_2 \subseteq \bigcup_{p\in H} \mathsf{Fut}_{\mathfrak{B}}(p)$, et $X = Y$ par maximalité de X.

Réciproquement, soit K dans $P_\cap$, et $Y = \varphi(K)$. On a

$$\begin{aligned} K =& \bigcap \{I \cdot u \mid u \in A^*, K \subseteq I \cdot u\} \\ =& \bigcap \{I \cdot u \mid u \in \bigcap_{p\in K} \mathsf{Past}_{\mathfrak{B}}(p)\} = \bigcap_{u\in Y_1} I \cdot u. \end{aligned} \tag{3.3}$$

Soit X dans $\mathfrak{F}(\mathfrak{L})$ telle que $Y_1 \subseteq X_1$ et $Y_2 \subseteq X_2$. Étant donné que $Y_1 \subseteq X_1$, on a $\psi(X) \subseteq K$. Pour tout p dans K, il existe v dans $\mathsf{Fut}_{\mathfrak{B}}(p) \subseteq Y_2 \subseteq X_2$. Donc $X_1 v \subseteq \mathfrak{L}$ et p est dans $\bigcap_{u\in X_1} I \cdot u = \psi(X)$. Cela implique, $\psi(X) = K$ et $Y = \varphi(\psi(X)) = X$. □

On peut voir cette construction sous un autre angle. Lorsque l'on calcule les ω-factorisations dans le monoïde syntaxique, les éléments de S_+ qui ont les mêmes contextes droits vont toujours se retrouver dans les mêmes ω-factorisations. Les classes d'équivalences de cette relation d'équivalence, que l'on définit concrètement par $\forall x, y \in S_+,\ x \sim y \iff \forall z \in S_\omega,\ x \cdot z = y \cdot z$, correspondent exactement aux états générés par la construction des sous-ensembles d'un automate prophétique.

Le calcul des ω-factorisations pures est plus compliqué et, mis à part dans l'ω-semigroupe syntaxique, nous n'avons pour le moment aucun moyen de les calculer efficacement. En effet, comme le montre l'exemple 38, deux ω-factorisations pures d'un même langage peuvent avoir le même facteur gauche ce qui implique que les ω-factorisations pures ne sont pas caractérisées par leur facteur gauche. De ce fait, il n'est pas possible de les calculer à partir de la construction par la méthode des sous-ensembles comme c'est le cas pour les ω-factorisations.

Pour le moment, le seul moyen dont nous disposons pour les calculer est par le biais du morphisme syntaxique (utilisant l'ω-semigroupe syntaxique). En effet, si (X, Y) est une ω-factorisation du langage, alors (X, T) est une ω-factorisation pure si et seulement si T est l'image inverse d'un sous-semigroupe maximal de S_+ tel que $X.T^\omega$ est inclus dans le langage et qu'il n'existe pas de facteur gauche X' contenant X tel que $X'.T$ est inclus dans le langage. On peut être légèrement plus précis en disant qu'il faut que T soit l'image inverse d'un sous-semigroupe maximal tel que l'image inverse de l'ω-puissance de chacun de ses idempotents soit dans Y. C'est actuellement la meilleure (car la seule ...) méthode que nous ayons pour calculer les ω-factorisations pures d'un langage.

3.3 Définition de l'automate universel sur ω-mots

Dans cette partie, nous étendons la définition d'automate universel d'un langage [35] aux mots infinis. Dans le cas des mots finis, l'automate universel d'un langage L est le plus petit automate qui reconnaît L et dans lequel tout automate équivalent s'envoie par morphisme.
Nous devons tout d'abord décrire la définition de l'automate universel d'un ω-langage $\mathfrak{L}$ et ensuite prouver qu'il est effectivement le plus petit automate de Büchi qui reconnaît $\mathfrak{L}$ et dans lequel tout automate équivalent (en forme normale) s'envoie par morphisme.

La définition de l'automate universel de langage ω-rationnel implique des ω-factorisations, des ω-factorisations pures ainsi que des factorisations positives.

Définition 3.3 Soit $\mathfrak{L}$ un langage ω-rationnel sur A. Pour tout Y dans $\mathfrak{F}_p(\mathfrak{L})$, on pose $\mathcal{Z}_Y = \{Z \in \mathfrak{F}_+(Y_2) \mid Z \neq (Y_2, Y_2)\}$. L'automate universel $\mathcal{U}_\mathfrak{L}$ de $\mathfrak{L}$ est un automate de Büchi défini de la façon suivante.
L'ensemble des états de $\mathcal{U}_\mathfrak{L}$ est l'union finie de $\mathfrak{F}(\mathfrak{L})$, $\mathfrak{F}_p(\mathfrak{L})$ et $\mathcal{Z}_Y$ pour tout $Y \in \mathfrak{F}_p(\mathfrak{L})$.
L'ensemble des états terminaux de $\mathcal{U}_\mathfrak{L}$ est $\mathfrak{F}_p(\mathfrak{L})$.
L'ensemble des états initiaux de $\mathcal{U}_\mathfrak{L}$ est $\{X \in \mathfrak{F}(\mathfrak{L}) \mid \varepsilon \in X_1\}$.
L'ensemble des transitions de $\mathcal{U}_\mathfrak{L}$ est

$$\begin{aligned}
&\{X \xrightarrow{a} X' \mid a \in A,\ X, X' \in \mathfrak{F}(\mathfrak{L}) \text{ et } X_1 a \subseteq X_1'\} \\
\cup&\{X \xrightarrow{a} Y \mid a \in A, X \in \mathfrak{F}(\mathfrak{L}), Y \in \mathfrak{F}_p(\mathfrak{L}),\ \text{et } X_1 a Y_2{}^\omega \subseteq \mathfrak{L}\} \\
\cup&\{K \xrightarrow{a} Y \mid a \in A,\ Y \in \mathfrak{F}_p(\mathfrak{L}), K \in \mathcal{Z}_Y \cup \{Y\}, a \in K_2\} \\
\cup&\{K \xrightarrow{a} Z \mid a \in A,\ \exists Y \in \mathfrak{F}_p(\mathfrak{L}), K \in \mathcal{Z}_Y \cup \{Y\}, Z \in \mathcal{Z}_Y \text{ et } aZ_2 \subseteq K_2\}.
\end{aligned}$$

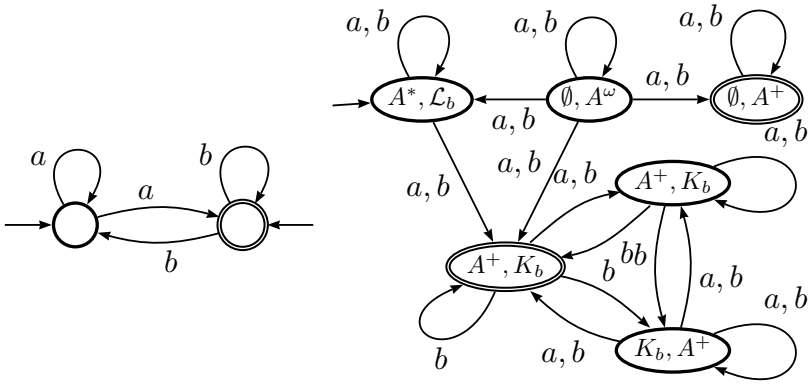


FIGURE 3.1 – Un automate de Büchi prophétique et l'automate universel reconnaissant les mots avec une infinité de 'b'.

EXEMPLE 3.3 Soit $\mathfrak{L}_b$ le langage des mots avec une infinité de 'b', qui est reconnu par l'automate de la Figure 3.1 (gauche). Nous avons $\mathfrak{F}(\mathfrak{L}_b) = \{(A^*, \mathfrak{L}_b), (\emptyset, A^\omega)\}$ et $\mathfrak{F}_p(\mathfrak{L}_b) = \{(\emptyset, A^+), (A^+, K_b)\}$, où $K_b = A^*bA^*$, et à partir des facteurs droits des éléments de $\mathfrak{F}_p(\mathfrak{L}_b)$ nous obtenons les ensembles de factorisations positives suivants : $\mathfrak{F}_+(A^+) = \{(A^+, A^+)\}$ et $\mathfrak{F}_+(K_b) = \{(A^+, K_b), (K_b, A^+)\}$. A noter que $\mathcal{Z}_{(\emptyset,A^+)} = \mathfrak{F}_+(A^+) \setminus \{(A^+, A^+)\} = \emptyset$.
La construction de l'automate universel de $\mathfrak{L}_b$ est montrée Figure 3.1 (droit).

3.4 Propriétés de base de l'automate universel

Les choix concernant la définition de $\mathcal{U}_\mathfrak{L}$ peut sembler arbitraire au premier abord. Cependant, les propositions qui vont suivre, qui semblent découler naturellement de la définition de l'automate universel et qui sont nécessaires à la justesse de cette notion, peuvent ne pas être vérifiées à partir du moment où l'on change ne serait-ce que légèrement la définition de l'automate universel.

Les conditions qui définissent les transitions de l'automate universel peuvent être généralisées pour caractériser les calculs dans cet automate.

Lemme 3.7 *Soit $\mathfrak{L}$ un langage ω-rationnel. Soit $\mathcal{U}_{\mathfrak{L}}$ l'automate universel de $\mathfrak{L}$ et soit $\mathcal{Z}_Y$ définit comme dans la Définition 3. Soit w un mot (fini et différent du mot vide). Soit X et X' dans $\mathfrak{F}(\mathfrak{L})$, Y dans $\mathfrak{F}_p(\mathfrak{L})$, Z dans $\mathcal{Z}_Y$, et K dans $\mathcal{Z}_Y \cup Y$. On a :*

a) $w \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, X') \Leftrightarrow X_1 w \subseteq X'_1$; *b)* $w \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, Y) \Leftrightarrow X_1 w Y_2^{\omega} \subseteq \mathfrak{L}$;
c) $w \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Y) \Leftrightarrow w \in K_2$; *d)* $w \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Z) \Leftrightarrow wZ_2 \subseteq K_2$.

Démonstration. On prouve ces propriétés par récurrence sur la longueur de w.
Si w est une lettre seule, le lemme correspond à la Définition 3.
Sinon, soit a la dernière lettre de w et soit w' tel que $w = w'a$.
a) Si $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, X')$, il existe un état T tel que $w' \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, T)$ et $a \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(T, X')$. Par hypothèse de récurrence, $X_1 w'a \subseteq T_1 a \subseteq X'_1$.
Réciproquement, soit $T \in \mathfrak{F}(\mathfrak{L})$ et telle que $T_1 = \{u \mid ua \in X'_1\}$. Comme $X_1 w'a \subseteq X'_1$, on a $X_1 w' \subseteq T_1$. En conséquence, $w'a \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, T)\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(T, X')$.
b) Si w appartient à $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, Y)$, il existe une factorisation de $w = w_1 a w_2$, et un état X' dans $\mathfrak{F}(\mathfrak{L})$ tels que $w_1 \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, X')$, $a \in \mathsf{Trans}_{\mathcal{A}}(X', Y)$ et $w_2 \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Y)$. Ainsi $X_1 w_1 \subseteq X'_1$, $w_2 Y_2 \subseteq Y_2$ et $X'_1 a Y_2^{\omega} \subseteq \mathfrak{L}$. En conséquence, $X_1 w_1 a w_2 Y_2^{\omega} \subseteq X'_1 a w_2 Y_2^{\omega} \subseteq X'_1 a Y_2^{\omega} \subseteq \mathfrak{L}$.
Réciproquement, si $X_1 w Y_2^{\omega} \in \mathfrak{L}$, avec $w = w'a$, soit $X' \in \mathfrak{F}(\mathfrak{L})$ et $X'_1 = \{u \mid uaY_2^{\omega} \in \mathfrak{L}\}$. Donc X' est dans $\mathfrak{F}(\mathfrak{L})$, $X_1 w' \subseteq X'_1$, et $X'_1 a Y_2^{\omega} \in \mathfrak{L}$. Par conséquent, $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X, X')\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(X', Y)$.
c) Si $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Y)$, il existe un état T tel que $w' \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, T)$ et $a \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(T, Y)$. Si $T = Y$, a est dans Y_2, w' est dans K_2, et comme $K_2 Y_2 \subseteq K_2$, $w'a$ est dans K_2 ; sinon, $w'aY_2 \subseteq w'T_2 \subseteq K_2$.
Réciproquement, si $w'a$ est dans K_2, soit $T \in \mathfrak{F}_+(Y_2)$ telle que $T_2 = \{v \mid K_1 w' v \in Y_2\}$. Si $T = (Y_2, Y_2)$, a est dans $T_2 = Y_2$ et $K_1 w' \subseteq T_1 = Y_2$, d'où $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Y)\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Y) \subseteq \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Y)$. Sinon, T est un état de $\mathcal{U}_{\mathfrak{L}}$, a est dans T_2, et $w'T_2 \subseteq K_2$, c'est pourquoi $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, T)\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(T, Y) \subseteq \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Y)$.
d) Si $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, Z)$, il existe un état T tel que $w' \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, T)$ et $a \in \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(T, Z)$. Ainsi, $w'aZ_2 \subseteq w'T_2 \subseteq K_2$.
Réciproquement, soit $T \in \mathfrak{F}_+(Y_2)$ telle que $T_2 = \{v \mid w'v \in K_2\}$. Si $T = (Y_2, Y_2)$, sans perte de généralité, remplaçons la par Y. Comme $w'aZ_2 \subseteq K_2$, par maximalité de T_2, $aZ_2 \subseteq T_2$. Par conséquence, $w'a$ est dans $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(K, T)\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(T, Z)$. □

Le futur et le passé des états de l'automate universel sont très liés aux factorisations qui définissent ces états comme c'est le cas sur les mots finis.

Proposition 3.8 *Soit $\mathfrak{L}$ un langage ω-rationnel et soit $\mathcal{U}_\mathfrak{L}$ l'automate universel de $\mathfrak{L}$.*
1. Pour tout X dans $\mathfrak{F}(\mathfrak{L})$,

$$a)\ \mathsf{Past}_{\mathcal{U}_\mathfrak{L}}(X) = X_1 \qquad b)\ \mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(X) = X_2.$$

2. Pour tout Y dans $\mathfrak{F}_p(\mathfrak{L})$ et pour tout $Z \in \mathcal{Z}_Y$,

$$a)\ \mathsf{Past}_{\mathcal{U}_\mathfrak{L}}(Y) = Y_1 \qquad b)\ \mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(Y,Y) = Y_2 \quad c)\ \mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(Y) = {Y_2}^\omega$$
$$d)\ \mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(Y,Z) = Z_1 \quad e)\ \mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(Z,Y) = Z_2.$$

Démonstration. 2 b) et 2 e) se déduisent directement du Lemme 7 c). 2 c) se déduit de 2 b) et du fait qu'il n'y a aucun autre état final accessible depuis Y.
– 2 d) D'après le Lemme 7 d), $\mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(Y,Z) = \{w \mid wZ_2 \subseteq Y_2\} = Z_1$.
– 1 b) Par définition, $\mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(X) = \cup_{Y\in\mathfrak{F}_p(\mathfrak{L})}\mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(X,Y)\mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(Y)$. D'après le Lemme 7, on a $\mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(X) = \cup_{Y\in\mathfrak{F}_p(\mathfrak{L})}\{w \mid X_1 w {Y_2}^\omega \subseteq \mathfrak{L}\}{Y_2}^\omega = \cup_{Y\in\mathfrak{F}_p(\mathfrak{L})}\{w{Y_2}^\omega \mid X_1 w {Y_2}^\omega \subseteq \mathfrak{L}\} = \{u \in A^\omega \mid X_1 u \subseteq \mathfrak{L}\} = X_2$.
Soit I l'ω-factorisation de $\mathfrak{L}$ telle que $I_2 = \mathfrak{L}$. Le mot vide ε est dans I_1 et donc I est un état initial.
– 1 a) Soit X appartenant à $\mathfrak{F}(\mathfrak{L})$. On a $X_1X_2 \subseteq \mathfrak{L} = I_2$, d'où, par le Lemme 7 a), tout mot de X_1 est dans $\mathsf{Past}_{\mathcal{U}_\mathfrak{L}}(X)$. Réciproquement, si w est dans $\mathsf{Past}_{\mathcal{U}_\mathfrak{L}}(X)$, il existe un état initial X' tel que w est dans $\mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(X',X)$, donc $X'_1 w \subseteq X_1$, et comme ε est dans X'_1, le mot w est dans X_1.
– 2 a) Soit Y dans $\mathfrak{F}_p(\mathfrak{L})$. On a $I_1Y_1{Y_2}^\omega \subseteq I_1\mathfrak{L} = \mathfrak{L}$, d'où, par le Lemme 7 b), tout mot de Y_1 est dans $\mathsf{Past}_{\mathcal{U}_\mathfrak{L}}(Y)$. Réciproquement, si w est dans $\mathsf{Past}_{\mathcal{U}_\mathfrak{L}}(Y)$, il existe un état initial X tel que w est dans $\mathsf{Trans}_{\mathcal{U}_\mathfrak{L}}(X,Y)$, donc $X_1 w {Y_2}^\omega \subseteq \mathfrak{L}$, et comme ε est dans X_1, le mot w est dans $X_1 w \subseteq Y_1$. □

Proposition 3.9 *L'automate universel d'un langage ω-rationnel est un automate de Büchi fini en forme normale.*

Démonstration. D'après la Proposition 2 et le Corollaire 3, si $\mathfrak{L}$ est un langage ω-rationnel, $\mathfrak{F}(\mathfrak{L})$ et $\mathfrak{F}_p(\mathfrak{L})$ sont finis. De plus, le facteur droit de n'importe quel Y appartenant à $\mathfrak{F}_p(\mathfrak{L})$ est un langage rationnel et $\mathfrak{F}(Y_2)$ est fini. Donc $\mathcal{U}_\mathfrak{L}$ est un automate de Büchi fini.
$\mathfrak{F}(\mathfrak{L})$ est l'ensemble des états transitoires de $\mathcal{U}_\mathfrak{L}$: par définition, $\mathfrak{F}(\mathfrak{L})$ ne

contient aucun état final et il n'y a aucune transition arrivant sur un état de $\mathfrak{F}(\mathfrak{L})$ et provenant de l'extérieur de $\mathfrak{F}(\mathfrak{L})$. Tout état initial est dans $\mathfrak{F}(\mathfrak{L})$ et donc est transitoire.
Pour tout Y dans $\mathfrak{F}_p(\mathfrak{L})$, d'après la Proposition 8, tout mot w dans Y_2 étiquette un circuit autour de Y, et donc la CFC de Y n'est pas triviale.
Par définition, pour tout état dans $\{Y\} \cup \mathcal{Z}_Y$ il n'y a pas de transition partant d'un des éléments de cet ensemble et arrivant sur un état en dehors de la CFC. Cette CFC est donc finale et ne contient que ces états et l'unique état final Y. Par définition, toute transition partant de l'extérieur de la CFC et arrivant dedans, arrive forcément sur Y. En conséquence, $\mathcal{U}_\mathfrak{L}$ est en forme normale. □

D'après la Proposition 8, il est évident que :

Proposition 3.10 *L'automate universel de $\mathfrak{L}$ reconnaît $\mathfrak{L}$.*

Démonstration. Soit I l'ensemble des états initiaux de $\mathcal{U}_\mathfrak{L}$, $\mathcal{U}_\mathfrak{L}$ reconnaît $\bigcup_{p\in I} \mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(p)$. Dans $\mathcal{U}_\mathfrak{L}$, un état $X \in \mathfrak{F}(\mathfrak{L})$ est initial si et seulement si $\varepsilon \in X_1$. Donc, comme $X_1X_2 \subseteq \mathfrak{L}$, alors $X_2 \subseteq \mathfrak{L}$. Étant donné que $\mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(X) = X_2$ alors $\mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(X) \subseteq \mathfrak{L}$. Réciproquement, soit $X \in \mathfrak{F}(\mathfrak{L})$ telle que $X_2 = \mathfrak{L}$ et $X_1 = \{u \mid u\mathfrak{L} \subseteq \mathfrak{L}\}$. Ainsi $\varepsilon \in X_1$ et X est un état initial de $\mathcal{U}_\mathfrak{L}$. Donc, comme $\mathsf{Fut}_{\mathcal{U}_\mathfrak{L}}(X) = \mathfrak{L}$, tout ω-mot dans $\mathfrak{L}$ est accepté par $\mathcal{U}_\mathfrak{L}$. □

L'automate universel de $\mathfrak{L}$ est canonique par rapport à $\mathfrak{L}$. Nous allons maintenant prouver qu'il est effectivement universel pour $\mathfrak{L}$, *i.e.* qu'il contient l'image par morphisme de tout automate de Büchi en forme normale qui reconnaît $\mathfrak{L}$.

Proposition 3.11 *(Universalité) Soit $\mathfrak{B}$ un automate de Büchi en forme normale qui reconnaît $\mathfrak{L}$. Il existe un morphisme φ de $\mathfrak{B}$ dans $\mathcal{U}_\mathfrak{L}$. De plus, on peut faire en sorte que ce morphisme envoie les états transitoires de $\mathfrak{B}$ sur les états transitoires de $\mathcal{U}_\mathfrak{L}$.*

Démonstration. D'abord nous définissons l'image par φ de tout état p de $\mathfrak{B}$ dans $\mathfrak{F}(\mathfrak{L}) \cup \mathfrak{F}_p(\mathfrak{L})$.

i) Si p est un état transient, soient $X_2 = \{v \mid \mathsf{Past}_\mathcal{A}(p)v \subseteq \mathfrak{L}\}$ et $X_1 = \{u \mid uX_2 \subseteq \mathfrak{L}\}$. On note $\varphi(p) = (X_1, X_2)$. Si p est initial alors $\varepsilon \in \mathsf{Past}_\mathcal{A}(p)$ et, comme $\mathsf{Past}_\mathcal{A}(p) \subseteq X_1$ par maximalité, $\varepsilon \in X_1$ et donc $\varphi(p)$ est initial.

ii) Si p est un état final. Soient $Y_1 = \{u \mid u\mathsf{Trans}_\mathcal{A}(p,p)^\omega \subseteq \mathfrak{L}\}$ et $Y_2 \in max\{T \mid Y_1T^\omega \subseteq \mathfrak{L} \text{ et } \mathsf{Trans}_\mathcal{A}(p,p) \subseteq T\}$. On pose $\varphi(p) = (Y_1, Y_2)$. Par définition, cet état est final.

iii) Si p appartient à une CFC contenant un état final q distinct de p. Soient $Y = \varphi(q)$, $Z_1 = \{u \mid u\mathsf{Trans}_{\mathcal{A}}(p,q) \subseteq Y_2\}$ et $Z_2 = \{v \mid Z_1 v \subseteq Y_2\}$. Si $(Z_1, Z_2) = (Y_2, Y_2)$, alors $K = Y$, sinon $K = Z$. On pose $\varphi(p) = K$.

Soient p et q deux états transitoires et soient $X = \varphi(p)$ et $X' = \varphi(q)$. Si $p \xrightarrow{a} q$ est une transition de $\mathfrak{B}$, alors $\mathsf{Past}_{\mathcal{A}}(p)a \subseteq \mathsf{Past}_{\mathcal{A}}(q)$. Donc $aX'_2 = \{av \mid \mathsf{Past}_{\mathcal{A}}(q)v \subseteq \mathfrak{L}\} \subseteq \{av \mid \mathsf{Past}_{\mathcal{A}}(p)av \subseteq \mathfrak{L}\} \subseteq X_2$. Ainsi, $X_1 a X'_2 \subseteq X_1 X_2 \subseteq \mathfrak{L}$ et par maximalité des factorisations, $X_1 a \subseteq X'_1$. En conséquence, il y a une transition $X \xrightarrow{a} X'$ dans $\mathcal{U}_{\mathfrak{L}}$.

Soient p un état transitoire, q un état final, et soient $X = \varphi(p)$ et $Y = \varphi(q)$. Si $p \xrightarrow{a} q$ est une transition dans $\mathfrak{B}$, alors $\mathsf{Past}_{\mathcal{A}}(p)a\mathsf{Trans}_{\mathcal{A}}(q,q)^{\omega} \subseteq \mathfrak{L}$. De ce fait, $\mathsf{Past}_{\mathcal{A}}(p)a \subseteq Y_1$ et $\mathsf{Past}_{\mathcal{A}}(p)aY_2^{\omega} \subseteq \mathfrak{L}$; donc $aY_2^{\omega} \subseteq X_2$ et $X_1 a Y_2^{\omega} \subseteq \mathfrak{L}$: il y a une transition $X \xrightarrow{a} Y$ dans $\mathcal{U}_{\mathfrak{L}}$.

Soit $p \xrightarrow{a} p'$ une transition appartenant à une CFC finale de $\mathfrak{B}$ avec un état final q et soit $Y = \varphi(q)$. On pose $Z_1 = \{u \mid u\mathsf{Trans}_{\mathfrak{B}}(p,q) \subseteq Y_2\}$ et $Z'_1 = \{u \mid u\mathsf{Trans}_{\mathfrak{B}}(p',q) \subseteq Y_2\}$. On a $a\mathsf{Trans}_{\mathcal{A}}(p',q) \subseteq \mathsf{Trans}_{\mathcal{A}}(p,q)$. Donc $Z_1 a = \{ua \mid u\mathsf{Trans}_{\mathcal{A}}(p,q) \subseteq Y_2\} \subseteq \{ua \mid ua\mathsf{Trans}_{\mathcal{A}}(p',q) \subseteq Y_2\} \subseteq \{v \mid v\mathsf{Trans}_{\mathcal{A}}(p',q) \subseteq Y_2\} = Z'_1$.

Soit $K = \varphi(p)$; par définition, $K_2 = \{v \mid Z_1 v \subseteq Y_2\}$. Si $\varphi(p') \neq Y$, $\varphi(p') = Z'$, et $aZ'_2 = a\{v \mid Z'_1 v \subseteq Y_2\} \subseteq \{av \mid Z_1 av \subseteq Y_2\} \subseteq K_2$. Ainsi il y a une transition $K \xrightarrow{a} Z'$ dans $\mathcal{U}_{\mathfrak{L}}$. Si $\varphi(p') = Y$, $Z'_1 = Y_2$, alors $Z_1 a \subseteq Y_2$ et a est dans K_2 : il y a une transition $K \xrightarrow{a} Y$ dans $\mathcal{U}_{\mathfrak{L}}$. □

Tout automate en forme normale qui reconnaît le langage ω-rationnel $\mathfrak{L}$ et qui remplit la propriété d'universalité contient l'automate universel. La maximalité des factorisations implique que la fusion d'états distincts augmenterait le nombre d'ω-mots reconnus.

Proposition 3.12 *Soit $\mathcal{V}$ un automate en forme normale reconnaissant $\mathfrak{L}$ tel qu'il existe un morphisme φ de $\mathcal{U}_{\mathfrak{L}}$ dans $\mathcal{V}$. Alors φ est injectif.*

Démonstration. Par contradiction, supposons qu'il existe un morphisme non injectif φ de $\mathcal{U}_{\mathfrak{L}}$ dans $\mathcal{V}$.

i) Si on fusionne par φ un état transitoire X dans $\mathfrak{F}(\mathfrak{L})$ avec un des état d'une CFC finale. Il existe une ω-factorisation Ω telle que $\Omega_1 = A^*$, et une transition $X \xrightarrow{a} \Omega$ pour toute lettre a, dans $\mathcal{U}_{\mathfrak{L}}$. Comme $\mathcal{V}$ est en forme normale, l'image de tout successeur de X doit être dans la même CFC (finale) que $\varphi(X)$. Mais Ω est un état initial et ne peut donc pas avoir son image dans une CFC finale.

ii) Soient $X, X' \in \mathfrak{F}(\mathfrak{L})$ deux états transitoires de $\mathcal{U}_{\mathfrak{L}}$. Si $\varphi(X) = \varphi(X') = \alpha$, alors $(\mathsf{Past}_{\mathcal{U}_{\mathfrak{L}}}(X) \cup \mathsf{Past}_{\mathcal{U}_{\mathfrak{L}}}(X')) \subseteq \mathsf{Past}_{\mathcal{V}}(\alpha)$ et $(\mathsf{Fut}_{\mathcal{U}_{\mathfrak{L}}}(X) \cup \mathsf{Fut}_{\mathcal{U}_{\mathfrak{L}}}(X')) \subseteq \mathsf{Fut}_{\mathcal{V}}(\alpha)$. Pour tout X dans $\mathfrak{F}(\mathfrak{L})$, $\mathsf{Past}_{\mathcal{U}_{\mathfrak{L}}}(X) = X_1$ et $\mathsf{Fut}_{\mathcal{U}_{\mathfrak{L}}}(X) = X_2$,

et donc $(X_1 \cup X_1')(X_2 \cup X_2') \subseteq \mathsf{Past}_{\mathcal{V}}(\alpha)\mathsf{Fut}_{\mathcal{V}}(\alpha) \subseteq \mathfrak{L}$. En conséquence, par maximalité de X et X', $X_1 = X_1'$ et $X_2 = X_2'$ et donc $X = X'$.

iii) Soient $Y, Y' \in \mathfrak{F}_p(\mathfrak{L})$ deux états terminaux de $\mathcal{U}_{\mathfrak{L}}$. Si $\varphi(Y) = \varphi(Y') = \beta$ dans $\mathcal{V}$ alors $(Y_1 \cup Y_1') = (\mathsf{Past}_{\mathcal{U}_{\mathfrak{L}}}(Y) \cup \mathsf{Past}_{\mathcal{U}_{\mathfrak{L}}}(Y')) \subseteq \mathsf{Past}_{\mathcal{V}}(\beta)$ et $(Y_2 \cup Y_2') = (\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Y) \cup \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y', Y')) \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \beta)$. Donc $(Y_1 \cup Y_1')(Y_2 \cup Y_2')^\omega \subseteq \mathsf{Past}_{\mathcal{V}}(\beta)\mathsf{Trans}_{\mathcal{V}}(\beta, \beta)^\omega \subseteq \mathfrak{L}$ et par maximalité, $Y = Y'$.

iv) Soient $Z, Z' \in \mathcal{Z}_Y$ deux états de la même CFC dont l'état final, Y, est tel que $\varphi(Y) = \beta$. Tout d'abord, nous prouvons que $\mathsf{Trans}_{\mathcal{V}}(\beta, \beta) = Y_2$. Comme $Y_2 = \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Y) \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \beta)$ et comme $Y_1 \subseteq \mathsf{Past}_{\mathcal{V}}(\beta)$, alors $Y_1 {Y_2}^\omega \subseteq Y_1 \mathsf{Trans}_{\mathcal{V}}(\beta, \beta)^\omega \subseteq \mathfrak{L}$. Ainsi, par maximalité de Y_2, $\mathsf{Trans}_{\mathcal{V}}(\beta, \beta) = Y_2$.
Si $\varphi(Z) = \varphi(Z') = \gamma$, alors $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Z, Y) \cup \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Z', Y) \subseteq \mathsf{Trans}_{\mathcal{V}}(\gamma, \beta)$ et de la même façon $\mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Z) \cup \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Z') \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \gamma)$. On a donc $(Z_1 \cup Z_1')(Z_2 \cup Z_2') \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \gamma)\mathsf{Trans}_{\mathcal{V}}(\gamma, \beta) \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \beta) = Y_2$, et par maximalité de Z et Z', $Z = Z'$.

v) Soit $Z \in \mathcal{Z}_Y$ appartenant à la CFC avec Y pour état final. Si $\varphi(Z) = \varphi(Y) = \beta$, alors $Z_1 = \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Y, Z) \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \beta) = Y_2$, et $Z_2 = \mathsf{Trans}_{\mathcal{U}_{\mathfrak{L}}}(Z, Y) \subseteq \mathsf{Trans}_{\mathcal{V}}(\beta, \beta) = Y_2$. Par maximalité de Z, $Z = (Y_2, Y_2)$, ce qui est impossible.

On en conclut donc qu'il n'existe aucune fusion d'état dans $\mathcal{U}_{\mathfrak{L}}$ qui ne préserve à la fois le langage reconnu et la forme normale. □

Les Propositions 10, 11 et 12 nous amène au résultat principal.

Théorème 3.13 *Pour tout langage ω-rationnel $\mathfrak{L}$, l'automate universel de $\mathfrak{L}$ est le plus petit automate de Büchi en forme normale qui reconnaît $\mathfrak{L}$ dans lequel tout automate de Büchi équivalent et en forme normale s'envoie par morphisme.*

EXEMPLE 3.4 Nous reprenons le langage ω-rationnel $\mathfrak{L}_2 = {\mathcal{L}_1}^\omega$, où $\mathcal{L}_1 = A^*(aa + bb)A^*$, reconnu par l'automate de Büchi (prophétique) $\mathfrak{B}_2$ de la figure 3.2 dont la normalisation est représentée sur la figure 3.3.

On note $K_a = \mathcal{L}_1 + aA^*a + a$ et $K_b = \mathcal{L}_1 + bA^*b + b$. Les seules ω-factorisations sont $(A^*, \mathfrak{L}_2)$ et $(\emptyset, A^\omega)$ et les ω-factorisations pures sont (A^+, K_a), (A^+, K_b) et $(\emptyset, A^+)$. Les factorisations positives de K_a sont :

$$(A^+, \mathcal{L}_1), (\mathcal{L}_1 + aA^* + A^*a, K_a), (K_a + A^*b, \mathcal{L}_1 + bA^*a),$$
$$(\mathcal{L}_1 + A^*a, \mathcal{L}_1 + aA^*), (\mathcal{L}_1 + aA^*, \mathcal{L}_1 + A^*a), (\mathcal{L}_1 + A^*b, \mathcal{L}_1 + bA^*),$$
$$(K_a, \mathcal{L}_1 + A^*a + aA^*), (\mathcal{L}_1 + aA^*b, K_a + bA^*), (\mathcal{L}_1, A^+)$$

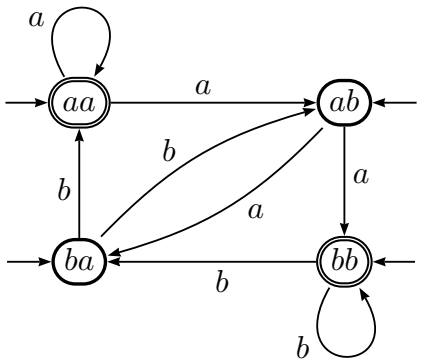


FIGURE 3.2 – L'automate de Büchi prophétique $\mathfrak{B}_2$ reconnaissant $\mathfrak{L}_2$.

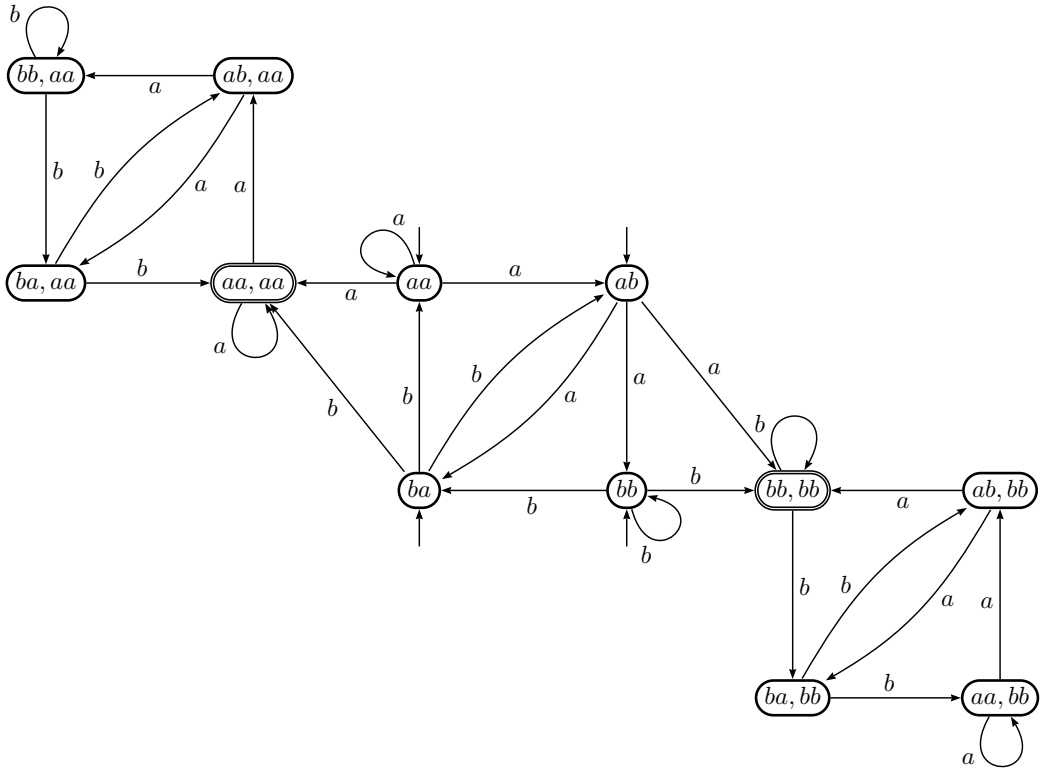


FIGURE 3.3 – La normalisation $\mathfrak{B}_{2\mathsf{nf}}$ de l'automate $\mathfrak{B}_2$.

et les factorisations positives de K_b sont :

$$(A^+, \mathcal{L}_1), (\mathcal{L}_1 + bA^* + A^*b, K_b), (K_b + A^*a, \mathcal{L}_1 + aA^*b)$$
$$(\mathcal{L}_1 + A^*b, \mathcal{L}_1 + bA^*), (\mathcal{L}_1 + bA^*, \mathcal{L}_1 + A^*b), (\mathcal{L}_1 + A^*a, \mathcal{L}_1 + aA^*),$$
$$(K_b, \mathcal{L}_1 + A^*b + bA^*), (\mathcal{L}_1 + bA^*a, K_b + aA^*), (\mathcal{L}_1, A^+)$$

L'automate universel de $\mathfrak{L}_2$ se compose donc de deux états transients et de trois CFCs finales (dont une n'est pas accessible). Pour des raisons de lisibilité, on préfère représenter l'écorché de l'automate universel de $\mathfrak{L}_2$, ici sur la figure 3.4. Cette notion a été introduite dans [31] pour les automates universels sur mots finis pour mettre en évidence certaines propriétés des automates universels et aussi pour permettre une lecture plus facile de ces derniers en "supprimant" des transitions qui peuvent être déduites à partir d'autres transitions dites "utiles". Nous avons adapté ce concept pour le cas des mots infinis en l'utilisant sur les factorisations positives de chaque CFC et le principe est le suivant : soient deux états (X, Y) et (X', Y') de la même CFC telles que $Y \subseteq Y'$, alors pour toute transition partant de (X, Y) vers un état q étiquetée par une lettre a, il existe une transition partant de (X', Y') vers ce même état q. Ainsi on peut "remplacer" toutes ces transitions de (X', Y') vers q par une unique ε-transition partant de (X', Y') et arrivant sur (X, Y).

Comme on peut le vérifier, il existe un morphisme $\varphi\colon \mathfrak{B}_{2\mathsf{nf}} \to \mathcal{U}_{\mathfrak{L}_2}$ de la normalisation de $\mathfrak{B}_2$ dans l'automate universel de $\mathfrak{L}_2$ défini de la façon suivante : tous les états transients dans $\mathfrak{B}_{2\mathsf{nf}}$ (c'est à dire aa,ab,ba et bb) s'envoient sur l'état (A^*, L_1) dans $\mathcal{U}_{\mathfrak{L}_2}$. Les états de la CFC finale de aa, aa dans $\mathfrak{B}_{2\mathsf{nf}}$ s'envoient sur les états de la CFC finale de (A^+, K_a) dans $\mathcal{U}_{\mathfrak{L}_2}$ de la façon suivante :

$$\varphi(aa, aa) = (A^+, K_a) \qquad \varphi(bb, aa) = (A^+, L_1)$$
$$\varphi(ab, aa) = (L_1 + A^*a, L_1 + aA^*) \qquad \varphi(ba, aa) = (L_1 + A^*b, L_1 + bA^*)$$

Les états de la CFC finale de bb, bb dans $\mathfrak{B}_{2\mathsf{nf}}$ s'envoient sur les états de la CFC finale de (A^+, K_b) dans $\mathcal{U}_{\mathfrak{L}_2}$ de la façon suivante :

$$\varphi(bb, bb) = (A^+, K_b) \qquad \varphi(aa, bb) = (A^+, L_1)$$
$$\varphi(ba, bb) = (L_1 + A^*b, L_1 + bA^*) \qquad \varphi(ab, bb) = (L_1 + A^*a, L_1 + aA^*)$$

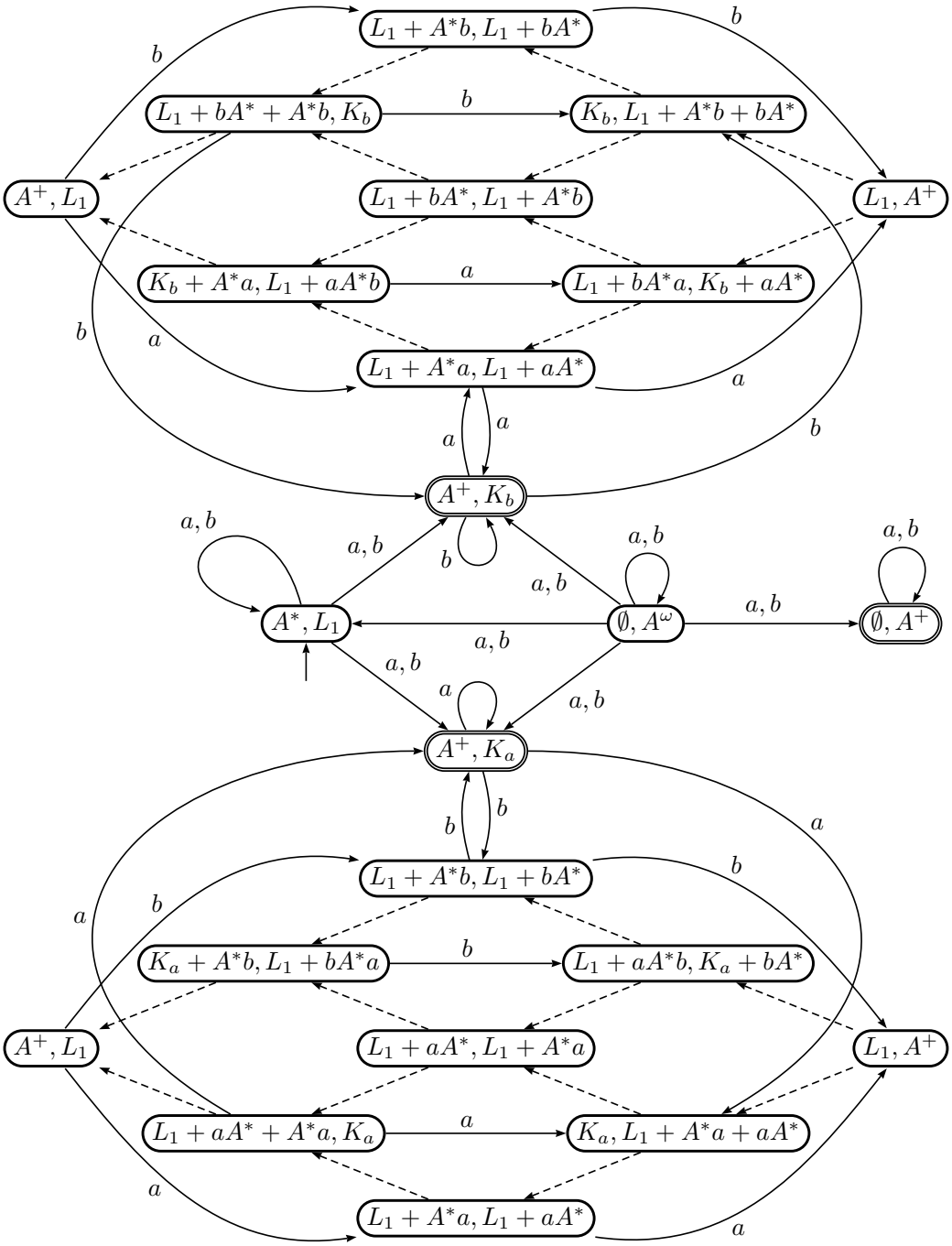


FIGURE 3.4 – L'automate universel $\mathcal{U}_{\mathfrak{L}_2}$ de $\mathfrak{L}_2$.

4 Conclusion

Nous avons introduit les factorisation de langages ω-rationnels. Elles ont abouti à la définition de l'automate universel pour de tels langages qui est effectivement calculable étant donné que tout type d'ω-factorisations est calculable dans un ω-semigroupe fini qui reconnaît le langage. Il reste la question du calcul efficace des ω-factorisations pures dans l'automate prophétique. Comme pour les langages rationnels sur les mots finis, l'automate universel peut être utilisé, malgré sa taille qui peut être au plus exponentielle en la taille de l'ω-semigroupe syntaxique, pour calculer un automate fini non-déterministe de taille minimum[42], ou dans des preuves théoriques sur l'existence d'automates avec des propriétés spécifiques (hauteur d'étoile [34], réversibilité [32], *etc.*).

De plus, comme tout automate a une image par morphisme dans l'automate universel, il peut être utilisé pour la construction d'automates avec un petit nombre d'états. Dans le cas des automates de Büchi, comme l'automate universel est en forme normale, le calcul d'un automate de Büchi avec un nombre minimal d'état n'est pas aussi direct que dans le cas des automate fini non-déterministe. Il s'agit d'une question qui nécessite une recherche plus approfondie.

Chapitre IV

Automates two-way à multiplicité

Une question classique en théorie des automates concerne la puissance d'expression d'un outil et plus particulièrement la différence entre les systèmes unidirectionnels et les systèmes bidirectionnels. Il est bien connu que les automates bidirectionnels peuvent être convertis en automates unidirectionnels et donc reconnaissent la même famille de langages [46, 43]. Cela dit depuis quelques années, il s'est avéré que cette équivalence n'est plus vraie dans le cas des automates bidirectionnels pondérés. Ainsi, dans ce chapitre, nous traitons des versions à multiplicité de ces outils. Dans [2] une construction pour passer d'un automate bidirectionnel à multiplicité (2-WFA pour two-way weighted finite automaton) à un automate unidirectionnel à multiplicité (1-WFA) est décrite ; cela requiert quelques hypothèses sur le semi-anneau (la commutativité) et sur l'automate lui-même. Une étude pour permettre de caractériser les langages reconnus par les transducteurs bidirectionnels a été proposée dans [22].

Dans une première partie, nous étudions plus précisément cette transformation et nous montrons qu'elle préserve la non-ambiguïté, mais pas le déterminisme. Nous présentons également une construction pour convertir un 1-WFA non-ambigu en un 2-WFA déterministe. En conséquence, sur les semi-anneaux commutatifs, à la différence du cas des automates unidirectionnels, les 2-WFA non-ambigus ne sont pas plus puissants que les 1-WFA déterministes.

Dans la seconde partie, nous traitons le cas particulier des automates min-plus qui se sont vus accordé beaucoup d'attention ces 3 dernières décennies. Le semi-anneau $\mathbb{N}$-min-plus est l'une des extensions les plus simples du semi-anneau de Boole et les automates min-plus sont donc une extension très naturelle des automates classiques avec des applications très variées dans le

traitement des langages naturels ou encore de l'optimisation. Ils constituent en effet de très puissants outils et sont utilisés dans certains résultats très importants comme le problème de la hauteur d'étoile.

Ici, nous étudions les automates bidirectionnels min-plus. Quand les poids sont tous positifs, nous étendons le résultat classique [46, 43] qui établit qu'un automate bidirectionnel fini est équivalent à un automate unidirectionnel fini.

Ensuite, nous montrons qu'en règle générale certains mots peuvent être acceptés dans un automate bidirectionnel par une famille de calculs dont la borne inférieure des poids est $-\infty$. Dans ce cas, le comportement de l'automate peut ne pas être rationnel. Nous prouvons que l'existence de tels mots acceptés est décidable. En revanche, nous prouvons que pour un automate $\mathbb{Z}$-min-plus donné, il est indécidable s'il existe un mot accepté avec un poids fini. Le contenu de ce chapitre a été publié dans deux articles [10] et [11].

1 Automates à multiplicité

Jusqu'à présent, les automates que nous avons manipulés se "contentaient" d'accepter ou non des éléments d'un monoïde. Il est possible d'être plus précis en comptant les calculs réussis pour un élément donné. Les compter ne veux pas nécessairement dire à l'aide d'entiers, tout ensemble dont la structure permet de faire les calculs qui nous intéressent serait acceptable. Cette structure est celle des semi-anneaux ...

En toute généralité, un automate sur un monoïde M à multiplicité dans un semi-anneau $\mathbb{K}$, que l'on appellera $\mathbb{K}$-automate sur M, est un graphe étiqueté par des éléments de $\mathbb{K}\langle\langle M\rangle\rangle$, l'ensemble des séries sur M à coefficients dans $\mathbb{K}$, et est associé à deux fonctions I (initiale) et T (finale) de l'ensemble des états dans $\mathbb{K}\langle\langle M\rangle\rangle$. Pour notre part, pour éviter d'aborder des sujets qui seraient lourds et inutiles dans cette étude comme le problème de la topologie (lorsque M est quelconque) qui nécessiterait des notions de limite et de continuité, nous allons nous contenter de traiter le cas des automates sur le monoïde libre qui permettent une définition quasi directe d'une topologie sur $\mathbb{K}\langle\langle A^*\rangle\rangle$.
De plus, à l'instar des automates "classiques", le fait de restreindre les étiquettes des transitions à une unique lettre n'est en rien préjudiciable quant à la puissance des automates. C'est pourquoi nous ne traiterons que ce dernier cas.

Définition 1.1 Un automate sur A à multiplicité (ou pondéré) dans un semi-anneau $\mathbb{K}$ est un 5-uplet $\mathcal{A} = \langle Q, A, E, I, T\rangle$ où Q est l'ensemble des états de $\mathcal{A}$, E est une fonction partielle de $Q \times A \times Q$ dans $\mathbb{K}$, et le support de

E, $\underline{E}$, est l'ensemble des transitions de $\mathcal{A}$. I et T sont des fonctions partielles de Q dans $\mathbb{K}$ appelées respectivement fonction initiale et finale. Le support de I, $\underline{I}$, est l'ensemble des états initiaux de $\mathcal{A}$, et le support de T, $\underline{T}$, est l'ensemble des états finals de $\mathcal{A}$.

Pour une transition t dans $\underline{E}$, on appellera $k = E(t)$ le coefficient (ou le poids) de t (si $k = 0_{\mathbb{K}}$ on considère plutôt qu'il n'y a pas de transition). On dira qu'un état p est initial (*resp.* final) si $I(p) = k$ (*resp.* $T(p) = k$) avec $k \neq 0_{\mathbb{K}}$ qu'on appellera le poids initial (*resp.* final) de p. Soit t une transition dans $\underline{E}$; si $t = (p, a, q)$, nous notons $\sigma(t) = p$, $\tau(t) = q$, $\lambda(t) = a$.

Les définitions de calcul, de calcul réussi et d'étiquette de calcul restent les mêmes. On ajoutera seulement que le poids d'un calcul est égal au produit (dans l'ordre) des poids des transitions qui le composent et que le poids d'un mot u de A^* dans l'automate est égal à la somme des poids des calculs réussis étiquetés par u.

EXEMPLE 1.1 Si l'on considère l'automate $\mathcal{A}_1$ non plus comme un $\mathbb{B}$-automate mais comme un $\mathbb{N}$-automate, alors la série qu'il reconnaît est la série s_a qui associe à chaque mot u de A^* le nombre de a qu'il contient.

REMARQUE 1.1 De la même façon que pour les automates "sans multiplicité", on suppose sans perte de généralité que les transitions initiales et finales sont toutes étiquetées par 1_{A^*}.
De plus on constate aisément qu'entre deux états, il ne peut pas y avoir deux transitions étiquetées par la même lettre.

Avec de telles définitions, on comprend bien qu'un $\mathbb{K}$-automate ne va pas seulement accepter ou non des mots mais va en plus leur associer un poids dans $\mathbb{K}$. C'est pourquoi le comportement d'un $\mathbb{K}$-automate n'est plus seulement l'ensemble des mots qu'il accepte mais bien une série de A^* dans $\mathbb{K}$.

Définition 1.2 Deux $\mathbb{K}$-automates sur A^* sont équivalents si et seulement s'ils réalisent la même série.

REMARQUE 1.2 La décidabilité de l'équivalence dépend du semi-anneau $\mathbb{K}$ ou de la forme des $\mathbb{K}$-automates. Sur les $\mathbb{B}$-automates, l'équivalence est bien sûr décidable de même que si $\mathbb{K}$ est un corps. Si $\mathbb{K}$ est le semi-anneau Rat A^* des parties rationnelles de A^*, l'équivalence est décidable si les $\mathbb{K}$-automates, qu'on appelle transducteurs, sont fonctionnels. Pour notre part, nous allons manipuler des automates sur le semi-anneau tropical qui sera présenté plus tard et avec lequel l'équivalence est décidable si le $\mathbb{K}$-automate

est non-ambigu.

On peut également remarquer que les automates que l'on manipulait jusque là, les automates "sans multiplicité", sont en réalité des $\mathbb{B}$-automates, autrement dit des automates à multiplicité dans $\mathbb{B}$. De ce fait, on peut définir une construction qui passe d'un $\mathbb{K}$-automate au $\mathbb{B}$-automate associé en basculant à $1_{\mathbb{B}}$ les poids non nuls (différents de $0_{\mathbb{K}}$) de toutes les transitions de l'automate de départ. C'est sur la base de cette remarque que l'on pose les définitions suivantes :

Définition 1.3 Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un $\mathbb{K}$-automate sur A. L'automate sous-jacent (appelé également automate support) de $\mathcal{A}$ est l'automate $\mathcal{B} = \langle Q, A, \underline{E}, \underline{I}, \underline{T}\rangle$ ("sans multiplicité").
Réciproquement, soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate ("sans multiplicité") sur A. L'*automate caractéristique* de $\mathcal{A}$ est le $\mathbb{K}$-automate ($\mathbb{K}$ étant fixé) $\mathcal{B} = \langle Q, A, F, J, U\rangle$ tel que :

$$\forall p \in Q, \quad J(p) = 1_{\mathbb{K}} \Longleftrightarrow p \in I \text{ et } U(p) = 1_{\mathbb{K}} \Longleftrightarrow p \in T$$

et

$$F = \{(p, a, 1_{\mathbb{K}}, q) \mid (p, a, q) \in E\}$$

Avec cette définition, on étend certains concepts des automates "classiques" aux $\mathbb{K}$-automates : un état est accessible (*resp.* co-accessible et utile) si l'état correspondant dans l'automate support est accessibles (*resp.* co-accessible et utile). Un $\mathbb{K}$-automate est accessible, co-accessible, émondé, déterministe, co-déterministe, complet ou encore co-complet si son automate support l'est.
Nous ne rentrerons pas dans les détails quant au principe de déterminisation des $\mathbb{K}$-automates car il s'avère nettement plus compliqué que dans le cas des automates "classiques" et car il n'entrera pas en compte dans cette étude.

Comme pour les automates classiques, il peut s'avérer plus pratique de travailler avec une autre représentation que la représentation ensembliste.

Définition 1.4 Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate à coefficients dans $\mathbb{K}$. La représentation linéaire de $\mathcal{A}$ (qu'on appellera également $\mathbb{K}$-représentation) est un triplet (λ, μ, γ) tel que :

- λ est un vecteur (colonne) de $\mathbb{K}^Q$ tel que, quel que soit p dans Q, $\lambda_p = I(p)$
- γ est un vecteur (ligne) de $\mathbb{K}^Q$ tel que, quel que soit p dans Q, $\gamma_p = T(p)$
- μ est un morphisme de A^* dans $\mathbb{K}^{Q\times Q}$ tel que, quels que soient p et q dans Q et quel que soit a dans A, $\mu(a)_{p,q} = k \Longleftrightarrow E(p, a, q) = k$.

Une telle représentation définit une application de A^* dans $\mathbb{K}$ qui associe à un mot $u = u_1 \dots u_n$ son poids dans $\mathcal{A}$, $\lambda \cdot \mu(u_1) \cdot \dots \cdot \mu(u_n) \cdot \gamma$ [1] et donc, par extension, définit une série s :

$$s = \sum_{u \in A^*} (\lambda \cdot \mu(u) \cdot \gamma) u$$

On dira ainsi que la $\mathbb{K}$-représentation (λ, μ, γ), et donc que l'automate $\mathcal{A}$, reconnaît la série s. Ce qui nous amène à la définition de série $\mathbb{K}$-reconnaissable :

Définition 1.5 Un série $\mathbb{K}\langle\langle A^* \rangle\rangle$ est $\mathbb{K}$-reconnaissable si et seulement si elle est reconnu par une $\mathbb{K}$-représentation. On note $\mathbb{K}\mathrm{Rec}\ A^*$ l'ensemble des séries $\mathbb{K}$-reconnaissables sur A^*.

Avec la définition qu'on en a fait, il est clair qu'il y a équivalence entre série reconnaissable par morphisme et série reconnaissable par $\mathbb{K}$-automate. Ces nouvelles définitions aboutissent à l'extension beaucoup plus générale du célèbre théorème de Kleene :

Théorème 1.1 *[Kleene-Schutzenberger]*

$$\mathbb{K}Rec\ A^* = \mathbb{K}Rat\ A^*$$

2 Automates bidirectionnels à multiplicité

Dans cette partie, nous étendons les automates à multiplicité en ajoutant la possibilité d'une lecture bidirectionnelle de l'entrée.

2.1 Automates et calculs

Pour tout alphabet A, nous supposons qu'il existe deux symboles supplémentaires $\vdash$ et $\dashv$ qui sont des marqueurs au début et à la fin des bandes de lecture des automates. Nous notons $A_{\vdash\dashv}$ l'alphabet $A \cup \{\vdash, \dashv\}$. Pour tout mot $w = w_0 \dots w_n$ dans A, $w_{\vdash\dashv}$ est le mot dans $A_{\vdash\dashv}$ qui est égal à $\vdash w_0 \dots w_n \dashv$.

Les $\mathbb{K}$-automates bidirectionnels et unidirectionnels partagent une partie de leur définition. Un $\mathbb{K}$-automate bidirectionnel est un 5-uplet $\mathcal{A} = (Q, A, E, I, T)$ où Q est un ensemble fini d'états, A est un alphabet fini, I et T sont des fonctions partielles de Q dans $\mathbb{K}$. Le support de I, $\underline{I}$, est l'ensemble

1. ici $\cdot$ représente le produit matriciel

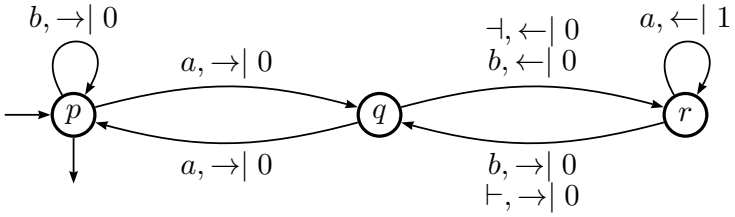


FIGURE 2.1 – Le $\mathcal{N}$-automate bidirectionnel $\mathcal{B}_1$.

des états initiaux de $\mathcal{A}$, et le support de T, $\underline{T}$, est l'ensemble des états finals de $\mathcal{A}$.

La définition des transitions, elle, diffère. Dans un $\mathbb{K}$-automate bidirectionnel, E est une fonction partielle de $Q \times (A_{\vdash} \times \{-1, +1\}) \times Q$ dans $\mathbb{K}$ et le support de E, $\underline{E}$, est l'ensemble des transitions de $\mathcal{A}$.
Soit t une transition $\underline{E}$; si $t = (p, a, d, q)$, nous notons $\sigma(t) = p$, $\tau(t) = q$, $\lambda(t) = a$, $\delta(t) = d$. Sur les figures, la valeur de δ est représentée pas une flèche gauche (-1) ou droite (+1). Par exemple, si $t = (p, a, -1, q)$ et $E_t = k$, nous notons $p \xrightarrow{a,\leftarrow|k} q$.

EXEMPLE 2.1 Soit $\mathcal{B}_1$ le $\mathcal{N}$-automate bidirectionnel de la Figure 2.1, où $\mathcal{N} = (\mathbb{N} \cup \{\infty\}, \min, +)$ est le semi-anneau tropical ; comme la loi multiplicative dans ce semi-anneau est la somme ordinaire, le poids d'un calcul dans cet automate est la somme des poids de ses transitions. Cet automate est déterministe (*cf.* Définition 1) et donc il y a un unique calcul pour chaque mot. Le comportement de cet automate est très simple. Pour tout block de 'a', il vérifie à l'aide d'une lecture gauche-droite, si la longueur de ce block est impaire ; si elle l'est, une lecture droite-gauche calcule la longueur du block ; sinon l'automate va au block de 'a' suivant. Au final, le poids d'un mot correspond à la somme des longueurs des blocks de 'a' de longueur impaire.

Définition 2.1 Soit $w = w_1 \dots w_n$ un mot de A^*, on pose $w_0 = \vdash$ et $w_{n+1} = \dashv$. Une *configuration* de $\mathcal{A}$ sur w est une paire (p, i) où i est dans $[0; n+1]$ et p est un état de $\mathcal{A}$. Un calcul ρ de $\mathcal{A}$ sur w est une séquence finie de configurations $((p_0, i_0), \dots, (p_k, i_k))$ telle que :

- $i_0 = 1$, $i_k = n + 1$, p_1 est dans $\underline{I}$ et p_k est dans $\underline{T}$;
- pour tout j dans $[0; k-1]$, il existe t_j dans $\underline{E}$ telle que
$$\sigma(t_j) = p_j,\ \tau(t_j) = p_{j+1},\ \lambda(t_j) = a_{ij} \text{ et } i_{j+1} = i_j + \delta(t_j).$$

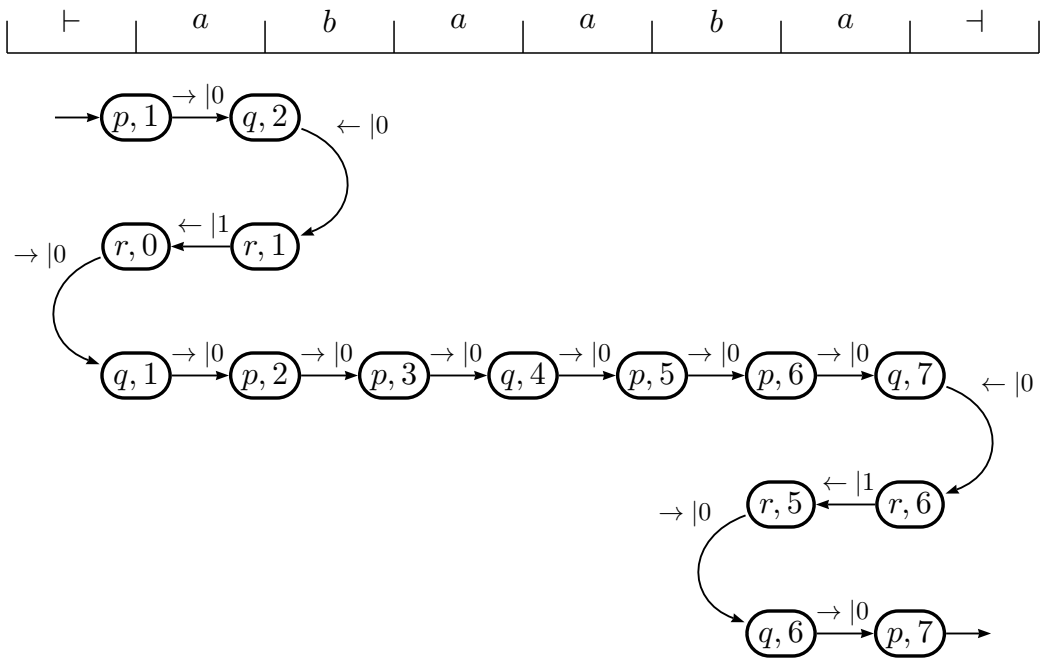


FIGURE 2.2 – Un calcul dans $\mathcal{B}_1$ sur le mot *abaaba*.

Le poids d'un tel calcul, que l'on note $|\rho|$, est $I(\sigma(t_1)) \otimes \bigotimes_{j=1}^{k} E(t_j) \otimes T(\tau(t_k))$. Le poids de w dans $\mathcal{A}$, que l'on note $\langle|\mathcal{A}|, w\rangle$, est l'addition des poids de tous les calculs qui étiquettent w dans $\mathcal{A}$. A noter qu'il peut y avoir une infinité de calculs avec la même étiquette w. Dans ce cas, pour définir le comportement de $\mathcal{A}$, il est nécessaire que $\mathbb{K}$ soit un semi-anneau topologique. Le poids d'un mot est défini si et seulement si la famille de poids de tous les calculs étiquetés par ce mot est sommable et le comportement de $\mathcal{A}$ est défini si le poids de tout mot accepté est défini. Pour plus de détails et discussion sur les semi-anneaux topologiques, se référer à [36].

EXEMPLE 2.2 Un calcul dans le $\mathcal{N}$-automate $\mathcal{B}_1$ sur un mot *abaaba* est représenté sur la Figure 2.2. Le poids de ce calcul est égal à 2.

Dans les automates bidirectionnels, dans le même calcul, il peut y avoir deux étapes au cours desquelles l'automate est dans le même état et lit la même lettre de l'entrée. Dans ce cas, on dit que le calcul contient un *circuit immobile*.

Définition 2.2 Soit $\rho = ((t_1, i_1), \ldots, (t_k, i_k))$ un calcul. S'il existe m, n in $[1, k]$, avec $m < n$ tels que $i_m = i_n$ et $\sigma(t_m) = \sigma(t_n)$, alors on dit que $((t_m, i_m), \ldots, (t_{n-1}, i_{n-1}))$ est un *circuit immobile* de ρ. Si ρ ne contient aucun circuit immobile, il est *réduit*.

Lemme 2.1 *Si un $\mathbb{K}$-automate bidirectionnel admet un calcul ρ qui n'est pas réduit, alors il admet également un calcul réduit avec la même étiquette.*

En avance sur les futurs chapitres, le lemme suivant enonce une propriété des automates de distance qui sont des automates sur un semi-anneau de type "min-+", c'est à dire avec l'opérateur *min* comme addition et l'opérateur + comme produit, et dont les poids ne sont pas négatifs.

Lemme 2.2 *Soit $\mathcal{A}$ un automate de distance bidirectionel sur un alphabet A. Pour tout w dans A^*, $\langle|\mathcal{A}|, w\rangle$ est le poids d'un calcul réduit de w.*

Démonstration. Par contradiction, supposons que, pour un mot w, il n'y a pas de calcul réduit dans $\mathcal{A}$ étiqueté par w avec un poids minimal. Alors soit $\rho = ((t_1, i_1), \ldots, (t_l, i_l))$ l'un des calculs non réduits étiquetés par w les plus courts avec un poids minimal. Étant donné qu'il n'est pas réduit, il existe j et k, avec $j < k$, tels que $i_j = i_k$ et $\sigma(t_j) = \sigma(t_k)$. Donc il existe un calcul valide $\rho' = ((t_1, i_1), \ldots, (t_{j-1}, i_{j-1}), (t_k, i_k), \ldots, (t_l, i_l))$ étiqueté par w avec $|\rho'| \leq |\rho|$ ce qui est contradictoire. □

Définition 2.3 Un automate bidirectionnel $\mathcal{A}$ est *non-ambigu* si tout mot étiquette au plus un calcul.

Les automates non-ambigus n'ont clairement que des calculs réduits.

2.2 $\mathbb{K}$-revêtements

Nous avons déjà présenté le concept de revêtement sous sa forme basique (c'est à dire pour les automates unidirectionnels "classiques"). Cette notion a été étendue aux automates à multiplicité unidirectionnels (on parle alors de $\mathbb{K}$-revêtement, pour plus de détails voir [45]) et nous l'avons, pour notre part, étendue aux automates à multiplicité bidirectionnels. La seule différence entre ces deux versions est l'ajout de la direction des transitions qui ne constitue qu'une simple information supplémentaire.

Définition 2.4 Soient $\mathcal{A} = (Q, A, E, I, T)$ et $\mathcal{B} = (R, A, F, J, U)$ deux $\mathbb{K}$-automates bidirectionnels. Une application de Q dans R est un *morphisme* si,
i) $\forall p \in \underline{I},\ J(\varphi(p)) = I(p)$;
ii) $\forall p \in \underline{T},\ U(\varphi(p)) = T(p)$;
iii) $\forall t = (p, a, \delta, q) \in \underline{E},\ \tilde{\varphi}(t) = (\varphi(p), a, \delta, \varphi(q)) \in \underline{F}$ et $F(\tilde{\varphi}(t')) = E(t)$.
Le morphisme est surjectif si $\varphi(Q) = R$, $\varphi(\underline{I}) = \underline{J}$, $\varphi(\underline{T}) = \underline{U}$, et $\tilde{\varphi}(\underline{E}) = \underline{F}$.

Définition 2.5 Soient $\mathcal{A} = (Q, A, E, I, T)$ et $\mathcal{B} = (R, A, F, J, U)$ deux $\mathbb{K}$-automates bidirectionnels. $\mathcal{A}$ est un $\mathbb{K}$-*revêtement* de $\mathcal{B}$ s'il existe un morphisme surjectif φ de $\mathcal{A}$ dans $\mathcal{B}$ tel que

$$
\begin{aligned}
&i)\ \forall r \in \underline{U},\ \varphi^{-1}(r) \subseteq \underline{T} \\
&ii)\ \forall r \in \underline{J},\ \exists! p \in \varphi^{-1}(r) \cap \underline{I} \\
&iii)\ \forall t \in \underline{F}, \forall p \in \varphi^{-1}(\sigma(t)), \exists! t' \in \tilde{\varphi}^{-1}(t), \sigma(t') = p.
\end{aligned}
$$

$\mathcal{A}$ est un $\mathbb{K}$-*co-revêtement* de $\mathcal{B}$ s'il existe un morphisme surjectif φ de $\mathcal{A}$ dans $\mathcal{B}$ tel que

$$
\begin{aligned}
&i)\ \forall r \in \underline{J},\ \varphi^{-1}(r) \subseteq \underline{I} \\
&ii)\ \forall r \in \underline{U},\ \exists! p \in \varphi^{-1}(r) \cap \underline{T} \\
&iii)\ \forall t \in \underline{F}, \forall q \in \varphi^{-1}(\tau(t)), \exists! t' \in \tilde{\varphi}^{-1}(t), \tau(t') = q.
\end{aligned}
$$

Proposition 2.3 *Soient $\mathcal{A}$ et $\mathcal{B}$ deux $\mathbb{K}$-automates bidirectionnels. Si $\mathcal{A}$ est un $\mathbb{K}$-revêtement (resp. un $\mathbb{K}$-co-revêtement) de $\mathcal{B}$, le morphisme correspondant φ induit une bijection entre les calculs dans $\mathcal{A}$ et ceux dans $\mathcal{B}$ telle que tout calcul dans $\mathcal{A}$ et son image dans $\mathcal{B}$ ont la même étiquette et le même poids.*

Démonstration. Supposons que $\mathcal{A}$ est un $\mathbb{K}$-revêtement de $\mathcal{B}$. Soit w un mot et soit $((p_0, i_0), \dots, (p_k, i_k))$ un calcul sur w dans $\mathcal{A}$. Pour tout j dans $[0; k]$, on pose $r_j = \varphi(p_j)$; d'après la définition d'un morphisme, $((r_0, i_0), \dots, (r_k, i_k))$ est un calcul sur w dans $\mathcal{B}$ avec le même poids.
Réciproquement, soit $((r_0, i_0), \dots, (r_k, i_k))$ un calcul dans $\mathcal{B}$. Soit p_0 l'unique état initial dans $\varphi^{-1}(r_0)$. Pour tout j dans $[0; k-1]$, soit $\delta_j = i_{j+1} - i_j$; la configuration (r_{j+1}, r_{j+1}) est atteinte à partir de la configuration (r_j, i_j) en utilisant la transition $(r_j, w_j, \delta_j, r_{j+1})$; inductivement, nous définissons p_{j+1} comme l'unique état dans $\varphi^{-1}(r_{j+1})$ tel que $(p_j, w_j, \delta_j, p_{j+1})$ est une transition de $\mathcal{A}$. Ainsi, $((p_0, i_0), \dots, (p_k, i_k))$ est un calcul sur w dans $\mathcal{A}$. Donc, tout calcul ρ de $\mathcal{B}$ est associé de façon unique à un calcul de $\mathcal{A}$ dont l'image par φ est ρ.

La preuve est similaire pour les $\mathbb{K}$-co-revêtements. □

Cette proposition implique qu'un automate bidirectionnel et son $\mathbb{K}$-revêtement (*resp.* $\mathbb{K}$-co-revêtement) sont équivalents ; de plus, si un automate bidirectionnel est non-ambigu, alors n'importe lequel de ses $\mathbb{K}$-(co-)revêtements l'est aussi.

2.3 δ-localité et δ-normalisation

Définition 2.6 Soit $\mathcal{A}$ un $\mathbb{K}$-automate bidirectionnel. Si, pour tout état p de $\mathcal{A}$, toute transition sortant de p a la même direction, alors $\mathcal{A}$ est δ-local. Si $\mathcal{A}$ est δ-local et si, pour tout état p de $\mathcal{A}$, toute transition arrivant sur p a la même direction, alors $\mathcal{A}$ est δ-normalisé.

Si Q est l'ensemble des états d'un $\mathbb{K}$-automate bidirectionnel, on note Q_+ (*resp.* Q_-) l'ensemble des états p tels que, pour toute transition t sortant de p, $\delta(t) = +1$ (*resp.* $\delta(t) = -1$) ; par convention, si p n'a pas de transition sortante, p est dans Q_+. Pour tout état p de Q_+ (*resp.* Q_-), on pose $\delta(p) = 1$ (*resp.* $\delta(p) = -1$). Si $\mathcal{A}$ est un $\mathbb{K}$-automate δ-local, $\{Q_+, Q_-\}$ est une partition de Q.

Si $\mathcal{A}$ est un automate δ-local, pour tout état p de Q, on pose $\delta_{\mathbf{O}}(p) = \delta(t)$, où t est n'importe laquelle des transitions sortant de p ; si $\mathcal{A}$ est δ-normalisé,

on pose également $\delta_{\mathbf{I}}(p) = \delta(t)$, où t est n'importe laquelle des transitions arrivant en p.

Proposition 2.4 *Tout $\mathbb{K}$-automate bidirectionnel admet un $\mathbb{K}$-co-revêtement δ-local.*

Démonstration. Dans cette preuve, on note $\pm = \{-1, +1\}$. Soit $\mathcal{A} = (R, A, F, J, U)$ un $\mathbb{K}$-automate bidirectionnel et soit $P = R \setminus (R_+ \cup R_-)$ l'ensemble des états de $\mathcal{A}$ tels qu'il y a au moins deux transitions avec des directions différentes sortant de chacun des états. Soient P_+ et P_- deux copies de P et soit $Q = R_+ \cup R_- \cup P_+ \cup P_-$. Soit φ l'application canonique de Q dans R : elle associe à chaque élément de P_+ ou de P_- l'élément correspondant de P.

Soit $\mathcal{B}' = (Q, A, E, I, T)$ le $\mathbb{K}$-automate défini par :

$$\underline{I} = \varphi^{-1}(\underline{J}); \quad \underline{T} = \varphi^{-1}(\underline{U}) \setminus P_-;$$
$$\underline{E} = \{(p, a, d, q) \mid (p, d) \in (P_+ \cup R_+) \times \{+1\} \cup (P_- \cup R_-) \times \{-1\}\};$$
$$\forall p \in \underline{I}, I(p) = J(\varphi(p)),\ \forall p \in \underline{T}, T(p) = U(\varphi(p)),\ \forall t \in \underline{E}, E(t) = F(\tilde{\varphi}(p)).$$

Le $\mathbb{K}$-automate $\mathcal{B}'$ est δ-local et c'est un $\mathbb{K}$-co-revêtement de $\mathcal{A}$. □

Comme nous l'avons vu, pour rendre un automate bidirectionnel δ-local, on construit un revêtement en scindant en deux, les états qui ont des transitions sortantes avec des directions différentes. La construction duale qui consiste à scinder les états qui ont des transitions entrantes avec des directions différentes conserve l'éventuelle δ-localité de l'automate bidirectionnel de départ. Si on applique cette dernière construction à un automate bidirectionnel δ-local, on obtient un automate bidirectionnel δ-normalisé.

EXEMPLE 2.3 L'automate $\mathcal{B}_1$ de la Figure 2.1 n'est pas δ-normalisé (il n'est pas δ-local) : dans les états q et r, il y a des transition sortantes avec une direction de -1 et d'autres avec une direction de $+1$. L'automate $\mathcal{B}'_1$ de la Figure 2.3 est un $\mathbb{K}$-automate δ-normalisé équivalent.

2.4 Slices

Définition 2.7 Soit $\mathcal{A} = (Q, A, E, I, T)$ un $\mathbb{K}$-automate bidirectionnel et soit $w = w_1 \dots w_k$ un mot. $\rho = ((p_1, i_1), \dots (p_n, i_n))$ un calcul étiqueté par w, et j dans $[1; k+1]$. Soit h la sous-séquence de toutes les paires (p_m, i_m) telles que $(i_m, i_{m+1}) = (j, j+1)$ ou $(i_{m-1}, i_m) = (j, j-1)$.
Le j-ième *slice* de ρ est le vecteur $s^{(j)}$ d'états obtenu par la projection du premier élément de chaque paire de h.

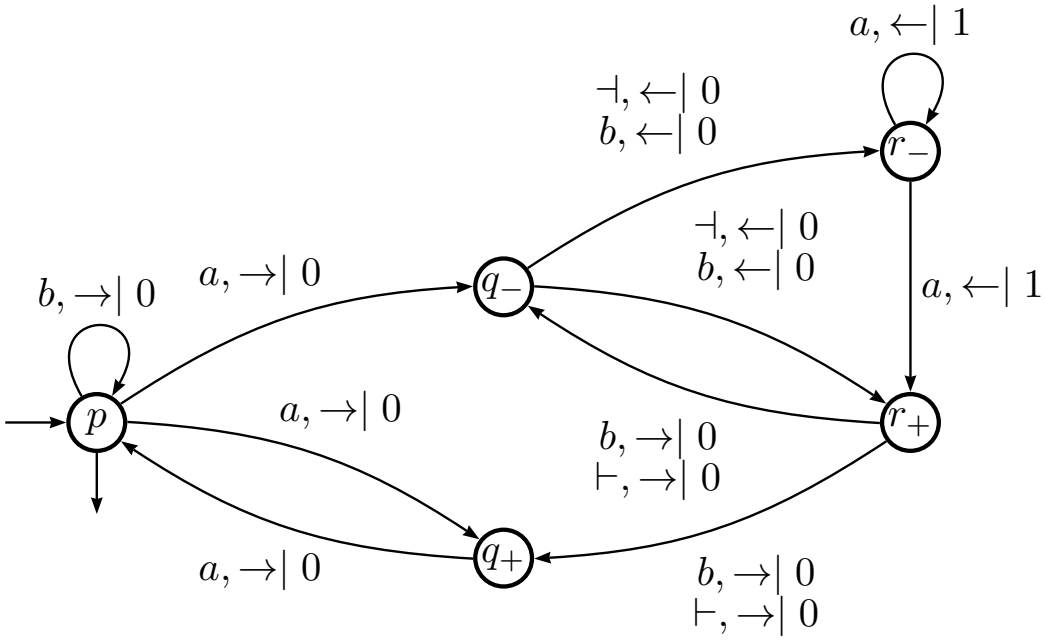


FIGURE 2.3 – Le $\mathbb{K}$-automate bidirectionnel δ-normalisé $\mathcal{B}'_1$.

La *signature* $S(\rho)$ de ρ est la séquence composée de ses slices. Les slices que nous définissons ici ne correspondent pas exactement aux *crossing sequences* définies dans [2].

EXEMPLE 2.4 Le vecteur $\begin{bmatrix} q \\ r \\ p \end{bmatrix}$ est le second (et le septième) slice du calcul de la Figure 2.2. La signature de ce calcul est :

$$\begin{pmatrix} p & q & & & & p & q \\ r, & r, & p, & q, & p, & r, & r \\ q & p & & & & q & p \end{pmatrix}. \tag{2.1}$$

La signature du calcul (unique) sur le mot *abaaba* dans le $\mathbb{K}$-automate $\mathcal{B}'_1$ est

$$\begin{pmatrix} p & q_- & & & & p & q_- \\ r_+, & r_-, & p, & q_+, & p, & r_+, & r_- \\ q_+ & p & & & & q_+ & p \end{pmatrix}. \tag{2.2}$$

Soit $\mathcal{A} = (Q, A, E, I, T)$ un $\mathbb{K}$-automate bidirectionnel δ-local. Pour définir un $\mathbb{K}$-automate unidirectionnel à partir des slices de $\mathcal{A}$, nous considérons l'ensemble X de sous-vecteurs de slices, qui sont des vecteurs v dans Q^* de taille impaire ; soit Y l'ensemble des vecteurs v dans Q^* avec une taille paire.

On définit inductivement deux fonctions partielles $\theta : X \times A \times X \to \mathbb{K}$ et $\eta : Y \times A \times Y \to \mathbb{K}$ par :

$$\begin{aligned}
&\eta(\varepsilon, a, \varepsilon) = 0_{\mathbb{K}},\\
\forall p, q \in Q, \qquad \delta(p) = 1 \Longrightarrow \forall u, v \in Y,\ &\theta(pu, a, qv) = E(p, a, 1, q) + \eta(u, a, v),\\
&\eta(u, a, pqv) = E(p, a, 1, q) + \eta(u, a, v),\\
\delta(p) = -1 \Longrightarrow \forall u, v \in X,\ &\theta(pqu, a, v) = E(p, a, -1, q) + \theta(u, a, v),\\
&\eta(qu, a, pv) = E(p, a, -1, q) + \theta(u, a, yv).
\end{aligned} \tag{2.3}$$

Étant donné que $\mathcal{A}$ est δ-local, pour tout triplet (u, a, v) dans $X \times A \times X$, si $\theta(u, a, v)$ est défini, il l'est de façon unique.

Pour tout vecteur pu dans X, pu est initial si p est dans $\underline{I}$ et si $(\varepsilon, \vdash, u)$ est dans $\underline{\eta}$; dans ce cas, on pose $\mathcal{I}(pu) = I(p) + \eta(\varepsilon, \vdash, u)$. De la même façon, tout vecteur up dans X est final si p est dans $\underline{T}$ et si $(u, \dashv, \varepsilon)$ est dans $\underline{\eta}$; dans ce cas, on pose $\mathcal{T}(up) = \eta(u, \dashv, \varepsilon) + T(p)$.

EXEMPLE 2.5 Par exemple, avec les slices du $\mathbb{K}$-automate $\mathcal{B}_1$,

$$\begin{aligned}
\theta\begin{pmatrix} p & & q_- \\ s_+, & a, & r \\ q_+ & & p \end{pmatrix} &= E(p, a, 1, q_-) + \eta\begin{pmatrix} s_+ \\ q_+ \end{pmatrix}, a, \begin{matrix} r \\ p \end{matrix}\Big)\\
&= E(p, a, 1, q_-) + E(r, a, -1, s_+) + \theta(q_+, a, p)\\
&= E(p, a, 1, q_-) + E(r, a, -1, s_+) + E(q_+, a, 1, p).
\end{aligned} \tag{2.4}$$

Le vecteur $\begin{bmatrix} p \\ s_+ \\ q_+ \end{bmatrix}$ est initial et

$$\mathcal{I}\begin{pmatrix} p \\ s_+ \\ q_+ \end{pmatrix} = I(p) + E(s_+, \vdash, 1, q_+). \tag{2.5}$$

Définition 2.8 Avec les définitions précédentes, l'*automate des slices* du $\mathbb{K}$-automate bidirectionnel $\mathcal{A} = (Q, A, E, I, T)$ est le $\mathbb{K}$-automate unidirectionnel infini $\mathcal{C} = (X, A, \theta, \mathcal{I}, \mathcal{T})$.

Proposition 2.5 *Soit $\mathbb{K}$ un semi-anneau* commutatif *et soit $\mathcal{A}$ un $\mathbb{K}$-automate bidirectionnel δ-local. Il y a une bijection φ entre les calculs dans $\mathcal{A}$ et les calculs dans l'automate des slices de $\mathcal{A}$ telle que, pour tout calcul ρ de $\mathcal{A}$,*
– ρ et $\varphi(\rho)$ ont la même étiquette et le même poids ;
– la signature de ρ est la séquence d'états qui composent $\varphi(\rho)$.

Démonstration. Soit π un calcul dans $\mathcal{C}$ étiqueté par w. Soit $\pi^{(k)}$ le préfixe de longueur k de π et soit $(v^{(0)}, \dots, v^{(k)})$ la séquence d'états de $\pi^{(k)}$. Nous montrons par récurrence sur la valeur de k qu'à partir de $\pi^{(k)}$, il n'y a qu'une seule façon de récupérer la restriction d'un calcul de $\mathcal{A}$ sur w aux k premières lettres. De plus, le poids de $\pi^{(k)}$ (incluant le poids initial) est égal au poids de cette restriction. Si $k = 0$, $\pi^{(k)}$ est réduit à un slice initial. D'après l'équation 2.3, la restriction des calculs dans l'automate bidirectionnel est définie de façon unique : $v_1^{(0)}$ est initial avec un poids $I(v_1^{(0)})$, et pour tout r dans $[1; (v^{(0)} - 1)/2]$, il y a une transition $v_{2r}^{(0)} \xrightarrow{\vdash,\rightarrow|h_r} v_{2r+1}^{(0)}$; le poids de cette restriction est en fait le poids initial de $v^{(0)}$ dans $\mathcal{C}$. Si $k > 0$, nous considérons la restriction de $k - 1$; cette restriction correspond à une union disjointe de parties des calculs et il n'y a qu'une seule façon de les connecter aux états du slice $v^{(k)}$ (comme $\mathcal{A}$ est δ-normalisé). Le poids de la transition entre $v^{(k-1)}$ et $v^{(k)}$ correspond exactement à la somme des poids des nouvelles transitions impliquées dans la restriction.

Finalement, à partir de la restriction de taille $|w|$, si nous considérons $v^{(|w|)}$ comme un état final de $\mathcal{C}$, par un argument similaire à celui pour l'état initial, nous arrivons à la conclusion qu'il y a un et un seul calcul dans $\mathcal{A}$ qui correspond à un calcul donné dans $\mathcal{C}$. □

Corollaire 2.6 *Soit* $\mathbb{K}$ *un semi-anneau* commutatif *et soit* $\mathcal{A}$ *un* $\mathbb{K}$*-automate bidirectionnel* δ*-local. Il existe un* $\mathbb{K}$*-automate unidirectionnel (fini)* $\mathcal{B}$ *tel qu'il existe une bijection* φ *entre les calculs réduits dans* $\mathcal{A}$ *et les calculs dans* $\mathcal{B}$ *telle que, pour tout calcul réduit* ρ *dans* $\mathcal{A}$*,*
– ρ *et* $\varphi(\rho)$ *ont la même étiquette et le même poids ;*
– *la signature de* ρ *est la séquence d'états de* $\varphi(\rho)$*.*

Démonstration. Soit $\mathcal{A} = (Q, A, E, I, T)$ un $\mathbb{K}$-automate bidirectionnel. Nous considérons des vecteurs formés d'éléments de Q tels qu'il n'y a pas d'état dans Q qui n'apparaisse deux fois à des positions avec la même parité. Pour tout k dans $\mathbb{N}$, nous posons

$$\begin{aligned} V_k =& \{v \in Q^{2k+1} \mid v_i = v_j \Rightarrow i \neq j \mod 2\} \\ =& \{v \in Q^{2k+1} \mid \forall p \in Q, \forall s \in [0;1], |\{i \mid v_i = p \text{ et } i = s \mod 2\}| \leqslant 1\} \end{aligned} \tag{2.6}$$

Pour tout k plus grand que $|Q| - 1$, V_k est vide. Soit $V = \bigcup_k V_k$; nous définissons le $\mathbb{K}$-automate unidirectionnel avec V pour ensemble d'états. Il est clair qu'un calcul est réduit si et seulement si tous les slices du calcul sont dans V.

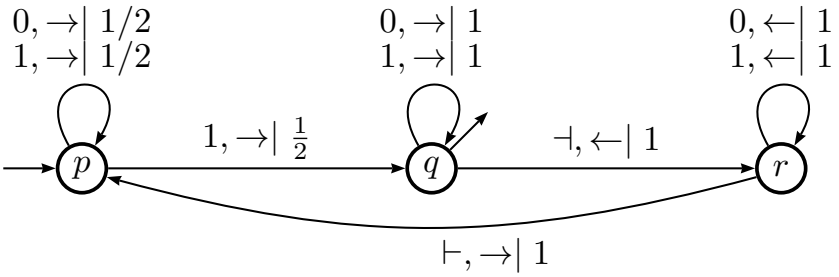


FIGURE 2.4 – Le $\mathbb{Q}$-automate bidirectionnel $\mathcal{B}_2$.

D'après la Proposition 5, la restriction de l'automate des slices à V fournit un automate fini qui respecte le corollaire. □

2.5 Calculs réduits et automates unidirectionnels

Pour le cas des automates sans multiplicité (ou Booléens), les automates bidirectionnels décrivent exactement les mêmes langages que les automates unidirectionnels [46, 43]. Ce n'est pas toujours le cas avec les automates à multiplicité. Par exemple, soit $\mathbb{K}$ le semi-anneau des langages sur l'alphabet $\{x, y\}$. Il n'est pas difficile de construire un $\mathbb{K}$-automate bidirectionnel déterministe sur l'alphabet $\{a\}$ tel que l'image de a^n est $x^n y^n$ (un premier passage gauche-droite produit un x pour chaque a, ensuite l'automate revient au début du mot et un second passage gauche droite produit un y pour chaque a). Cette fonction n'est clairement pas rationnelle et ne peut pas être réalisée par un $\mathbb{K}$-automate unidirectionnel.

Dans [2], une proposition plus générale est prouvée : si le semi-anneau est commutatif et si le nombre de slices d'un automate bidirectionnel $\mathcal{A}$ est fini, il existe un automate unidirectionnel équivalent.

EXEMPLE 2.6 Le $\mathbb{Q}$-automate bidirectionnel $\mathcal{B}_2$ calcule pour chaque mot w sur l'alphabet $\{0, 1\}$ la valeur $\frac{x}{1-x}$, où $x = \sum_{i\in[1;|w|]} \frac{w_i}{2^i}$. Bien que $\mathbb{Q}$ soit commutatif, il n'existe pas de $\mathbb{Q}$-automate unidirectionnel équivalent à cet automate ; $s_2 = |\mathcal{B}_2|$ n'est pas une série rationnelle.

EXEMPLE 2.7 Soit $\mathcal{A}'_1$ l'automate unidirectionnel de distance obtenu à partir de $\mathcal{B}_1$ (Figure 2.5). Il a été prouvé dans [26] que cette fonction ne peut pas être réalisée par un automate de distance unidirectionnel déterministe et même par une union finie de tels automates.

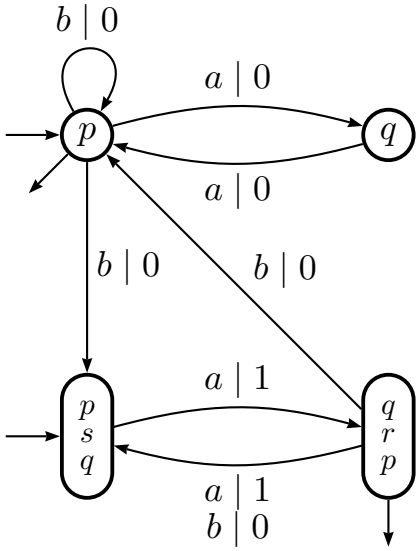


FIGURE 2.5 – L'automate de distance unidirectionnel non-ambigu $\mathcal{A}'_1$.

3 Non-ambiguité et déterminisme

Le résultat de [2] peut être exprimé plus précisément pour les automates non-ambigus.

Proposition 3.1 *Soit $\mathbb{K}$ un semi-anneau commutatif. Tout $\mathbb{K}$-automate bidirectionnel non-ambigu est équivalent à un $\mathbb{K}$-automate unidirectionnel non-ambigu.*

En fait, tout les calculs dans un tel automate sont réduits et donc il n'y a qu'un nombre fini de slices possibles ; ainsi l'automate des slices est fini. De plus, pour tout mot accepté, il n'y a qu'un seul calcul acceptant, et donc une seule signature possible qui correspond à un unique calcul acceptant dans l'automate des slices. C'est pourquoi l'automate des slices est non-ambigu.

3.1 Des $\mathbb{K}$-automates unidirectionnels non-ambigus vers les $\mathbb{K}$-automates bidirectionnels déterministes

Définition 3.1 Un automate bidirectionnel est déterministe si
i) il possède au plus un état initial ;
ii) pour tout état p et toute lettre a, il y a au plus une transition sortant de p étiquetée par a ;
iii) pour tout état final p, il n'y a aucune transition sortant de p étiquetée par $\dashv$.

La dernière condition implique que si un état final est atteint à la fin d'un mot, il n'y a pas de choix non-déterministe entre finir le calcul et lire le marqueur de fin pour continuer la lecture.

Théorème 3.2 *Soit $\mathbb{K}$ semi-anneau. Tout $\mathbb{K}$-automate unidirectionnel non-ambigu est équivalent à un $\mathbb{K}$-automate bidirectionnel déterministe.*

La preuve de ce théorème est largement inspirée de [15].

Démonstration. Soit $\mathcal{A} = (I, E, T)$ un $\mathbb{K}$-automate unidirectionnel non-ambigu avec Q pour ensemble d'états.

Nous considérons l'application μ de A dans l'ensemble des matrices booléennes de taille $Q \times Q$ définie par :

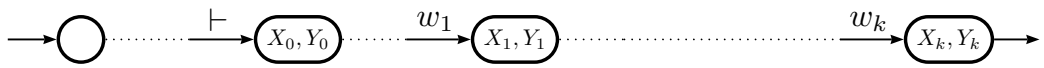
$$\forall a \in A,\ \forall p, q \in Q,\ \mu(a)_{p,q} = 1 \Longleftrightarrow (p, a, q) \in \underline{E}. \tag{3.1}$$

Le monoïde engendré par $\{\mu(a) \mid a \in A\}$ est le *monoïde de transition* M de $\mathcal{A}$. L'application μ s'étend naturellement en un morphisme de monoïdes de A^* dans M. Tout sous-ensemble de Q peut être vu comme un vecteur dans $\mathbb{B}^Q$; pour tout mot w, $\underline{I}\mu(w)$ est l'ensemble des états accessibles à partir d'un état initial en suivant un calcul étiqueté par w et inversement, $\mu(w)\underline{T}$ est l'ensemble des états à partir desquels un état final peut être atteint en suivant un calcul étiqueté par w.

Comme $\mathcal{A}$ est non-ambigu, pour toute paire de mots (u, v), $\underline{I}\mu(u) \cap \mu(v)\underline{T}$ a au plus un élément (sinon il existerait une infinité de calculs acceptant uv) ; de la même manière, pour toute lettre, il existe au plus une transition (p, a, q) dans $\mathcal{A}$ avec p dans $\underline{I}\mu(u)$ et q dans $\mu(v)\underline{T}$ (autrement il existerait de nombreux calculs acceptant uav).

Pour tout mot $w = w_1 \dots w_k$, pour tout i dans $[0; k]$, on pose $X_i(w) = \underline{I}\mu(w_1 \dots w_i)$ et $Y_i(w) = \mu(w_{i+1} \dots w_k)\underline{T}$.

On construit un $\mathbb{K}$-automate bidirectionnel déterministe $\mathcal{B}$ équivalent à $\mathcal{A}$. Si w est accepté par $\mathcal{B}$, pour tout i dans $[1; k]$, l'état atteint après la dernière lecture de w_i contient l'information $(X_i(w), Y_i(w))$:



A partir de $(X_{i-1}(w), Y_{i-1}(w))$ et $(X_i(w), Y_i(w))$, la transition étiquetée par w_i dans le calcul étiqueté par w peut être déduite : il s'agit de l'unique transition (p, w_i, q) avec p dans $X_{i-1}(w)$ et q dans $Y_i(w)$. De même (X_0, Y_0) détermine le poids initial et (X_k, Y_k) détermine le poids final.

L'ensemble X_i peut être aisément déduit à partir de X_{i-1} : $X_i = X_{i-1}\mu(w_i)$. Le calcul de Y_i à partir de Y_{i-1} est plus subtile.

Soient x et y deux éléments de M. S'il existe z dans M tel que $x = zy$, on dit que $x \leqslant_L y$; cette relation est un pré-ordre. S'il existe également t tel que $tx = y$, on dit que x et y sont L-équivalents.

Soit u un facteur de w qui commence en w_{i+1}. Il est clair que $\mu(w_i u) \leqslant_L \mu(u)$. Si $\mu(w_i u)$ et $\mu(u)$ sont L-équivalents, il existe y dans M tel que $y\mu(w_i u) = \mu(u)$. Dans ce cas, il est également clair que $y\mu(w_i \dots w_k) = \mu(w_{i+1} \dots w_k)$ et donc, $Y_i = yY_{i-1}$. L'automate bidirectionnel peut effectuer ces calculs, puisqu'ils reposent sur le monoïde de transition, qui est fini. L'automate calcul de façon incrémentale pour tout j dans $[i; k]$ la valeur de $\mu(w_{i+1} \dots w_j)$ jusqu'à ce que $\mu(w_i \dots w_j) <_L \mu(w_{i+1} \dots w_j)$ et $\mu(w_i \dots w_{j+1}) \equiv_L \mu(w_{i+1} \dots w_{j+1})$. Si on arrive à $j = k$, alors $Y_i = \mu(w_{i+1} \dots w_j)T$, sinon, $Y_i = yY_{i-1}$ où y est tel que $y\mu(w_i \dots w_{j+1}) = \mu(w_{i+1} \dots w_{j+1})$.

Une fois que Y_i est calculé, l'automate doit retourner à la position i. L'automate est dans une certaine position j telle que $\mu(w_i \dots w_j) <_L \mu(w_{i+1} \dots w_j)$; *a fortiori*, pour tout r dans $[i+1; j]$, $\mu(w_i \dots w_j) <_L \mu(w_r \dots w_j)$. L'automate saute donc toute position plus petite que j jusqu'à ce qu'il arrive à un certain point s tel que $\mu(w_i \dots w_j) = \mu(w_s \dots w_j)$. A ce moment, on a $s = i$.

L'ensemble des états de $\mathcal{B}$ est l'union de cinq types d'états :

- $Q_0 = \{i\}$ est l'état initial ; dans cet état, l'automate lit l'entrée de gauche à droite jusqu'à ce qu'il atteigne le marqueur de fin $\dashv$. Ensuite il va dans un état de Q_1.

- $Q_1 \subseteq \mathcal{P}$; dans cet état, l'automate lit l'entrée de droite à gauche ; après avoir lu le suffixe v, l'état correspond à $\mu(v)\underline{T}$. Quand le marqueur de début $\vdash$ est atteint, l'automate va dans un état de Q_2.

- $Q_2 \subseteq \mathcal{P}^2$; ces états correspondent aux paires (X_i, Y_i) ; les transitions arrivant sur ces états correspondent aux transitions de l'automate unidirectionnel ; leur poids est le même. De même, un état dans Q_2 peut être final s'il appartient à $\mathcal{P} \times \{T\}$. Quand l'automate est dans un de ces états, soit il s'arrête, soit il commence à traiter une nouvelle lettre ; cette lettre est lue et stockée dans l'état suivant qui appartient à Q_3.

- $Q_3 \subseteq A \times M \times \mathcal{P}^2$; l'automate reste dans les états Q_3 aussi longtemps que nécessaire pour calculer Y_i à partir de Y_{i-1}. Il stock la lettre courante a ainsi que l'image dans le monoïde de transition du facteur u qui suit a et qui finit à la position actuelle. Si l'état stock $\mu(u)$ qui est plus grand pour l'ordre L que $\mu(au)$ et si la lettre lue b est telle que $\mu(aub)$ et $\mu(ub)$ sont L-équivalents, alors il existe y tel que $y\mu(aub) = \mu(ub)$; donc $Y_i = yY_{i-1}$, l'automate stock $\mu(au)$ et passe à un état dans Q_4.

- $Q_4 \subseteq M^2 \times \mathcal{P}^2$; l'automate reste dans un état de Q_4 pendant qu'il lit le mot u de la droite vers la gauche ; il stock l'image du suffixe v de u qui est lu ; on a $\mu(v) >_L \mu(au)$ jusqu'à ce que $v = u$; lorsque $v = u$, l'automate lit la lettre a et vérifie que $\mu(av) = \mu(au)$; à ce moment, il connait X_i et Y_{i+1}, et donc, il peut fournir en sortie le poids de l'unique transition compatible avec a, X_i et Y_{i+1}, et passe à l'état $(X_{i+1} = X_i\mu(a), Y_{i+1})$.

$$
\begin{aligned}
F = \{&i \xrightarrow{a,\rightarrow} i \in Q_0 \mid a \in A\}\\
&\cup \{i \xrightarrow{\dashv,\leftarrow} \underline{T} \in Q_1\}\\
&\cup \{Y \xrightarrow{a,\leftarrow} \mu(a)Y \in Q_1 \mid Y \in Q_1, a \in A\}\\
&\cup \{Y \xrightarrow{\vdash,\rightarrow|I_k} (I, Y) \in Q_2 \mid Y \in Q_1, k \in I \cap Y\}\\
&\cup \{(X, Y) \xrightarrow{a,\rightarrow} (a, 1_{\mathbb{K}}, X, Y) \in Q_3 \mid (X, Y) \in Q_2, a \in A\}\\
&\cup \{(a, x, X, Y) \xrightarrow{b,\rightarrow} (a, x\mu(b), X, Y) \in Q_3 \mid (a, x, X, Y) \in Q_3,\\
&\qquad\qquad b \in A, \mu(a)x\mu(b) <_L x\mu(b)\}\\
&\cup \{(a, x, X, Y) \xrightarrow{b,\leftarrow} (\mu(a)x, 1, X, yY) \in Q_4 \mid (a, x, X, Y) \in Q_3,\\
&\qquad\qquad b \in A, y \in M, y\mu(a)x\mu(b) = x\mu(b)\}\\
&\cup \{(a, x, X, Y) \xrightarrow{\dashv,\leftarrow} (\mu(a)x, 1, X, T) \in Q_4 \mid (a, x, X, Y) \in Q_3\}\\
&\cup \{(x, y, X, Y) \xrightarrow{a,\leftarrow} (x, \mu(a)y, X, Y) \in Q_4 \mid (x, y, X, Y) \in Q_4,\\
&\qquad\qquad a \in A, x <_L \mu(a)y\}\\
&\cup \{(x, y, X, Y) \xrightarrow{a,\rightarrow|k} (X\mu(a), Y) \in Q_2 \mid (x, y, X, Y) \in Q_4,\\
&\qquad a \in A, x = \mu(a)y, \exists (p, q) \in X \times Y, \exists p \xrightarrow{a|k} q \in \mathcal{A}\}.
\end{aligned}
\tag{3.2}
$$

□

Pour tout X dans $\mathcal{P}$, si l'état $(X, \underline{T})$ appartient à Q_2, $(X, \underline{T})$ est final avec un poids T_p, où p est l'unique état dans $X \cap \underline{T}$.

EXEMPLE 3.1 Soit $\mathcal{A}'_1$ le $\mathbb{K}$-automate unidirectionnel non-ambigu de la Figure 2.5. Nous numérotons les états de ce $\mathbb{K}$-automate : $[p] = 1$, $[q] = 2$, $[p, s, q] = 3$ et $[q, r, p] = 4$. Le monoïde de transition est généré par les matrices suivantes :

$$
\alpha = \mu(a) = \begin{bmatrix} 0&1&0&0\\ 1&0&0&0\\ 0&0&0&1\\ 0&0&1&0 \end{bmatrix}, \qquad \beta = \mu(b) = \begin{bmatrix} 1&0&1&0\\ 0&0&0&0\\ 0&0&0&0\\ 1&0&1&0 \end{bmatrix}. \tag{3.3}
$$

On a les identités suivantes : $\alpha^2 = 1$, $\beta^2 = \beta$, $\beta\alpha\beta = \beta$. On a donc $1 \equiv_L \alpha$, $\alpha\beta \equiv_L \beta$ et $\alpha\beta\alpha \equiv_L \beta\alpha$, tandis que $\beta <_L 1$ et $\beta\alpha <_L 1$. A noter que β et $\beta\alpha$ ne sont pas comparables. Nous pouvons dérouler la preuve du Théorème 2 pour calculer le $\mathbb{K}$-automate bidirectionnel déterministe équivalent $\mathcal{D}_1$ de la Figure 3.1.

Corollaire 3.3 *Soit $\mathbb{K}$ un semi-anneau commutatif. Tout $\mathbb{K}$-automate bidirectionnel non-ambigu est équivalent à un $\mathbb{K}$-automate bidirectionnel déterministe.*

REMARQUE 3.1 cette conversion peut conduire à une explosion combinatoire. Par exemple, le $\mathbb{K}$-automate bidirectionnel déterministe $\mathcal{D}_1$ de la figure 3.1 possède 27 états dans sa partie complète.

Une borne supérieure du nombre d'états peut être calculée. Soit n le nombre d'états du $\mathbb{K}$-automate unidirectionnel non-ambigu.

- Q_0 possède un seul état ;
- Q_1 possède au plus $2^n - 1$ états ;
- Q_2 est construit avec des paires de sous-ensembles de Q qui ont un élément en commun, donc Q_2 possède au plus $n3^{n-1}$ états ;
- Q_3 est construit avec des paires de Q_2 auxquelles on a ajouté une lettre et un élément du monoïde de transition (qui peut avoir jusqu'à 2^{n^2} éléments) ; donc Q_3 possède au plus $|A|n3^{n-1}2^{n^2}$ états ;
- Q_4 est construit avec deux sous-ensembles (non vides) de Q et deux éléments du monoïde de transition ; sa taille est bornée par 2^{2n+2n^2}.

4 Automates tropicaux bidirectionnels

Les automates tropicaux sont des automates à multiplicité dans des semi-anneaux de type *min-plus*[2]. Pour tout sous-monoïde additif $\mathbb{K}$ de $\mathbb{R}$, on peut définir le semi-anneau min-plus $\mathcal{K} = (\mathbb{K} \cup \{\infty\}, \min, +)$. Par exemple, à partir de $\mathbb{N}$, $\mathbb{Z}$ et $\mathbb{R}_+$, on peut définir respectivement les semi-anneaux $\mathcal{N}$,$\mathcal{Z}$ et $\mathcal{R}_+$. A noter que dans un $\mathcal{K}$-automate, seuls les poids dans $\mathbb{K}$ apparaissent (∞ est un moyen algébrique de spécifier l'absence de transition). Comme nous l'avons déjà mentionné, nous appelons *automates de distance* tout automate min-plus avec des poids non négatifs. Ainsi, les $\mathcal{N}$-automates et les $\mathcal{R}_+$-automates sont tous des automates de distance.

2. On appelle également semi-anneaux tropicaux des semi-anneaux qui possèdent des propriétés similaires aux semi-anneaux min-plus comme par exemple $(\mathbb{N} \cup \{-\infty, \infty\}, max, +)$

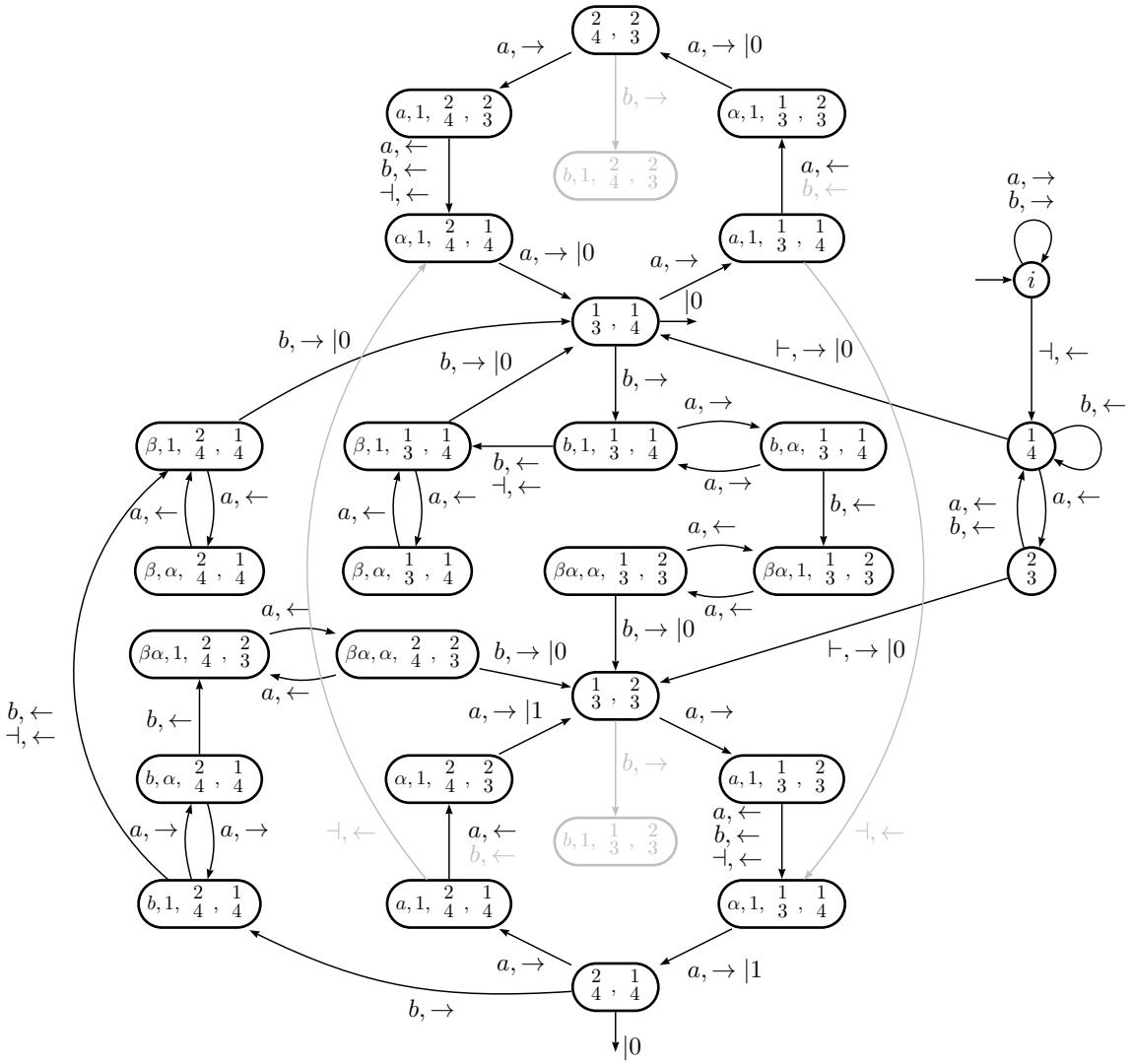


FIGURE 3.1 – L'automate de distance bidirectionnel déterministe $\mathcal{D}_1$. Les transitions et les états en grisé ne sont pas accessibles. Pour plus de clarté, les transitions sortant d'états non accessibles ne sont pas dessinées. Tout doublet correspond à l'ensemble des éléments non nuls d'un vecteur booléen de taille 4. Les poids sont écrits uniquement sur les transitions lorsqu'ils proviennent du poids d'une transition (ou d'un poids initial/final) de $\mathcal{A}'_1$.

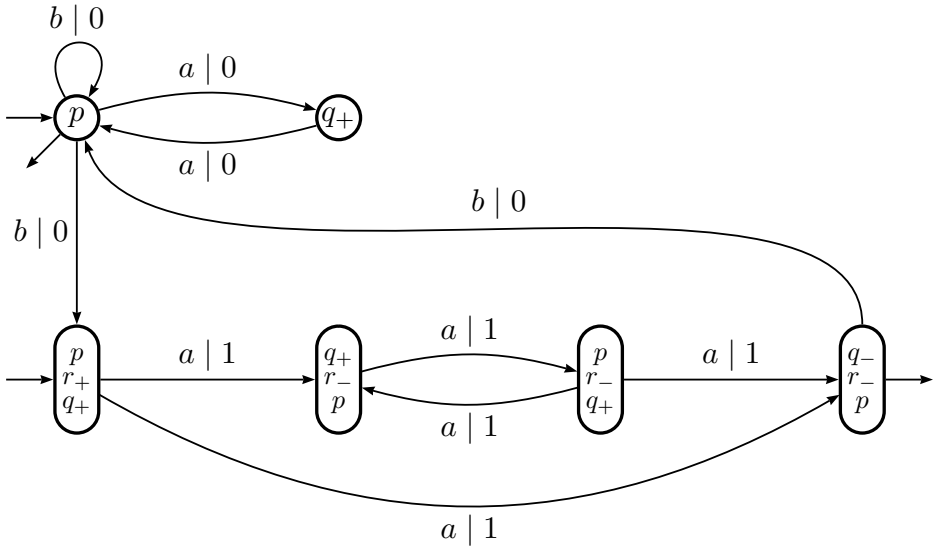


FIGURE 4.1 – Un $\mathcal{N}$-automate unidirectionnel équivalent à $\mathcal{B}_1$.

Dans la pratique, on utilise le semi-anneau $\mathbb{P} = ([0;1], max, \cdot)$ (le poids d'un calcul correspond au produit des probabilités du calcul et le poids d'un mot est le poids du calcul sur ce mot avec la plus grande probabilité). L'application $x \to -logx$ est en fait un isomorphisme de $\mathbb{P}$ dans $\mathcal{R}_+$. Tout résultat sur les automates des distances est donc valable pour les $\mathbb{P}$-automates également.

4.1 Automates de distance bidirectionnel

D'après le Lemme 2 pour simuler un automate de distance bidirectionnel par un automate unidirectionnel, nous avons besoin de ne simuler que les calculs réduits.

En fait, si un calcul dans un automate bidirectionnel contient un circuit immobile, la signature de ce calcul contient un vecteur dans lequel deux entrées avec un index de même parité sont égales. Nous avons déjà prouvé plus tôt que la restriction de l'automate des slices d'un automate $\mathcal{A}$ aux états étiquetés par un vecteur qui ne contient pas ce genre d'entrée résulte en un automate unidirectionnel dans lequel tout calcul correspond à un calcul réduit dans $\mathcal{A}$ avec le même poids et que tout calcul réduit dans $\mathcal{A}$ possède un représentant dans l'automate unidirectionnel fini.

EXEMPLE 4.1 A partir de la δ-normalisation de $\mathcal{B}_1$, on peut construire un $\mathcal{N}$-automate unidirectionnel équivalent (représenté sur la Figure 4.1).

Et enfin, d'après le Lemme 2, on arrive à la proposition suivante :

Proposition 4.1 *Tout automate de distance bidirectionnel est équivalent à un automate de distance unidirectionnel.*

4.2 Automates Min-plus bidirectionnels

A partir de maintenant, nous traitons des automates bidirectionnels sur les semi-anneaux min-plus basés sur des sous-monoïdes de $\mathbb{R}$ qui contiennent des éléments négatifs. Dans ce cas, un mot peut étiqueter une infinité de calculs avec des poids de plus en plus petits.
Nous dirons qu'un automate min-plus bidirectionnel est *valide* si le poids de tout mot accepté est fini. Nous traitons le problème visant à décider si un automate min-plus bidirectionnel est valide ou pas.

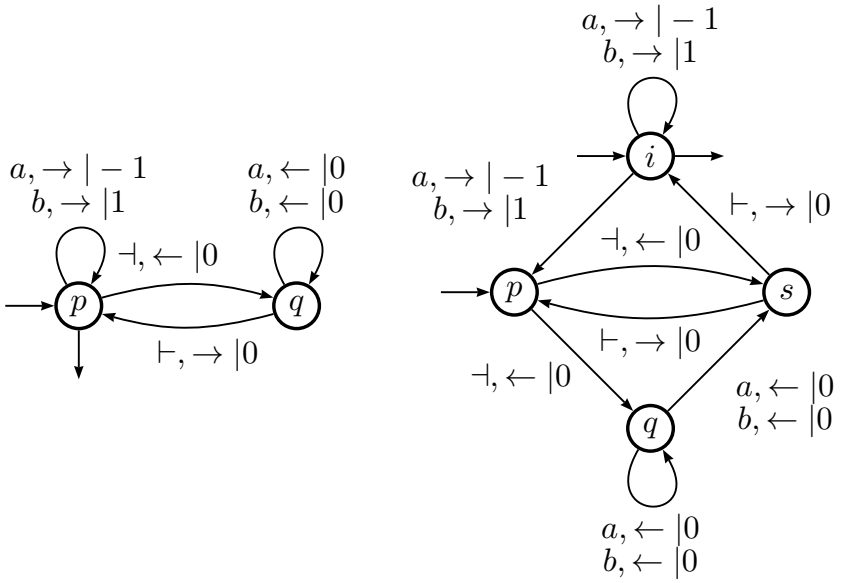


FIGURE 4.2 – Les $\mathcal{Z}$-automates bidirectionnels $\mathcal{B}_4$ et $\mathcal{B}'_4$.

EXEMPLE 4.2 Soit $\mathcal{B}_4$ le $\mathcal{Z}$-automate bidirectionnel de la Figure 4.2 (gauche). A chaque fois que cet automate lit un mot de gauche à droite, il calcul la différence entre le nombre de 'b' et le nombre de 'a'. Comme pour chaque mot accepté, il peut exister un nombre non borné de lecture gauche-droite, s'il y a plus de 'a' que de 'b', le poids des calculs n'est pas borné inférieurement. Donc, le comportement de cet automate est défini uniquement pour les mots dans lesquels le nombre de 'a' est au plus égal au nombre de 'b'.

L'automate $\mathcal{B}'_4$ de la Figure 4.2 (droite) correspond à la δ-normalisation de $\mathcal{B}_4$.

Cet exemple prouve le fait suivant.

Proposition 4.2 *Il existe des automates min-plus bidirectionnels tels que le langage de mots acceptés avec un poids fini n'est pas rationnel.*

Théorème 4.3 *Il est décidable si un automate min-plus bidirectionnel est valide ou pas.*

Pour prouver ce théorème, nous avons besoin de considérer une autre restriction de l'automate des slices. A la différence du cas des automates de distance où nous voulions que les circuits immobiles n'apparaissent pas du tout, nous voulons ici détecter quand ces circuits immobiles apparaissent, mais nous voulons manipuler un automate fini. C'est pourquoi nous autorisons chaque circuit immobile à apparaître au plus une fois.

Pour cette raison, nous considérons les slices qui appartiennent à $W = \bigcup_k W_k$ avec W_k défini pour tout k dans $\mathbb{N}$ comme suit :

$$W_k = \{v \in Q^{2k+1} | \forall p \in Q, \forall s \in [0;1], |\{i \mid v_i = p \text{ et } i \mod 2 = s\}| \leqslant 2\}. \tag{4.1}$$

Nous considérons la restriction de l'automate des slices à W.

Proposition 4.4 *Soit $\mathcal{A}$ un $\mathbb{K}$-automate bidirectionnel et soit $\mathcal{C}$ la restriction de l'automate des slices de $\mathcal{A}$ à W. Si $\mathcal{A}$ accepte un calcul qui contient un circuit immobile avec un poids négatif, alors il existe un calcul dans $\mathcal{A}$ qui contient un circuit immobile et qui a une image dans $\mathcal{C}$.*

Démonstration. Soit ρ un calcul dans $\mathcal{A}$ qui contient un circuit immobile avec un poids négatif et une taille minimal. Si ρ n'a pas d'image dans $\mathcal{C}$, il existe un état p qui apparaît (au moins) trois fois dans un slice v_1 de ρ (soient p_1, p_2 et p_3 ces trois occurrences), et un état q qui apparaît (au moins) deux fois dans un slice v_2 de ρ (soient q_1 et q_2 ces deux occurrences) : q correspond à la fin du circuit immobile de poids négatif. Donc p correspond à la fin de deux circuits immobiles consécutifs.
Différents cas apparaissent. Si un des deux circuits immobiles consécutifs possède un poids négatif, alors l'autre peut être supprimé pour simplifier le calcul.
De la même façon, si un des deux circuits immobiles consécutifs n'a pas de partie commune avec le circuit immobile de poids négatif, il peut être supprimé.

Le seul cas restant est celui dans lequel le calcul ρ peut être décomposé de la façon suivante :

$$\rightarrow i \xrightarrow{w_1|k_1} p_1 \xrightarrow{w_2|k_2} q_1 \xrightarrow{w_3|k_3} p_2 \xrightarrow{w_4|k_4} q_2 \xrightarrow{w_5|k_5} p_3 \xrightarrow{w_6|k_6} t \rightarrow . \tag{4.2}$$

Dans ce cas, on a $k_3 + k_4 < 0$ et le calcul plus court

$$\rightarrow i \xrightarrow{w_1|k_1} p_1 = p_2 \xrightarrow{w_4|k_4} q_2 = q_1 \xrightarrow{w_3|k_3} p_2 = p_3 \xrightarrow{w_6|k_6} t \rightarrow \tag{4.3}$$

contient un circuit immobile de poids négatif. □

Dans l'automate $\mathcal{C}$, tout calcul qui contient un état dans $W \setminus V$ contient un circuit immobile. Le problème est de détecter si un tel circuit immobile a un poids négatif. La solution consiste à comparer le poids de ce calcul avec le poids du calcul sans le circuit immobile. Dans cet optique, nous définissons un automate qui est une sorte de *carré* de l'automate $\mathcal{C}$ (*cf.* [4]) : il compare les calculs de $\mathcal{C}$ qui diffèrent par des circuits immobiles.

Nous allons tout d'abord considérer l'ensemble suivant : $X = \{(x, y, z) \in \mathcal{P}(Q)^3 \mid xz, xyz \in W\}$. Un élément (x, y, z) dans X est dit *spécial* si $y_1 = z_1$. A partir de la fonction θ, nous définissons un fonction (partielle) $\tilde{\theta} : X \times A \times X \longrightarrow \mathbb{K}$ de la façon suivante

$$\tilde{\theta}((x, y, z), a, (t, u, v)) = \theta(xyz, a, tuv) - \theta(xz, a, tv), \tag{4.4}$$

pour tout triplet $((x, y, z), a, (t, u, v))$ qui respecte les trois conditions suivantes :

$$\begin{aligned} (x, a, t), (y, a, u) \text{ et } (z, a, v) &\in \underline{\theta}; \\ (xy_1, a, t), (y_1^{-1}yz_1, a, u) \text{ et } (z_1^{-1}z, a, v) &\in \underline{\theta}, y_1 = z_1 \text{ et } \delta_{\mathbf{I}}(y_1) = -1; \\ (x, a, tu_1), (y, a, u_1^{-1}uv_1) \text{ et } (z, a, v_1^{-1}v) &\in \underline{\theta}, u_1 = v_1 \text{ et } \delta_{\mathbf{I}}(u_1) = 1. \end{aligned} \tag{4.5}$$

Soit $X_0 = \{(x, y, z) \in X \mid y = 1\}$. Nous définissons sur X la relation $(x, y, z) \equiv (x', y', z')$ si et seulement si $y = y' = 1$ et $xz = x'z'$. Dans le quotient $\tilde{X}$ de X par $\equiv$, tout élément qui n'est pas dans X_0 est l'unique élément de sa classe, tandis que le quotient de X_0 est isomorphe à W. De plus, cette équivalence est cohérente avec la définition de $\tilde{\theta}$. Soit $\mathcal{P} = (\tilde{X}, A, \tilde{\theta}, J, U)$ un automate unidirectionnel défini de la façon suivante : la fonction de transition est $\tilde{\theta}$; toute transition qui correspond à l'une des deux dernières lignes de (4.5) est appelée transition spéciale. Nous posons $J(x, y, z) = \theta(0, \vdash, xyz) - \theta(0, \vdash, xz)$ si x est non-vide ou bien si x est vide avec $y_1 = z_1$ (état initial spécial). De la même manière, $U(x, y, z) = \theta(xyz, \dashv, 0) - \theta(xz, \dashv, 0)$ si z est non vide ou bien si z est vide avec $x_1 = y_1$ (état final spécial).

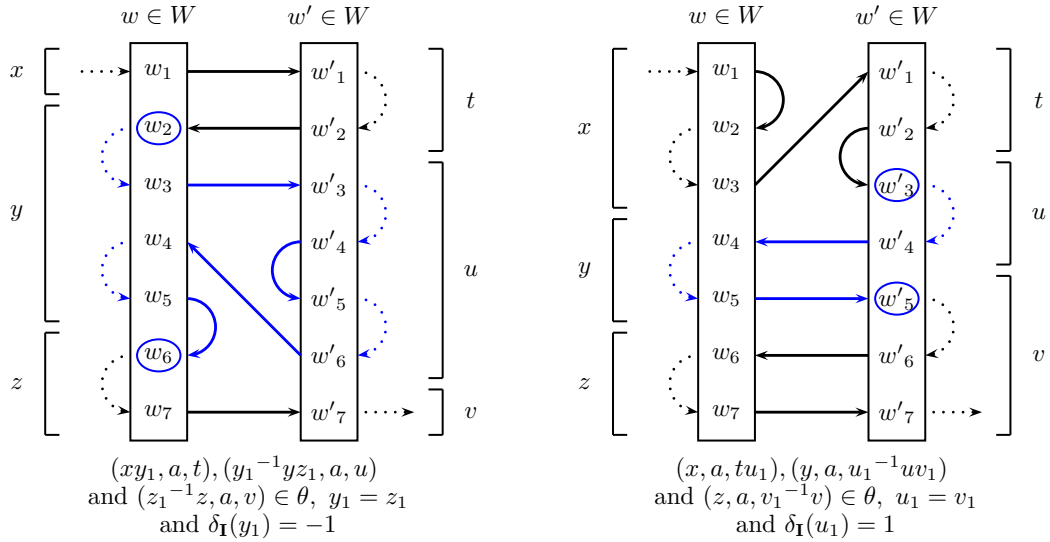


FIGURE 4.3 – Un calcul valide sur le mot $w = w_1 \dots w_k$.

Tout calcul dans l'automate $\mathcal{P}$ qui contient une (et une seule) transition spéciale (ou état spécial initial ou final) correspond à deux calculs dans l'automate des slices. Chaque état du premier calcul est obtenu à partir de tous les états (x, y, z) du calcul dans $\mathcal{P}$ en concaténant x, y et z, tandis que chaque état du second calcul est donné par la concaténation de x et z. Ces deux calculs correspondent à deux calculs dans $\mathcal{A}$, l'un avec un circuit immobile, le second dans lequel le circuit immobile a été supprimé.

EXEMPLE 4.3 A partir de l'automate $\mathcal{B}'_4$ de la Figure 4.2 (droite), nous pouvons construire l'automate $\mathcal{P}_4$ de la Figure 4.4. Les états dans X_0 sont étiquetés par un vecteur, et les autres par trois vecteurs. Les transitions spéciales ainsi que les états initiaux spéciaux sont en rouge (il n'y a pas d'état final spécial). Chaque calcul dans cet automate qui contient une transition spéciale (ou un état initial spécial) correspond à deux calculs dans $\mathcal{B}'_4$, l'un avec un circuit immobile, l'autre sans ce circuit. Une telle paire de calculs dans $\mathcal{B}'_4$ peut correspondre à de nombreux calculs dans $\mathcal{P}_4$, ceci dépendant de l'endroit où l'on coupe dans le calcul contenant le circuit immobile. Par exemple, considérons le calcul $\left(i, \begin{smallmatrix} s \\ i \end{smallmatrix}, \_\right) \xrightarrow{a,-1} \left(\_, \begin{smallmatrix} i \\ q \end{smallmatrix}, i\right) \xrightarrow{a,-1} \left(\_, \begin{smallmatrix} p \\ q \end{smallmatrix}, i\right)$; il correspond au calcul de la Figure 4.5 et à la coupure entre les deux états rouges. Le poids du calcul avec le circuit immobile est -4, sans ce circuit immobile, il est de -2 ; la différence est -2 ce qui correspond bien au poids du calcul dans $\mathcal{P}_4$.

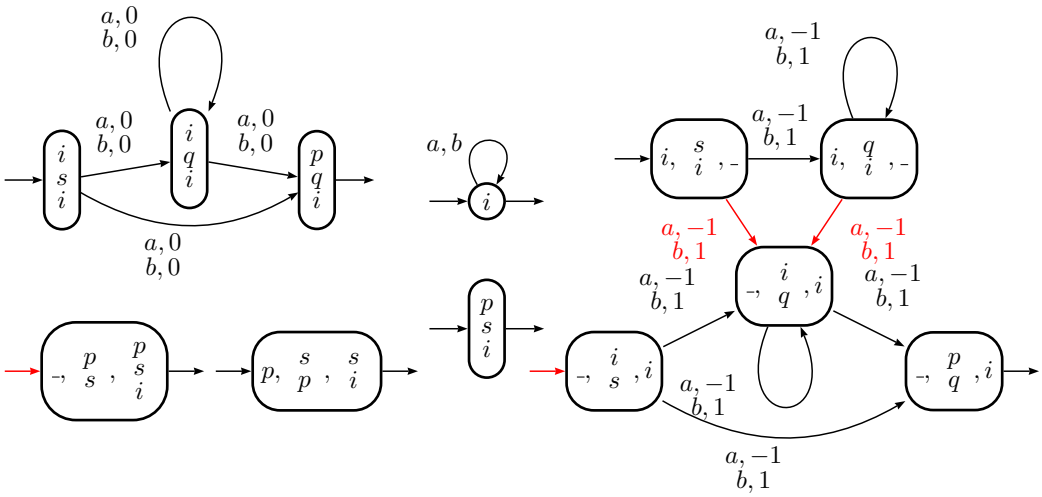


FIGURE 4.4 – L'automate unidirectionnel $\mathcal{P}_4$.

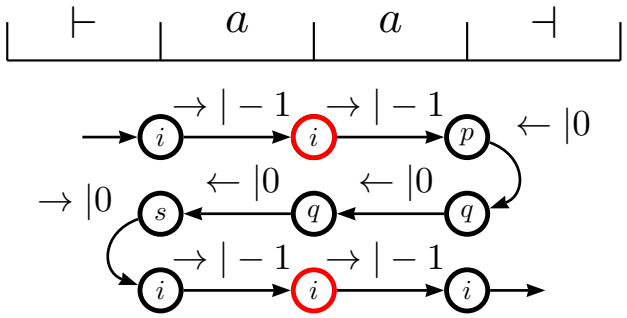


FIGURE 4.5 – Un calcul dans l'automate $\mathcal{B}_4$ contenant un circuit immobile de poids négatif.

Proposition 4.5 *Soit $\mathcal{A}$ un $\mathbb{K}$-automate bidirectionnel et soit $\mathcal{P}$ construit comme décrit au-dessus. Si $\mathcal{A}$ accepte un calcul qui contient un circuit immobile de poids négatif, alors il y a un calcul dans $\mathcal{P}$ qui contient une transition spéciale (ou un état initial/final spécial) de poids négatif.*

Cette propriété peut être vérifiée sur l'automate $\mathcal{P}$ en un temps polynomial (voir [33] pour un exemple), et cela implique le Théorème 3.

Si l'automate bidirectionnel n'est pas valide, il pourrait être intéressant de calculer une description effective du langage sur lequel le comportement est défini. D'après la Proposition 2 ce langage peut ne pas être rationnel ; pire, il est indécidable de savoir s'il est vide.

Théorème 4.6 *Soit $\mathcal{A}$ un $\mathcal{Z}$-automate bidirectionnel. Il est indécidable s'il existe un mot w accepté par $\mathcal{A}$ avec un poids fini.*

Démonstration. D'après [27], il est indécidable, pour un $\mathcal{Z}$-automate unidirectionnel donné $\mathcal{B} = (Q, A, E, I, T)$, s'il existe un mot w dont le poids dans $\mathcal{B}$ n'est pas négatif. Soit r un élément qui n'est pas dans Q et soit $\mathcal{A} = (Q \cup \{r\}, A, F, I, T)$ le $\mathcal{Z}$-automate bidirectionnel défini de la façon suivante :

$$\begin{aligned} F =& \{p \xrightarrow{a,\rightarrow|k} q \mid p \xrightarrow{a|k} q \in E\} \\ =& \{p \xrightarrow{\dashv,\leftarrow|k} r \mid p \in \underline{T}, T(p) = k\} \\ =& \{r \xrightarrow{\vdash,\rightarrow|k} p \mid p \in \underline{I}, I(p) = k\} \\ =& \{r \xrightarrow{a,\leftarrow|0} r \mid a \in A\} \end{aligned} \tag{4.6}$$

Pour tout mot w, tout calcul étiqueté par w dans $\mathcal{A}$ est une séquence de calculs étiquetés par w dans $\mathcal{B}$. Ainsi, le poids d'un mot w dans $\mathcal{A}$ est défini si et seulement s'il n'y a pas de calcul étiqueté par w avec un poids négatif dans $\mathcal{B}$, c'est à dire si le poids de w dans $\mathcal{B}$ n'est pas négatif. □

5 Conclusion

Le principe des automates bidirectionnels n'est pas aussi simple que celui des automates unidirectionnels. Ils sont, par exemple, difficiles à manipuler à l'aide d'outils algébriques comme les matrices de transitions. Dans ce chapitre, nous avons donc développé des outils spécifiques pour étudier ces automates : les circuits immobiles, la δ-normalisation, et les slices, qui ont

conduit à la définition de l'automate des slices d'un $\mathbb{K}$-automate bidirectionnel.

Nous avons utilisé ces outils pour prouver quelques propriétés intéressantes telles que l'équivalence, si $\mathbb{K}$ est un semi-anneau commutatif, entre les $\mathbb{K}$-automates unidirectionnels non-ambigus et les $\mathbb{K}$-automates bidirectionnels non-ambigus, qui sont eux-mêmes équivalent aux $\mathbb{K}$-automates bidirectionnels déterministes.

Nous nous sommes ensuite concentré sur les automates de distance bidirectionnels et les $\bar{\mathcal{Z}}$-automates bidirectionnels. L'équivalence entre les automates de distance bidirectionnels et les automates de distance unidirectionnels était assez simple à montrer mais l'existence de circuits immobiles de poids négatifs a rendu le cas des $\bar{\mathcal{Z}}$-automates bidirectionnels plus compliqué. Au final, nous sommes arrivé à des résultats inattendus comme un exemple qui prouve que le langage des mots dont le poids est défini dans $\bar{\mathcal{Z}}$ peut ne pas être rationnel. Nous avons également prouver qu'il est décidable si le comportement d'un $\bar{\mathcal{Z}}$-automate bidirectionnel est défini ou non mais il est indécidable s'il existe un mot dont le poids est défini dans $\bar{\mathcal{Z}}$.

Les automates bidirectionnels à multiplicité ont été généralisés de différentes façons, par l'ajout de galets (pebbles) par exemple, mais quelques questions restent ouvertes sur ces objets. Tout d'abord, comme pour les automates unidirectionnels avec ε-transitions [36], une étude sur le "sens" de la validité pour ces automates devrait être effectuée.
La déterminisation d'automates bidirectionnels à multiplicité reste largement inconnue. Un premier pas serait de comprendre comment convertir directement un 2-WFA non-ambigu en un 2-WFA déterministe (sans passer par un 1-NFA).
La seconde partie de ce papier introduit quelques questions ouvertes. Malgré le fait qu'il est indécidable de savoir si le domaine d'un automate tropical bidirectionnel est vide, est-il possible de donner une caractérisation de ce domaine ?

Chapitre V

Génération aléatoire

Dans cette dernière partie, nous fournissons une méthode de génération aléatoire, d'abord pour les automates acycliques, accessibles et déterministes, puis nous nous concentrons sur la classe plus restreinte des automates acycliques minimaux. L'ensemble de ce chapitre à été publié une première fois dans [7] en dans une version longue dans [8].
En théorie des langages, les automates acycliques sont exactement les automates qui reconnaissent les langages finis. Pour cette raison, ils jouent un rôle important dans certains domaines d'application spécifiques, tels que le traitement du langage naturel. D'un point de vue algorithmique, ils jouissent souvent de solutions plus efficaces que les automates classiques ; un exemple bien connu est celui de l'algorithme de minimisation linéaire proposé par Revuz pour les automates acycliques déterministes [44]. Ils apparaissent également comme première étape dans certains algorithmes, deux exemples de ce cas sont liés à la construction de Glushkov [12, 13, 14], ainsi que certaines extensions de l'automate d'Aho-Corasick [40].

Dans le développement et l'analyse d'algorithmes, il est très utile d'avoir accès à des générateurs exhaustifs et aléatoires pour les entrées des algorithmes que l'on veut étudier : le générateur exhaustif est utilisé pour analyser le comportement de l'algorithme pour des entrées de petite taille, mais ne peut pas être utilisé pour des entrées trop grandes puisqu'elles seraient trop nombreuses ; le nombre d'entrées de taille n augmente généralement de façon au minimum exponentielle en n. Ces générateurs peuvent être utilisés soit pour tester si une implémentation est conforme et efficace, soit pour aider le chercheur pendant qu'il tente de trouver des résultats théoriques à propos de l'analyse du cas général de l'algorithme.

Un générateur exhaustif pour les automates acycliques déterministes minimaux a été fourni par Almeida, Moreira et Reis [1], quant à nous, nous proposons un algorithme pour générer aléatoirement des automates acycliques,

accessibles et déterministes ainsi que des automates acycliques minimaux, avec une distribution qui est quasiment uniforme, en utilisant des techniques de Markov. L'idée est de partir d'un automate acyclique à n états (ou d'un automate acyclique minimal), et ensuite de réaliser un certain nombre T de mutations de cet automate, une mutation étant un petite transformation locale qui préserve les propriétés requises (déterministe, accessible et acyclique avec le même nombre d'états, ou minimal avec le même nombre d'états). Comme chaque mutation est effectuée en temps $O(n)$, la complexité de notre algorithme est $O(nT)$. Plus T est grand, plus la distribution de la sortie est proche de la distribution uniforme. Pour une certaine distance par rapport à l'uniformité, il est en général difficile de donner une bonne estimation de la valeur de T adéquate ; ceci est directement lié au *temps de mixage* de la chaîne de Markov, qui est généralement un problème difficile [28]. Nous ne traitons pas ce problème, mais les simulations que nous effectuons semblent indiquer qu'un choix de T polynomial en n fourni un générateur aléatoire correct, au moins pour la plupart des applications.

A noter que les autres méthodes génériques pour générer aléatoirement des structures combinatoires de façon uniforme semblent échouer dans notre cas. Par exemple, les méthodes récursives [23] ou les échantillonneurs de Boltzmann [21], qui ont été utilisés pour les automates déterministes [18, 3, 25], reposent sur une bonne description récursive de l'entrée, qui est encore inconnue pour les automates acycliques. A notre connaissance, le seul résultat combinatoire sur les automates acycliques est dû à Liskovets [30], qui a fourni une formule précise du nombre d'automates acycliques, mais qui ne peut être directement translatée en une description récursive utilisable.

Travaux apparentés : comme nous l'avons mentionné, notre algorithme est un complément du générateur exhaustif de Almeida, Moreira et Reis [1] pour tester des conjectures et des algorithmes basés sur des automates acycliques déterministes ou des automates acycliques minimaux. L'idée d'utiliser des chaînes de Markov pour ce type d'objets a débutée avec des travaux sur les graphes acycliques, qui étaient destinés à la visualisation de graphes [38, 39]. Bien qu'utilisant la même idée générale, les automates acycliques déterministes ne ressemblent pas tant que cela aux graphes acycliques, principalement parce qu'ils n'ont qu'un nombre linéaire d'arcs étiquetés (transitions). En particulier, le diamètre de la chaîne de Markov, qui se trouve être une borne inférieure du temps de mixage, est quadratique pour les graphes acycliques mais linéaire dans notre cas. De plus, les automates considérés ici doivent être accessibles, ce qui n'est pas une condition naturelle pour les graphes (il n'y a pas de notion de sommet initial) ; Melançon et Philippe ont considéré des graphes acycliques simplement connectés dans [39], mais

ce n'est pas la même notion d'accessibilité. Ils utilisent une optimisation élégante basée sur le renversement d'arc, ce qui préserve la connectivité mais pas l'accessibilité ; donc cela ne peut pas être réutilisé ici.

Dans la partie 1, nous rappelons quelques notations basiques à propos des automates ; et dans la partie 2 quelques concepts classiques sur les chaînes de Markov sont détaillés. Un générateur d'automates acycliques est décrit dans la Section 3, et sont exactitude est prouvée dans la Section 3.2. L'algorithme sur les automates acycliques minimaux est décrit dans la Section 4, et son exactitude est prouvée dans la Section 4.2. Finalement, dans la partie 5, nous expliquons comment adapter le second algorithme pour traiter le cas de la génération d'automates acycliques minimaux avec certaines contraintes sur l'ensemble des états finals.

1 Définitions et notations

Dans cette section, nous fournissons quelques définitions, certaines inédites et d'autres bien connues, qui seront indispensables pour la suite.

1.1 Automates

Dans cette partie, nous ne manipulerons que des automates déterministes, c'est à dire des automates qui sont des tuples $\langle Q, A, \delta, \{i\}, F\rangle$ où l'unique état initial est i et où la fonction de transition (partielle) δ est une fonction de $Q \times A$ dans Q (l'image d'une paire (état,lettre) ne peut être qu'un singleton ou $\emptyset$). Dans la suite, si p est un état d'un automate $\mathcal{A}$, on note A_p l'ensemble des lettres $a \in A$ telles que $\delta(p, a)$ est défini.

Définition 1.1 Un état $q \in Q$ est *transitoire* si pour tout $w \in A^+$, $\delta(q, w) \neq q$. Un état qui n'est pas transitoire est appelé *récurrent*. Un automate est *acyclique* quand chacun de ses états est transitoire.

Cette nouvelle définition d'automate acyclique nous sera utile dans la suite. A noter qu'il est impossible pour un automate complet d'être acyclique.

Dans la suite, sans perte de généralité, l'ensemble des états Q d'un automate déterministe à n états sera toujours $\{1, \ldots, n\}$, 1 sera toujours l'état initial, et nous assumons également qu'à partir de maintenant $n \geq 2$. De plus, étant donné que nous ne traitons désormais que des automates acycliques, accessibles et déterministes, nous les appellerons simplement "automates acycliques". On note $\mathbb{A}_n$ l'ensemble de tous les automates acycliques à n états. Nous considérons également, à partir de maintenant, que $|A| \geq 2$, le cas

$|A| = 1$ étant trivial pour les questions que nous traitons dans ce qui suit.

1.2 Automates en hamac

La notion d'automate en hamac va être très utile dans la seconde partie où nous nous concentrons sur les automates acyclique minimaux.

Définition 1.2 Un automate acyclique $\mathcal{A}$ est appelé automate en *hamac* s'il possède un unique état sans transition sortante, appelé l'*état cible*.

Proposition 1.1 *Soit $\mathcal{A} = (Q, A, \delta, \{1\}, F)$ un automate en hamac et soit $s \in Q$ un état de $\mathcal{A}$. Alors il existe un chemin de s à l'état cible.*

Démonstration. Étant donné que $\mathcal{A}$ est acyclique tous les chemins dans $\mathcal{A}$ ont une longueur bornée par $n-1$. Soit t l'état cible. Soit w un chemin tel que $\delta(s, w)$ est défini et tel que la longueur de w est maximale parmi les chemins partant de s. Par contradiction, on suppose que $\delta(s, w) = r \neq t$. Alors r possède une transition étiquetée par la lettre a. Donc $\delta(s, wa)$ est définie et $|wa| = |w| + 1 > |w|$. Ce qui est impossible étant donné que la longueur de w est maximale parmi les chemins partant de s. Donc $\delta(s, w) = t$. □

1.3 Automates minimaux

Pour des raisons pratiques, nous avons besoin de calculer l'automate minimal d'un langage rationnel $\mathcal{L}$ en partant d'un automate déterministe qui le reconnait. Pour cela, nous avons besoin d'une nouvelle relation d'équivalence, appelée *équivalence de Nerode*. Cette relation d'équivalence $\sim$ est définie sur les états d'un automate donné $\mathcal{A}$ de la façon suivante :

$$p \sim q \iff \mathsf{Fut}_{\mathcal{A}}(p) = \mathsf{Fut}_{\mathcal{A}}(q),$$

Si un automate est déterministe l'équivalence satisfait la propriété suivante.

$$p \sim q \iff \begin{cases} p \in F \iff q \in F, \\ A_p = A_q, \\ \forall a \in A_p,\ \delta(p, a) \sim \delta(q, a). \end{cases}$$

Théorème 1.2 *Un automate déterministe est* minimal *s'il est émondé*[1] *et si toute classe d'équivalence est réduite à un singleton.*

1. Dans la littérature l'automate minimal est défini soit comme un automate complet soit comme un automate émondé. Nous utilisons la seconde définition ici.

L'équivalence de Nérode peut être utilisée pour calculer l'automate minimal de $\mathcal{L}$ à partir d'un automate déterministe qui reconnaît $\mathcal{L}$: on fusionne deux états qui sont équivalents jusqu'à ce que la relation d'équivalence soit l'égalité. L'automate ainsi obtenu est exactement l'automate minimal du langage, et la plupart des algorithmes de minimisation procèdent de cette façon.

Comme nous le verrons dans la Section 4.2, les automates acycliques minimaux sont toujours des automates en hamac. De même que nous voulons que l'état initial soit étiqueté par 1, il est pratique de choisir une étiquette (toujours la même) pour l'état cible : dans ce qui suit nous notons $\mathbb{M}_n$ l'ensemble des automates acycliques minimaux à n états, dont l'état initial est 1 et l'état cible est n.

1.4 Automorphisme d'automates

Soit $\mathcal{A} = (Q, A, \delta, \{1\}, F)$ un automate déterministe. Une bijection ϕ de Q dans Q est un *automorphisme* quand

- $\phi(1) = 1$,
- $\forall p \in Q,\ A_p = A_{\phi(p)}$,
- $\forall p \in Q,\ \forall a \in A_p,\ \delta(\phi(p), a) = \phi(\delta(p, a))$,
- $\forall p \in Q,\ p \in F \iff \phi(p) \in F$.

Comme l'a fait remarqué Liskovets [29], le seul automorphisme d'un automate accessible et déterministe est l'identité. Ceci a l'importante conséquence combinatoire qu'il y a exactement $(n-1)!$ façons distinctes d'étiqueter avec $\{1, \ldots, n\}$ les états d'un tel automate de taille n, avec la convention que 1 est l'état initial. En particulier, pour une famille $\mathcal{F}_n$ d'automates accessibles et déterministes de taille n, stable par ré-étiquetage, le fait de générer aléatoirement et de façon uniforme un élément étiqueté de $\mathcal{F}_n$, puis de supprimer les étiquettes (formellement : de considérer sa classe d'équivalence à l'étiquetage près) n'influence pas l'uniformité.

Pour les éléments de $\mathbb{M}_n$, comme nous demandons que 1 soit l'état initial et n l'état cible, le même résultat est vrai, mais il y a maintenant exactement $(n-2)!$ éléments dans chaque classe d'équivalence, à l'étiquetage près.

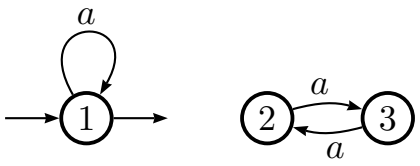


FIGURE 1.1 – Un automate déterministe qui n'est pas accessible. A noter que son groupe d'automorphisme n'est pas trivial : si nous permutons l'étiquette 2 avec l'étiquette 3 nous obtenons le même automate.

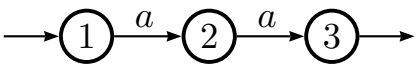


FIGURE 1.2 – Un automate déterministe accessible. A la différence de l'automate précédent, toute permutation non triviale des étiquettes produit un automate différent.

2 Chaînes de Markov et génération aléatoire

Dans cette section nous décrivons la chaîne de Markov utilisée pour générer aléatoirement et de façon presque uniforme les éléments de $\mathbb{A}_n$, pour tout $n \geq 2$ fixé. Nous en profitons pour rappeler la notion de base d'une chaîne de Markov dont nous aurons besoin dans la suite. Pour plus d'informations sur les chaînes de Markov pour la génération aléatoire, voir [28].

L'entrée de l'algorithme consiste en deux entiers positifs : le nombre d'états n, et le nombre d'itérations T. L'algorithme repose sur un processus de *chaîne de Markov* : il se déplace aléatoirement dans l'ensemble $\mathbb{A}_n$ et retourne l'automate atteint après T itérations.

La chaîne de Markov de l'algorithme peut être vue comme un graphe orienté dont les sommets sont des éléments de $\mathbb{A}_n$.

Définition 2.1 Une chaîne de Markov est dite *irréductible* quand son graphe est fortement connexe.

Un arc d'un automate $\mathcal{A}$ vers un autre automate $\mathcal{B}$ est étiqueté par un réel $r \in [0, 1]$, qui représente la probabilité de se déplacer de l'automate $\mathcal{A}$ à l'automate $\mathcal{B}$ en une seule itération. Pour deux automates $\mathcal{A}$ et $\mathcal{B}$ de $\mathbb{A}_n$, on note $\mathcal{P}_{\mathcal{A},\mathcal{B}}$ l'étiquette de l'arc de $\mathcal{A}$ à $\mathcal{B}$, s'il existe, sinon on note $\mathcal{P}_{\mathcal{A},\mathcal{B}} = 0$. Comme il s'agit d'une probabilité, on a :

$$\forall \mathcal{A} \in \mathbb{A}_n, \sum_{\mathcal{B} \in \mathbb{A}_n} \mathcal{P}_{\mathcal{A},\mathcal{B}} = 1.$$

Définition 2.2 Une distribution sur $\mathbb{A}_n$ est une application p de $\mathbb{A}_n$ dans $[0, 1]$ telle que

$$\sum_{\mathcal{A} \in \mathbb{A}_n} p(\mathcal{A}) = 1$$

Une *distribution stationnaire* π d'une chaîne de Markov est une distribution qui reste globalement inchangée après chaque déplacement aléatoire, *i.e.*

$$\forall \mathcal{B} \in \mathbb{A}_n, \qquad \pi(\mathcal{B}) = \sum_{\mathcal{A} \in \mathbb{A}_n} \pi(\mathcal{A}) \times \mathcal{P}_{\mathcal{A},\mathcal{B}}$$

On note $\mathcal{P}^{(i)}_{\mathcal{A},\mathcal{B}}$ la probabilité de se déplacer de $\mathcal{A}$ à $\mathcal{B}$ en i itérations de l'algorithme. On défini la *période* d'un sommet $\mathcal{A}$ comme le plus grand diviseur commun des longueurs de tous les circuits autour de $\mathcal{A}$: $pgcd(\{i \in \mathbb{N} \mid \mathcal{P}^{(i)}_{\mathcal{A},\mathcal{A}} > 0\})$. Un sommet est dit *apériodique* si sa période vaut 1 [2].

Définition 2.3 Une chaîne de Markov est *apériodique* lorsque tous ses états sont apériodiques. Une chaîne de Markov est *ergodique* lorsqu'elle est à la fois irréductible et apériodique.

Une célèbre propriété des chaînes de Markov ergodiques avec un nombre fini de sommets est qu'elles ont une unique distribution stationnaire et qu'en partant de n'importe lequel des sommets, la distribution obtenue après T itérations tend vers la distribution stationnaire lorsque T tend vers l'infini.

Théorème 2.1 *[[28]] Si une chaîne de Markov avec un nombre fini de sommets est ergodique alors son unique distribution stationnaire est la distribution uniforme.*

Cela fourni une méthode générique pour construire un générateur aléatoire sur un ensemble fini non-vide E : concevoir une chaîne de Markov ergodique dont l'ensemble des sommets est E et telle que la distribution stationnaire est la distribution uniforme. Partir de n'importe lequel des sommets, puis se déplacer (de façon aléatoire) pendant un temps suffisamment grand pour obtenir un élément de E aléatoirement et de façon quasiment uniforme.

C'est exactement ce que nous faisons dans notre cas. Une petite partie de la chaîne de Markov qui est derrière notre algorithme est représentée sur la Figure 2.1. Chaque itération consiste soit à ne rien faire, soit à modifier une transition. La description complète de l'algorithme est donnée dans la Section 3. Notre résultat principal, qui est prouvé dans la Section 3.2 est le suivant :

Théorème 2.2 *Pour tout $n \geq 2$, la chaîne de Markov sur $\mathbb{A}_n$ est ergodique et sa distribution stationnaire est la distribution uniforme.*

Comme les classes d'isomorphismes des automates à n états ont toutes la même cardinalité, notre générateur aléatoire uniforme sur $\mathbb{A}_n$ conduit à un générateur sur les classes d'isomorphismes d'automates qui est également uniforme (voir la Section 1.4).

A noter que le nombre d'itérations T doit être suffisamment grand pour

2. S'il y a une boucle de longueur 1 autour de $\mathcal{A}$, la période de $\mathcal{A}$ est 1 par définition.

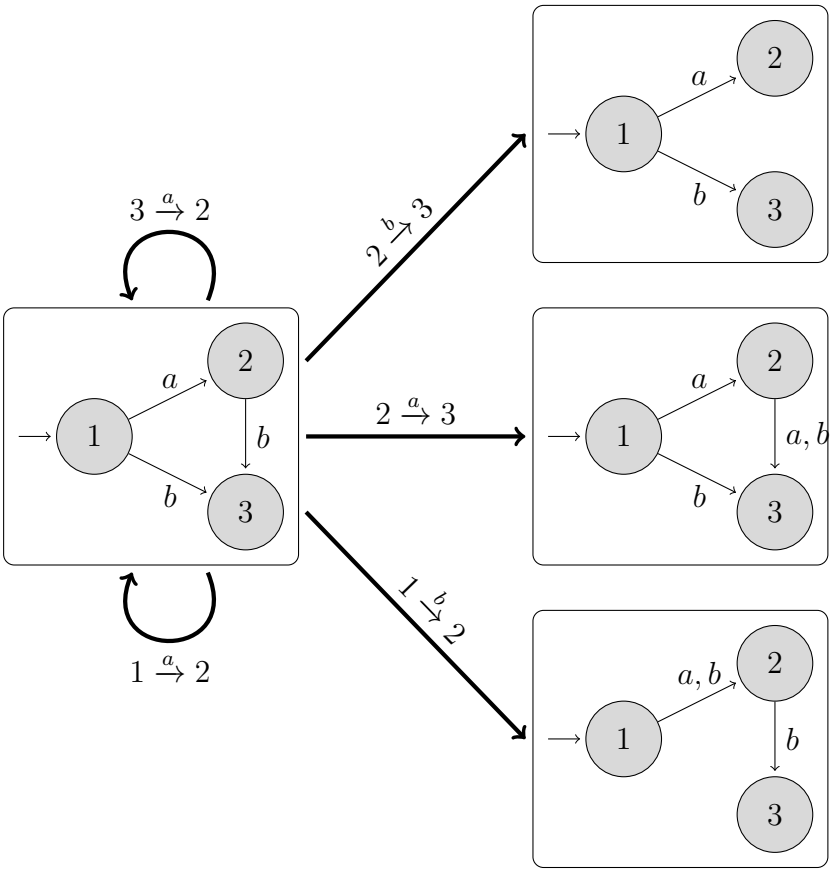


FIGURE 2.1 – Une partie de la chaîne de Markov : à chaque itération un élément $p \xrightarrow{a} q$ de $Q \times A \times Q$ est choisi aléatoirement. S'il correspond à une transition de l'automate, comme par exemple $2 \xrightarrow{b} 3$, alors on le supprime. S'il n'y a pas de transition étiquetée par a et partant de p, on l'ajoute ; c'est le cas pour $2 \xrightarrow{a} 3$. Lorsqu'il y a déjà une transition étiquetée par a et partant de p, elle est redirigée vers q ; c'est le cas pour $1 \xrightarrow{b} 2$. La mutation n'est pas appliquée si cela entraîne que l'automate n'est plus acyclique ($3 \xrightarrow{a} 2$) ou s'il n'est plus accessible ($1 \xrightarrow{a} 2$).

s'approcher assez près de la distribution uniforme. Le choix de T est un problème difficile [28] et ne sera pas étudié ici. Le diamètre du graphe de la chaîne de Markov constitue une borne inférieure pour T, et nous montrerons dans la Section 3.2 que, dans notre cas, ce diamètre est linéaire.

3 Génération aléatoire d'automates acycliques

3.1 Algorithme

Dans cette section, nous décrivons notre méthode de génération aléatoire pour des automates acycliques classiques dont le pseudocode est donné dans l'algorithme 1. La notation $\mathcal{A} \oplus p \xrightarrow{a} q$ représente l'automate $\mathcal{A}$ dans lequel on a ajouté la transition $p \xrightarrow{a} q$. De façon similaire, la notation $\mathcal{A} \ominus p \xrightarrow{a} q$ représente l'automate $\mathcal{A}$ où l'on a supprimé la transition $p \xrightarrow{a} q$, si elle existe.

Algorithm 1: Génération aléatoire d'un automate acyclique.

Input: n the number of state, T the desired number of iterations
Output: the generated automaton

$\mathcal{A} \leftarrow$ any deterministic, accessible and acyclic automaton with n states
for $i \leftarrow 0$ **to** T **do**
  $p \leftarrow Uniform(Q)$, $a \leftarrow Uniform(A)$, $q \leftarrow Uniform(Q \setminus \{p\})$
  if $\delta(p, a)$ *is undefined* **then**
    if *IsAcyclic*$(\mathcal{A} \oplus p \xrightarrow{a} q)$ **then** $\mathcal{A} = \mathcal{A} \oplus p \xrightarrow{a} q$
  else if $\delta(p, a) = q$ **then**
    if *IsAccessible*$(\mathcal{A} \ominus p \xrightarrow{a} q)$ **then** $\mathcal{A} = \mathcal{A} \ominus p \xrightarrow{a} q$
  else
    $r \leftarrow \delta(p, a)$
    if *IsAccessible*$(\mathcal{A} \ominus p \xrightarrow{a} r)$ **then**
      $\mathcal{A} = \mathcal{A} \ominus p \xrightarrow{a} r$
      if *IsAcyclic*$(\mathcal{A} \oplus p \xrightarrow{a} q)$ **then**
        $\mathcal{A} = \mathcal{A} \oplus p \xrightarrow{a} q$
      else
        $\mathcal{A} = \mathcal{A} \oplus p \xrightarrow{a} r$
Randomly choose the set of final states of $\mathcal{A}$
return $\mathcal{A}$

L'algorithme a deux arguments : le nombre n d'états et une valeur T qui indique le nombre souhaité d'itérations (il est assez difficile de savoir

quand la distribution uniforme est atteinte c'est pourquoi il est pratique de le spécifier). Après avoir choisi n'importe quel automate acyclique $\mathcal{A}$ dans $\mathbb{A}_n$ comme point de départ, l'algorithme répète les étapes suivantes T fois : choisir uniformément un arc étiqueté $p \xrightarrow{a} q$ avec $p \neq q$ ($p = q$ n'est pas intéressant étant donné que nous traitons des automates acycliques). Ensuite il y a trois cas possibles :

- Il n'y a pas de transition partant de p et étiquetée par a. Dans ce cas, on essaye d'ajouter $p \xrightarrow{a} q$ à $\mathcal{A}$ et on teste s'il est toujours acyclique auquel cas la transition est ajoutée.
- Il y a déjà une transition $p \xrightarrow{a} q$ dans $\mathcal{A}$. Dans ce cas, on teste si $\mathcal{A}$ est toujours accessible lorsqu'on la supprime. S'il l'est, la transition est supprimée, sinon $\mathcal{A}$ reste inchangé.
- Il y a une transition partant de p, étiquetée par a et arrivant dans un état r, avec $r \neq q$. Dans ce dernier cas, on teste tout d'abord si $\mathcal{A}$ est toujours accessible si l'on redirige $\delta(p, a)$ vers q. Si c'est le cas, on applique la redirection, sinon $\mathcal{A}$ reste inchangé.

Au cours de ce processus, nous avons régulièrement besoin de vérifier l'accessibilité et l'acyclicité de $\mathcal{A}$.

Le test pour l'accessibilité est implémenté de la façon suivante. Nous gardons à jour, pour tout état q, un compteur qui indique le nombre total de transitions arrivant sur q. Chaque fois qu'on ajoute ou qu'on supprime une telle transition, ce compteur est incrémenté ou bien décrémenté. Ainsi, pour tester l'accessibilité, nous n'avons qu'à vérifier, après que la transition ait été supprimée, si le compteur de l'état sur lequel arrivait la transition a atteint 0 ou pas ; ceci est une conséquence du Lemme 2 (voir la Section 3.2). La complexité en temps est clairement en $O(1)$.

L'acyclicité est testée par l'algorithme classique, utilisant un parcours en profondeur qui s'exécute en temps $O(n)$, étant donné que le nombre de transitions est linéaire dans un automate déterministe.

Nous obtenons ainsi le résultat suivant.

Proposition 3.1 *Chaque itération de l'algorithme est exécutée en temps $O(n)$. La complexité en temps de l'algorithme dans le pire cas est $O(Tn)$ et sa complexité en espace est $O(n)$.*

3.2 Preuves

Dans cette section, nous prouvons les principaux faits qui sont utilisés dans le premier algorithme pour générer correctement un automate acyclique avec une distribution quasi uniforme, et avec la complexité annoncée.

Une opération qui consiste à supprimer, ajouter ou modifier une transition est appelée une *opération élémentaire*.

Lemme 3.2 *Soit $\mathcal{A}$ un automate acyclique de taille n et $\mathcal{B} = \mathcal{A} \ominus p \xrightarrow{a} q$, où $q \neq 1$ et $p \xrightarrow{a} q$ est n'importe laquelle des transitions de $\mathcal{A}$. S'il existe au moins une transition qui arrive en q dans $\mathcal{B}$ alors $\mathcal{B}$ est accessible.*

Démonstration. $\mathcal{B}$ est clairement acyclique. $\mathcal{B}$ possède une transition $r \xrightarrow{b} q$, pour un certain état r et une certaine lettre b. L'état r est accessible dans $\mathcal{A}$, et $r \neq q$. Comme $\mathcal{A}$ et $\mathcal{B}$ ne diffèrent que par une transition qui arrive en q, r est également accessible dans $\mathcal{B}$. Donc, q est accessible dans $\mathcal{B}$ parce que l'on peut suivre un chemin de 1 à r, puis utiliser la transition $r \xrightarrow{b} q$. Comme tous les autres états restent accessibles pour la même raison que r, $\mathcal{B}$ est accessible. □

A noter que le résultat du Lemme 2 ne s'applique pas aux automates qui ne sont pas acycliques.

Lemme 3.3 *La chaîne de Markov de l'algorithme est symétrique, c'est à dire que, pour tout $\mathcal{A}$ et $\mathcal{B}$ dans $\mathbb{A}_n$, $\mathcal{P}_{\mathcal{A},\mathcal{B}} = \mathcal{P}_{\mathcal{B},\mathcal{A}}$.*

Démonstration. On rappelle que la probabilité de tirer un triplet (p, a, q) donné, avec $p \in Q$, $q \in Q \backslash \{p\}$, et $a \in A$ est $\frac{1}{n(n-1)|A|}$. Soient $\mathcal{A}$ et $\mathcal{B}$ dans $\mathbb{A}_n$ tels que $\mathcal{P}_{\mathcal{A},\mathcal{B}} > 0$. Alors il existe une opération élémentaire qui transforme $\mathcal{A}$ en $\mathcal{B}$. Supposons que $\mathcal{B} = \mathcal{A} \oplus p \xrightarrow{a} q$. La probabilité de tirer le triplet (p, a, q) est $\frac{1}{n(n-1)|A|}$. Maintenant, à partir de $\mathcal{B}$, la seule opération élémentaire pour atteindre $\mathcal{A}$ est de supprimer la transition $p \xrightarrow{a} q$. Donc, nous avons besoin de tirer le triplet (p, a, q) et la probabilité de cet évènement est $\frac{1}{n(n-1)|A|}$. Si $\mathcal{B} = \mathcal{A} \ominus p \xrightarrow{a} q$ alors $\mathcal{A} = \mathcal{B} \oplus p \xrightarrow{a} q$ et donc nous sommes dans le même cas que précédemment et $\mathcal{P}_{\mathcal{A},\mathcal{B}} = \mathcal{P}_{\mathcal{B},\mathcal{A}}$.

Supposons que l'opération élémentaire qui transforme $\mathcal{A}$ en $\mathcal{B}$ soit de rediriger la transition $p \xrightarrow{a} q$ de $\mathcal{A}$ pour obtenir $p \xrightarrow{a} s$ dans $\mathcal{B}$. Pour cela, il faut tirer le triplet (p, a, s) et la probabilité de cet évènement est $\frac{1}{n(n-1)|A|} = \mathcal{P}_{\mathcal{A},\mathcal{B}}$. La seule opération élémentaire pour atteindre $\mathcal{A}$ à partir de $\mathcal{B}$ est de rediriger la nouvelle transition $p \xrightarrow{a} s$ en $p \xrightarrow{a} q$ qui possède la même probabilité, pour les mêmes raisons. Ainsi $\mathcal{P}_{\mathcal{A},\mathcal{B}} = \mathcal{P}_{\mathcal{B},\mathcal{A}}$ dans ce cas également. □

Lemme 3.4 *La chaîne de Markov de l'algorithme est ergodique.*

FIGURE 3.1

Démonstration. Il nous faut montrer qu'elle est à la fois irréductible et apériodique.

Pour prouver l'irréductibilité, nous montrons que, dans la chaîne de Markov, il y a un chemin partant de n'importe lequel des automates acycliques $\mathcal{A}$ de $\mathbb{A}_n$ vers un automate S_n de $\mathbb{A}_n$, où S_n est l'automate acyclique dont les seules transitions sont $i \xrightarrow{a} i+1$, pour $i \in \{1, \ldots, n-1\}$ (illustré sur la figure 3.1).

Soit $\mathcal{A}$ un automate acyclique et soit a une lettre dans A. Soit E l'ensemble des états qui sont accessibles depuis l'état initial en ne lisant que la lettre a (plusieurs fois éventuellement). E n'est pas vide étant donné qu'il contient au moins l'état initial. De façon répétée, on supprime toute transition $p \xrightarrow{\alpha} q$ où $q \in E$ et $p \notin E$. Ensuite, on supprime toute transition restante $p \xrightarrow{\alpha} q$ où $p, q \in E$ et $\alpha \neq a$. Ces actions sont des déplacements valides dans la chaîne de Markov d'après le Lemme 2 étant donné que nous conservons à tout moment les transitions $p \xrightarrow{a} q$ avec $p, q \in E$. Soit ℓ l'unique état dans E duquel aucune transition étiquetée par a ne sort.

Si $|E| < n$, on choisit un état s de $\mathcal{A}$ qui n'est pas dans E et on ajoute une transition $\ell \xrightarrow{a} s$. Comme il n'y a pas de chemin entre s et un état de E, cette opération ne peut pas créer un cycle. Ensuite, on supprime itérativement toutes les transitions arrivant sur s excepté $\ell \xrightarrow{a} s$. On ajoute s à E, l'ensemble E a grandi d'un état. La taille de E étant finie, cette opération peut être répétée jusqu'à ce que E contienne tous les états de $\mathcal{A}$.

Ainsi, à partir d'un certain point, $|E| = n$ et $\mathcal{A}$ est isomorphe à S_n, étant donné que tous les états sauf l'état initial possèdent une unique transition entrante, qui est étiquetée par a. La seule différence avec S_n est que les états ne sont pas nécessairement dans le bon ordre. Nous expliquons donc maintenant comment on peut les ré-ordonner.

Soit $b \in A$ tel que $b \neq a$. Pour toute transition $p \xrightarrow{a} q$ de $\mathcal{A}$, nous ajoutons à $\mathcal{A}$ la transition $p \xrightarrow{b} q$ par le biais d'opérations élémentaires, qui ne créent pas de cycle. Désormais nous supprimons toutes les transitions étiquetées par a, $\mathcal{A}$ reste accessible grâce aux transitions étiquetées par b. Nous sommes dans le cas $|E| < n$ vu plus haut, où l'ensemble E ne contient que l'état 1. Pour atteindre l'automate S_n, il suffit de choisir les nouveaux états à ajouter à E comme vu précédemment mais cette fois-ci dans l'ordre de leur étiquette. Après avoir supprimé toutes les transitions étiquetées par b, nous obtenons finalement l'automate S_n.

Donc pour tout $\mathcal{A}$ dans $\mathbb{A}_n$, il existe un chemin de $\mathcal{A}$ à S_n dans la chaîne de Markov. D'après le Lemme 3 il existe également un chemin de S_n à $\mathcal{A}$: la chaîne de Markov est donc irréductible. Pour tout automate $\mathcal{A}$ dans $\mathbb{A}_n$, pour tout état $p \neq 1$ dans $\mathcal{A}$ et pour toute lettre a dans A, si l'arc choisi par l'algorithme est $(p, a, 1)$ alors $\mathcal{A}$ reste inchangé : ajouter la transitions ferait apparaître un cycle. Ainsi, tout sommet dans la chaîne de Markov possède un cycle de taille 1 (une boucle), elle est donc apériodique. □

du Théorème 2. C'est une conséquence des lemmes précédents : d'après le Lemme 4 la chaîne de Markov de l'algorithme est ergodique et d'après le Lemme 3 elle est symétrique. D'après un résultat classique en théorie des chaînes de Markov [28], sa distribution stationnaire est la distribution uniforme sur $\mathbb{A}_n$. □

Lemme 3.5 *Le diamètre de la chaîne de Markov est de l'ordre de* $\Theta(n)$.

Démonstration. En utilisant la construction proposée dans la preuve du Lemme 4, tout $\mathcal{A} \in \mathbb{A}_n$ est à une distance d'au plus $(|A| + 5)n$ par rapport à S_n. Le diamètre de la chaîne de Markov est donc d'au plus $2(|A| + 5)n$, ce qui est de l'ordre de $O(n)$. La borne inférieure, de l'ordre de $\Omega(n)$, est obtenue en considérant la distance de S_n par rapport à un automate acyclique dont les arcs sont étiquetés par une lettre $b \neq a$. □

4 Génération aléatoire d'automates acycliques minimaux

4.1 Algorithme

Dans cette section, nous expliquons comment modifier le premier algorithme de façon à générer, cette fois-ci, des automates acycliques minimaux.

L'algorithme 2 est très semblable à l'algorithme 1. Il possède deux arguments, n le nombre d'états et T le nombre d'itérations voulu. L'algorithme part d'un automate acyclique minimal de taille n et se déplace aléatoirement T fois dans une chaîne de Markov sur $\mathbb{M}_n$ l'ensemble des automates acycliques minimaux de taille n. Ensuite il retourne le dernier automate atteint.

Algorithm 2: Génération aléatoire d'un automate acyclique minimal

Input: n the number of state, T the desired number of iterations
Output: the generated automaton

$\mathcal{A} \leftarrow$ any minimal and acyclic automaton of $\mathbb{M}_n$
for $i \leftarrow 0$ **to** T **do**
  $x \leftarrow Uniform(\{0, 1\})$
  if $x = 0$ **then**
    $p \leftarrow Uniform(Q)$, $a \leftarrow Uniform(A)$, $q \leftarrow Uniform(Q \setminus \{p\})$
    if $\delta(p, a)$ *is undefined* **then**
      if *IsAcyclic*($\mathcal{A} \oplus p \xrightarrow{a} q$) ***and*** *IsMinimal*($\mathcal{A} \oplus p \xrightarrow{a} q$) **then**
        $\mathcal{A} = \mathcal{A} \oplus p \xrightarrow{a} q$
    else if $\delta(p, a) = q$ **then**
      if *IsTrim*($\mathcal{A} \ominus p \xrightarrow{a} q$) ***and*** *IsMinimal*($\mathcal{A} \ominus p \xrightarrow{a} q$) **then**
        $\mathcal{A} = \mathcal{A} \ominus p \xrightarrow{a} q$
    else
      $r \leftarrow \delta(p, a)$
      if *IsTrim*($\mathcal{A} \ominus p \xrightarrow{a} r$) **then**
        $\mathcal{A} = \mathcal{A} \ominus p \xrightarrow{a} r$
        if *IsAcyclic*($\mathcal{A} \oplus p \xrightarrow{a} q$) ***and*** *IsMinimal*($\mathcal{A} \oplus p \xrightarrow{a} q$) **then**
          $\mathcal{A} = \mathcal{A} \oplus p \xrightarrow{a} q$
        else
          $\mathcal{A} = \mathcal{A} \oplus p \xrightarrow{a} r$
  else
    $p \leftarrow Uniform(Q)$
    if *IsMinimal*($\mathcal{A} \oplus \overline{p}$) **then**
      $\mathcal{A} = \mathcal{A} \oplus \overline{p}$
return $\mathcal{A}$

Un déplacement dans la chaîne de Markov se fait en appliquant un des deux cas suivants avec une probabilité $\frac{1}{2}$ pour chacun d'eux :

- Tirer de façon uniforme un état $p \in Q$ et modifier son statut final (cette opération est représentée par $\overline{p}$). Cela signifie que si p est un état final, nous le rendons non final, et s'il ne l'est pas, on le rend final.
- Tirer de façon uniforme deux états $p, q \in Q$ et une lettre $a \in A$. Nous avons ensuite les trois cas de l'algorithme précédent excepté qu'ici nous avons également besoin de vérifier que l'automate résultant est toujours minimal. De plus, nous ne faisons pas que vérifier l'accessibilité mais aussi la co-accessibilité, ce qui signifie que l'automate doit être émondé.

A noter que dans cet algorithme, nous utilisons trois tests pour vérifier si l'automate est acyclique, émondé et minimal.

Pour savoir si l'automate est acyclique, nous utilisons le même algorithme de parcours en profondeur que précédemment qui s'exécute en temps $O(n)$. Nous avons encore besoin de vérifier l'accessibilité mais nous avons désormais besoin de vérifier la co-accessibilité également car il faut que l'automate résultant soit émondé. Nous utilisons donc le même principe que pour l'accessibilité et nous l'appliquons à la co-accessibilité en comptant le nombre de transitions sortantes de chaque état. La démonstration de l'exactitude de ce test est donnée dans la Section 4.2. Il est toujours clair que ce test s'exécute en temps constant. Pour le test de minimalité, nous pouvons utiliser l'algorithme de Revuz [44] qui traite spécialement des automates acycliques et dont le temps d'exécution est linéaire par rapport au nombre d'états.

Pour chaque déplacement l'algorithme réalise au plus quatre tests, celui d'accessibilité, celui de co-accessibilité, celui d'acyclicité et celui de minimalité. La complexité en temps de chaque test est de l'ordre de $O(n)$ et comme nous pratiquons T déplacements dans l'algorithme, nous obtenons le résultat suivant.

Proposition 4.1 *Pour la complexité en temps, le pire cas de l'algorithme qui génère des automates acycliques minimaux est de l'ordre de $O(nT)$ et sa complexité en espace est $O(n)$.*

La propriété suivante prouve l'efficacité de l'algorithme.

Théorème 4.2 *La chaîne de Markov de l'algorithme qui génère des automates acycliques minimaux est ergodique et sa distribution stationnaire est la distribution uniforme sur $\mathbb{M}_n$.*

Notre preuve du Théorème 2 donne également des informations sur le diamètre de la chaîne de Markov sous-jacente.

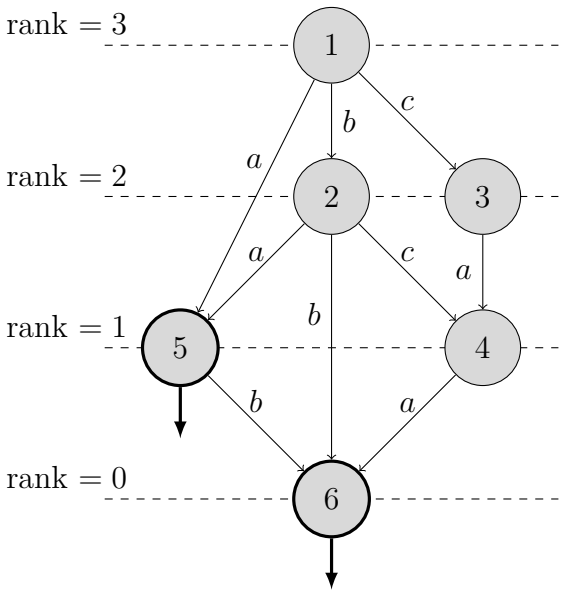


FIGURE 4.1 – Un automate en hamac avec 6 états. Les états ont été organisés de façon à mettre en avant la valeur de leur rang.

Proposition 4.3 *Le diamètre de la chaîne de Markov qui génère les automates acycliques minimaux est de l'ordre de* $\Theta(n)$.

4.2 Preuves

Cette section est dédiée à la preuve du Théorème 2 et à l'estimation du diamètre de la chaîne de Markov sous-jacente.

Comme un automate en hamac est acyclique, la longueur d'un chemin de n'importe lequel des états à l'état n est bornée supérieurement par $n-1$. Nous utilisons cette remarque pour définir le rang d'un état dans un automate en hamac.

Définition 4.1 Soit $\mathcal{A}$ un automate en hamac. Pour tout état p de $\mathcal{A}$, le *rang* de p, noté $\eta_{\mathcal{A}}(p)$ (ou $\eta(p)$ s'il n'y a pas d'ambiguité), est la longueur du plus long chemin parmi tous les chemins de p à l'état cible n.

Comme deux états équivalents ont le même futur, ils ont le même rang. Remarquons également que si p est un ancêtre de q alors $\eta(p) > \eta(q)$. De plus, si p n'est pas l'état cible, alors il possède un successeur direct q tel que $\eta(p) = \eta(q) + 1$.

Définition 4.2 Soit $\mathcal{A} = (Q, A, \delta, \{1\}, F)$ un automate acyclique. Deux états p et q sont *directement équivalents* s'ils satisfont les propriétés suivantes.

- $p \in F \iff q \in F$,
- $A_p = A_q$,
- $\forall a \in A_q$, $\delta(p, a) = \delta(q, a)$.

Observons que deux états directement équivalents sont également équivalents au sens de Nérode.

Lemme 4.4 *Soit $\mathcal{A} = (A, Q, \delta, \{1\}, F)$ un automate acyclique tel que l'état n n'a pas de transition. $\mathcal{A}$ est un automate minimal si et seulement si $n \in F$ et si $\mathcal{A}$ est un automate en hamac tel que deux états distincts de $\mathcal{A}$ ne sont jamais directement équivalents.*

Démonstration. Nous supposons tout d'abord que $\mathcal{A}$ est un automate acyclique minimal. Soient t et r deux états de $\mathcal{A}$ sans transition sortante. Les états t et r doivent être des états finals, sinon $\mathcal{A}$ n'est pas émondé. Donc $t \in F$ et $r \in F$ et $A_r = A_t = \emptyset$. Ainsi $t \sim s$ et comme $\mathcal{A}$ est minimal, alors $t = s$. Cela prouve que n est le seul état sans transition sortante et que $n \in F$. Donc $\mathcal{A}$ est un automate en hamac. Soient p et q deux états qui sont directement équivalents et donc également équivalents au sens de Nérode. Comme $\mathcal{A}$ est minimal, $p = q$.

Supposons maintenant que A est un automate en hamac d'état cible n, lequel est final, et tel que deux états différents ne sont jamais directement équivalents. Il y a un chemin de n'importe quel état vers l'état n, qui est dans F. Cela signifie que $\mathcal{A}$ est co-accessible, et donc que $\mathcal{A}$ est émondé. Étant donné que $\mathcal{A}$ est un automate en hamac, nous pouvons utiliser la notion de rang (voir la Définition 1) pour prouver que $\mathcal{A}$ est minimal. Par l'absurde, supposons que $\mathcal{A}$ possède deux états qui sont équivalents, nous allons prouver qu'ils sont directement équivalents. L'ensemble $E = \{s \in Q, \exists s', s \neq s',\ s \sim s'\}$ n'est pas vide. Soit $s \in E$ un élément de rang minimal $\eta(s)$. Comme $s \in E$, il existe $s' \neq s$ tel que $s \sim s'$, et donc $\eta(s) = \eta(s')$. Pour tout $a \in A_s$, $\delta(s, a) \sim \delta(s', a)$, donc $\eta(\delta(s, a)) = \eta(\delta(s', a)) < \eta(s)$. Mais $\eta(s)$ est minimal dans E, c'est pourquoi $\delta(s, a)$ n'est pas dans E et ainsi $\delta(s, a) = \delta(s', a)$. Cela signifie que s et s' sont directement équivalents, ce qui est une contradiction : $\mathcal{A}$ est donc minimal. □

Lemme 4.5 *La chaîne de Markov de l'algorithme qui génère les automates acycliques minimaux est irréductible.*

Démonstration. Pour prouver que la chaîne de Markov est irréductible, nous devons montrer que pour toute paire d'automates acycliques minimaux il existe un chemin de l'un à l'autre dans la chaîne de Markov. Comme la chaîne de Markov est symétrique c'est suffisant pour prouver qu'à partir de n'importe quel automate $\mathcal{A}$, il existe un chemin vers un automate spécifique. Nous partons de $\mathcal{A} = (Q, A, \delta_{\mathcal{A}}, \{1\}, F)$ dans $\mathbb{M}_n$ et prouvons que nous pouvons atteindre l'automate en ligne $\mathcal{D} = (Q, A, \delta_{\mathcal{D}}, \{1\}, F_{\mathcal{D}})$, où $F_{\mathcal{D}} = \{n\}$, $\forall p \in Q \setminus \{n\}$, $A_p = \{a\}$ et $\delta(p, a) = p + 1$. Comme nous traitons des automates en hamac uniquement (voir le lemme4), nous utilisons la notion de rang pour les états.

Tout d'abord, nous supposons que $\eta_{\mathcal{A}}(1) = n - 1$ et nous montrons que l'on peut atteindre l'automate en ligne à partir de $\mathcal{A}$ en se déplaçant dans la chaîne de Markov de $\mathbb{M}_n$. Étant donné que $\eta_{\mathcal{A}}(1) = n - 1$, tous les états ont un rang différent, qui varie de 0 à $n - 1$. Tout état s possède au moins une transition arrivant sur l'état de rang $\eta(s) - 1$. Pour tout état $s \neq n$, nous retirons toutes les transitions partant de cet état excepté une qui arrive sur l'état de rang $\eta(s) - 1$ (il s'agit de déplacements valides dans la chaîne de Markov). Ensuite, parmi les états finals, nous supprimons tous les états de l'ensemble F, excepté l'état n. On obtient un automate en ligne $\mathcal{B}$ où les états ne sont pas nécessairement dans l'ordre et où les transitions ne sont pas obligatoirement étiquetées par la lettre a. A noter qu'après chaque modification les états conservent leur rang, ainsi l'automate reste minimal et acyclique.

Comme $|A| \geq 2$, alors pour tout état s de $\mathcal{B}$ il y a une lettre b telle que $\delta_{\mathcal{B}}(s, b)$ n'est pas défini. Maintenant nous allons montrer que pour deux états p et q tels que p est un ancêtre de q, avec $q \neq n$, nous pouvons ré-ordonner les états en utilisant des déplacements dans la chaîne de Markov, de façon à ce que p devienne un ancêtre direct de q sans pour autant changer l'ordre des ancêtres de p. Une fois que tous les états sont dans l'ordre, nous pouvons remplacer chaque transition par une transition étiquetée par a, atteignant ainsi l'automate $\mathcal{D}$. Le processus, décrit à la Figure 4.2 est le suivant. Ajoutons une transition de p à q, en utilisant une des lettres restantes. Rendons q final. A noter que, jusqu'à maintenant, le rang des états n'a pas changé, et que l'automate est toujours acyclique et minimal. Soient s l'ancêtre direct de q et r le successeur direct de q. Redirigeons la transition entre s et q vers r. Remarquons que l'automate reste un automate en hamac, et qu'à l'exception

des états r et 1, tous les états ne possèdent qu'une seule transition entrante. De plus q et s ne sont pas directement équivalents car q est un état final alors que s ne l'est pas. Ainsi l'automate est toujours acyclique et minimal. Soit t le successeur direct de p. Comme q n'est accessible qu'à partir de p, t n'est pas un ancêtre de q. Redirigeons la transition de q vers t. Étant donné que p est un ancêtre de q, p et q ne peuvent pas être directement équivalents. Désormais retirons la transition entre p et t et supprimons q de l'ensemble des états finals. Nous avons modifié le successeur direct de p. L'ordre de tous les ancêtres de p n'a pas changé.

En utilisant ce processus de façon répétée, nous pouvons ré-ordonner les états pour obtenir l'automate $\mathcal{D}$.

Supposons maintenant que $\eta_{\mathcal{A}}(1) < n-1$. En utilisant des déplacements dans la chaîne de Markov, nous allons transformer $\mathcal{A}$ en un automate $\mathcal{B}$, où le rang de 1 dans $\mathcal{B}$ est plus grand que le rang de 1 dans $\mathcal{A}$. Après moins de $n-1$ déplacements dans la chaîne de Markov, nous atteindrons un automate où le rang de 1 est $n-1$, de façon à ce que l'on puisse appliquer la construction vue au-dessus.

Pour tout état $s \in Q$, nous avons $0 \leq \eta_{\mathcal{A}}(s) \leq \eta_{\mathcal{A}}(1) \leq n-2$. Comme $|Q| = n$, il existe deux états p, p' tels que $\eta_{\mathcal{A}}(p) = \eta_{\mathcal{A}}(p')$.

Soit p un état tel que $\eta_{\mathcal{A}}(p)$ est minimal et tel qu'il existe un autre état p' avec $\eta_{\mathcal{A}}(p) = \eta_{\mathcal{A}}(p')$. Soit $q \neq p$ un autre état tel que $\eta_{\mathcal{A}}(q)$ est maximal dans l'ensemble des états qui ne sont pas des ancêtres de p (cet ensemble n'est pas vide étant donné qu'il contient au moins p'). L'état p possède au moins une transition arrivant sur un état r de rang $\eta_{\mathcal{A}}(r) = \eta_{\mathcal{A}}(p) - 1$. Soit $\mathcal{B}$ l'automate obtenu à partir de $\mathcal{A}$ en redirigeant cette transition vers q.

A présent nous prouvons que $\mathcal{B}$ est minimal et acyclique. Il est clair que $\mathcal{B}$ est déterministe, et, étant donné que la redirection est effectuée vers q, qui n'est pas un ancêtre de p dans $\mathcal{A}$, $\mathcal{B}$ est également acyclique. L'état p' possède une transition arrivant sur un état r' tel que $\eta_{\mathcal{A}}(p') - 1 = \eta_{\mathcal{A}}(r')$. Ainsi $\eta_{\mathcal{A}}(r') = \eta_{\mathcal{A}}(r) < \eta_{\mathcal{A}}(p)$ et donc $r = r'$, étant donné qu'il ne peut y avoir deux états distincts de même rang et strictement plus petit que $\eta_{\mathcal{A}}(p)$, d'après la définition de p. En conséquence, il y a une transition de p' à r dans $\mathcal{B}$, et $\mathcal{B}$ est accessible. Comme l'unique état de $\mathcal{A}$ sans aucune transition sortante est n, n est également le seul état sans aucune transition sortante dans $\mathcal{B}$. Ainsi $\mathcal{B}$ est un automate en hamac. On note $\eta_{\mathcal{B}}$ le rang des états de $\mathcal{B}$ et $\delta_{\mathcal{B}}$ sa fonction de transition. A noter que $\eta_{\mathcal{A}}(p) < \eta_{\mathcal{B}}(p)$ vu que $\eta_{\mathcal{A}}(q) \geq \eta_{\mathcal{A}}(p') = \eta_{\mathcal{A}}(p)$, et qu'il y a une transition de p à q dans $\mathcal{B}$.

En raisonnant par l'absurde, supposons que $\mathcal{B}$ n'est pas minimal ; alors il existe deux états o et o', $o' \neq o$, qui ne sont pas directement équivalents dans $\mathcal{B}$ (voir le Lemme 4). A noter que $o = p$ ou $o' = p$, sinon les transitions de o et

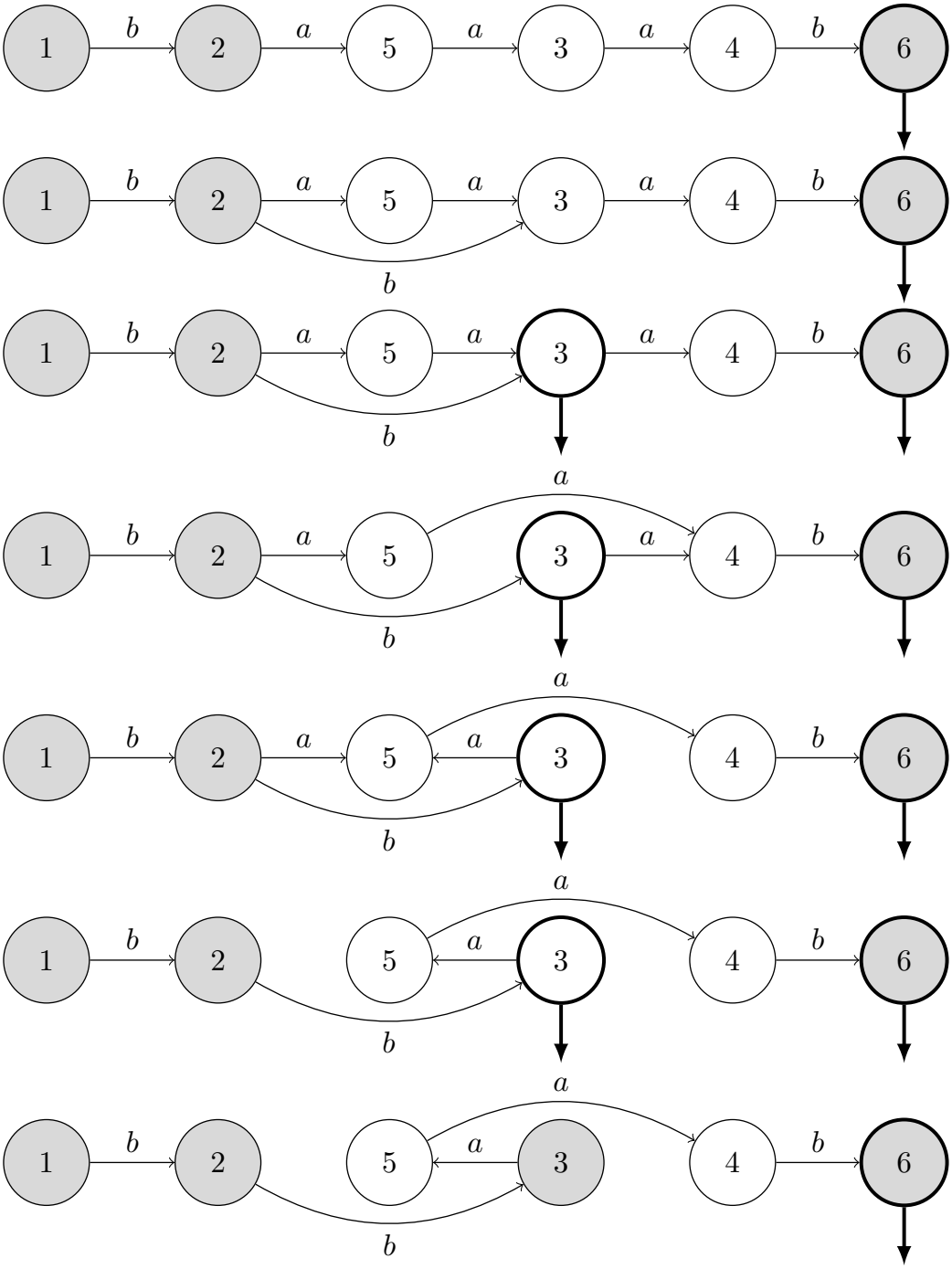


FIGURE 4.2 – Différents déplacements dans une chaîne de Markov pour positionner l'état 3 après l'état 2. Au départ, seuls les états 1,2 et 6 sont correctement ordonnés. Ensuite, après cinq déplacements, les états 1,2,3 et 6 sont dans l'ordre

de o' seraient les mêmes dans $\mathcal{A}$ et dans $\mathcal{B}$, et donc o et o' seraient directement équivalents dans $\mathcal{A}$, ce qui est impossible. Sans perte de généralité, nous supposons que $p = o'$. Comme o est directement équivalent à p dans $\mathcal{B}$, alors o n'est pas un ancêtre de p dans $\mathcal{B}$ et o est un ancêtre de q dans $\mathcal{B}$. Les transitions de o sont les mêmes dans $\mathcal{A}$ et dans $\mathcal{B}$. En conséquence, o n'est pas un ancêtre de p dans $\mathcal{A}$ et est un ancêtre de q dans $\mathcal{A}$. Donc $\eta_{\mathcal{A}}(q) < \eta_{\mathcal{A}}(o)$, ce qui est absurde étant donné que $\eta_{\mathcal{B}}(q)$ est maximal parmi les états qui ne sont pas des ancêtres de p dans $\mathcal{A}$. Ce qui entraîne que $\mathcal{B}$ est minimal.

Nous prouvons maintenant que le rang de n'importe quel ancêtre s de p a augmenté : $\eta_{\mathcal{A}}(s) < \eta_{\mathcal{B}}(s)$. En raisonnant par l'absurde, supposons que l'ensemble $J = \{j \in Q,\ j$ est un ancêtre de p dans $\mathcal{B}$ et $\eta_{\mathcal{B}}(j) \leq \eta_{\mathcal{A}}(j)\}$ n'est pas vide. Soit s un état de J tel que $\eta_{\mathcal{A}}(s)$ est minimal. Soit t un successeur direct de s tel que $\eta_{\mathcal{A}}(t) = \eta_{\mathcal{A}}(s) - 1$ in $\mathcal{A}$. Trois cas se présentent :

• S'il n'y a pas de chemin de t à p dans $\mathcal{A}$: comme s est un ancêtre de p dans $\mathcal{B}$ et dans $\mathcal{A}$ alors $\eta_{\mathcal{B}}(p) < \eta_{\mathcal{B}}(s)$. De plus $s \in J$, donc $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{A}}(s)$, et $\eta_{\mathcal{B}}(p) \leq \eta_{\mathcal{B}}(s) - 1 \leq \eta_{\mathcal{A}}(s) - 1 = \eta_{\mathcal{A}}(t)$. Étant donné que p est un ancêtre direct de q dans $\mathcal{B}$, par définition de la transformation amenant à l'automate $\mathcal{B}$, alors $\eta_{\mathcal{B}}(q) < \eta_{\mathcal{B}}(p)$, et donc $\eta_{\mathcal{B}}(q) < \eta_{\mathcal{A}}(t)$. Étant donné que, d'après la construction, il y a un chemin de q à p dans $\mathcal{A}$, le rang de q n'est pas changé au cours de la transformation : $\eta_{\mathcal{B}}(q) = \eta_{\mathcal{A}}(q)$. Donc $\eta_{\mathcal{A}}(q) < \eta_{\mathcal{A}}(t)$. C'est impossible étant donné que q a été choisi comme état de rang maximal dans $\mathcal{A}$ parmi les états qui ne sont pas des ancêtres de p, et t est l'un de ces états.

• Si $t = p$: alors on a $\eta_{\mathcal{A}}(t) < \eta_{\mathcal{B}}(t)$, étant donné que le rang de p a augmenté. Donc $\eta_{\mathcal{A}}(t) + 1 = \eta_{\mathcal{A}}(s) \leq \eta_{\mathcal{B}}(t)$. Mais comme s appartient à J, $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{A}}(s)$, et donc $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{B}}(t)$. C'est également impossible car s est un ancêtre de t dans $\mathcal{B}$.

• Si t est un ancêtre de p : comme s est un ancêtre de t dans $\mathcal{A}$ alors $\eta_{\mathcal{A}}(t) < \eta_{\mathcal{A}}(s)$. Par hypothèse, le rang de s dans $\mathcal{A}$ est minimal parmi les états de J, et donc $t \notin J$. Étant donné que $t \notin J$ et que t est un ancêtre de p, on a $\eta_{\mathcal{A}}(t) < \eta_{\mathcal{B}}(t)$. Ainsi $\eta_{\mathcal{A}}(t) + 1 = \eta_{\mathcal{A}}(s) \leq \eta_{\mathcal{B}}(t)$. Comme s appartient à J, $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{A}}(s)$. En conséquence, $\eta_{\mathcal{B}}(s) \leq \eta_{\mathcal{B}}(t)$. Il s'agit à nouveau d'une contradiction, étant donné que s est un ancêtre de t dans $\mathcal{B}$.

Ainsi, J est vide et, pour tout ancêtre s de p, nous avons prouvé que $\eta_{\mathcal{A}}(s) < \eta_{\mathcal{B}}(s)$. C'est vrai en particulier pour l'état initial 1, dont le rang a augmenté au cours de la transformation de $\mathcal{A}$ vers $\mathcal{B}$, $\mathcal{B}$ étant toujours un automate acyclique minimal. En répétant cette construction, on obtient un automate minimal dont l'état initial a le rang $n - 1$, ce qui a été étudié dans la première partie de la preuve : en partant de n'importe quel automate acyclique minimal $\mathcal{A}$, il y a un chemin dans la chaîne de Markov qui atteint $\mathcal{D}$. □

Nous avons à présent les éléments pour prouver le Théorème 2.

Démonstration du Théorème 2. Pour montrer l'ergodicité de la chaîne de Markov, nous devons prouver qu'elle est irréductible et apériodique. L'irréductibilité de la chaîne de Markov est prouvée dans le Lemme 5. On observe que dans l'algorithme, à partir de n'importe quel automate minimal, nous avons une probabilité non nulle de choisir un triplet (p, a, q) qui permet de rester au même endroit dans la chaîne. Tous les sommets de la chaîne de Markov sont donc apériodiques et la chaîne de Markov l'est également. Ainsi la chaîne de Markov est ergodique, et elle possède une unique distribution stationnaire vers laquelle est converge. Et comme la chaîne de Markov est symétrique par construction, la distribution stationnaire est la distribution uniforme sur $\mathbb{M}_n$. □

Une estimation du diamètre de la chaîne de Markov peut être déduite de la preuve du Lemme 5.

Démonstration de la Proposition 3. Soit $\mathcal{A}$ un automate acyclique minimal quelconque de $\mathbb{M}_n$. Dans la preuve du Lemme 5 nous avons utilisé au plus $n-1$ déplacements dans la chaîne de Markov pour passer de $\mathcal{A}$ à un automate minimal $\mathcal{B}$ tel que $\eta_{\mathcal{B}}(1) = n - 1$. Ainsi pour tout état, nous supprimons au plus $|A|$ transitions pour obtenir un automate en ligne, et donc, de façon globale, au plus $n|A|$. Ensuite nous ré-ordonnons les étiquettes des états, en utilisant au plus 6 déplacements pour chaque état, comme décrit par la Figure 4.2, ce qui correspond, globalement, à au plus $6n$ déplacements. Nous remplaçons chaque transition par une autre transition étiquetée par la lettre a, en utilisant au plus $2n$ déplacements dans la chaîne de Markov. Finalement, en utilisant au plus n déplacements, nous pouvons supprimer tous les états finals sauf le dernier, ce qui amène à l'automate $\mathcal{D}$ avec un chemin en $O(n)$ déplacements. La borne inférieure en $\Omega(n)$ est facilement obtenue, en partant d'un automate en ligne qui ne possède aucune transition étiquetée par a.

Ainsi, l'automate le plus éloigné de $\mathcal{D}$ est à une distance $\Theta(n)$ et le diamètre de la chaîne est donc également $\Theta(n)$. □

5 Conclusion

Dans cette dernière partie, nous avons vu sur deux cas comment construire des chaînes de Markov dans le but de générer des automates acycliques, déterministes et accessibles, soit dans un cadre général, soit en se restreignant à la classe des automates acycliques minimaux. Ces techniques peuvent être

facilement adaptées pour de nombreuses autres familles d'automates acycliques déterministes : des contraintes simples peuvent être facilement prises en compte en changeant légèrement l'algorithme. Des exemples de contraintes simples pourraient être "l'état initial ne possède pas de transition sortante étiquetée par a" ou bien "le mot ab n'est pas reconnu".

Plus intéressant, la chaîne de Markov pour générer des automates acycliques minimaux peut être adaptée pour considérer des automates acycliques minimaux avec un nombre d'états finals fixé. La possibilité de changer la finalité d'un état doit être supprimée, et remplacée par la possibilité d'échanger le statut final de deux états au hasard (pour conserver le nombre d'états finals constant). La preuve de l'ergodicité peut être aisément adaptée et le diamètre serait grossièrement le même que dans le cas traité ici. Cela peut être utile de réduire (grandement) la probabilité de produire des automates minimaux possédant un grand nombre d'états finals, comme c'est le cas pour la distribution uniforme sur l'ensemble des automates acycliques minimaux.

La principale question qui subsiste et qui semble être sérieusement difficile, serait d'obtenir des bornes supérieures intéressantes sur le temps de mélange des chaînes de Markov que nous avons utilisées.

Liste des exemples

Index

Bibliographie

[1] M. Almeida, N. Moreira, and R. Reis. Exact generation of minimal acyclic deterministic finite automata. *Int. J. Found. Comput. Sci.*, 19(4) :751–765, 2008.

[2] Marcella Anselmo. Two-way automata with multiplicity. In *ICALP'90*, volume 443 of *Lect. Notes in Comput. Sci.*, pages 88–102, 1990.

[3] F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1-3) :86–104, 2007.

[4] M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch. Squaring transducers : an efficient procedure for deciding functionality and sequentiality. *Theor. Comput. Sci.*, 292(1) :45–63, 2003.

[5] J. A. Brzozowski and E. J. McCluskey. Signal flow graph techniques for sequential circuit state diagrams. *j-IEEE-TRANS-ELEC-COMPUT*, EC-12(2) :67–76, April 1963.

[6] Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4) :481–494, October 1964.

[7] V. Carnino and S. De Felice. Random generation of deterministic acyclic automata using markov chains. In *CIAA*, pages 65–75, 2011.

[8] V. Carnino and S. De Felice. Sampling different kinds of acyclic automata using markov chains. *Theor. Comput. Sci.*, 450 :31–42, 2012.

[9] V. Carnino and S. Lombardy. Factorizations and universal automaton of omega languages. In *Developments in Language Theory*, pages 338–349, 2013.

[10] V. Carnino and S. Lombardy. On determinism and unambiguity of weighted two-way automata. In *AFL*, pages 188–200, 2014.

[11] V. Carnino and S. Lombardy. Tropical two-way automata. *Theor. Comput. Sci.*, 8705 :195–206, 2014.

[12] P. Caron, J.-M. Champarnaud, and L. Mignot. Small extended expressions for acyclic automata. In S. Maneth, editor, *CIAA*, volume 5642 of *Lecture Notes in Computer Science*, pages 198–207. Springer, 2009.

[13] P. Caron, J.-M. Champarnaud, and L. Mignot. Acyclic automata and small expressions using multi-tilde-bar operators. *Theor. Comput. Sci.*, 411(38-39) :3423–3435, 2010.

[14] P. Caron and D. Ziadi. Characterization of glushkov automata. *Theor. Comput. Sci.*, 233(1-2) :75–90, 2000.

[15] O. Carton. Two-way transducers with a two-way output tape. In *DLT'12*, volume 7410 of *Lect. Notes in Comput. Sci.*, pages 263–272, 2012.

[16] O. Carton and M. Michel. Unambiguous büchi automata. *Theor. Comput. Sci.*, 297(1-3) :37–81, 2003.

[17] O. Carton, D. Perrin, and J.-E. Pin. Automata and semigroups recognizing infinite words. In *Logic and Automata*, pages 133–168, 2008.

[18] J.-M. Champarnaud and T. Paranthoën. Random generation of DFAs. *Theor. Comput. Sci.*, 330(2) :221–235, 2005.

[19] A.H. Clifford and G.B. Preston. *The Algebraic Theory of Semigroups.* Mathematical Surveys and Monographs.

[20] J. Horton Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, 1971.

[21] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability & Computing*, 13(4-5) :577–625, 2004.

[22] E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. From two-way to one-way finite state transducers. *CoRR*, abs/1301.5197, 2013.

[23] P. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132(2) :1–35, 1994.

[24] VM Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16(5) :1–53, 1961.

[25] P.-C. Héam, C. Nicaud, and S. Schmitz. Parametric random generation of deterministic tree automata. *Theor. Comput. Sci.*, 411(38-39) :3469–3480, 2010.

[26] I. Klimann, S. Lombardy, J. Mairesse, and C. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327(3) :349–373, 2004.

[27] Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Internat. J. Algebra Comput.*, 4(3) :405–425, 1994.

[28] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. AMS, 2008.

[29] Valery A. Liskovets. The number of initially connected automata. *Cybernetics*, 4 :259–262, 1969. English translation of Kibernetika (3) 1969, 16-19.

[30] Valery A. Liskovets. Exact enumeration of acyclic deterministic automata. *Discrete Applied Mathematics*, 154(3) :537–551, 2006.

[31] S. Lombardy. *Approche structurelle de quelques problèmes de la théorie des automates*. PhD thesis, Paris, ENST, Grenoble, 2001. Th. : informatique et réseaux.

[32] S. Lombardy. On the construction of reversible automata for reversible languages. In *ICALP*, pages 170–182, 2002.

[33] S. Lombardy and J. Mairesse. Series which are both max-plus and min-plus rational are unambiguous. *RAIRO - Theor. Inf. and Appl.*, 40(1) :1–14, 2006.

[34] S. Lombardy and J. Sakarovitch. Star height of reversible languages and universal automata. In *LATIN*, pages 76–90, 2002.

[35] S. Lombardy and J. Sakarovitch. The universal automaton. In *Logic and Automata*, pages 457–504, 2008.

[36] S. Lombardy and J. Sakarovitch. The validity of weighted automata. *Internat. J. Algebra Comput.*, 23 :863–913, 2013.

[37] R. McNaughton and H. Yamada. Regular expressions and finite state graphs for automata. *IRE Trans. on Electronic Comput*, EC-9(1) :38–47, 1960.

[38] G. Melançon, I. Dutour, and M. Bousquet-Mélou. Random generation of directed acyclic graphs. *Electronic Notes in Discrete Mathematics*, 10 :202–207, 2001.

[39] Guy Melançon and Fabrice Philippe. Generating connected acyclic digraphs uniformly at random. *Inf. Process. Lett.*, 90(4) :209–213, 2004.

[40] M. Mohri. String-matching with automata. *Nord. J. Comput.*, 4(2) :217–231, 1997.

[41] D. Perrin and J.-E. Pin. Semigroups and automata on infinite words. In *Semigroups, Formal Languages and Groups*, pages 49–72, 1995.

[42] L. Polák. Minimalizations of nfa using the universal automaton. *Int. J. Found. Comput. Sci.*, 16(5) :999–1010, 2005.

[43] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2) :114–125, 1959.

[44] D. Revuz. Minimisation of acyclic deterministic automata in linear time. *Theor. Comput. Sci.*, 92(1) :181–189, 1992.

[45] J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.

[46] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3(2) :198–200, 1959.

[47] Ken Thompson. Programming techniques : Regular expression search algorithm. *Commun. ACM*, 11(6) :419–422, June 1968.